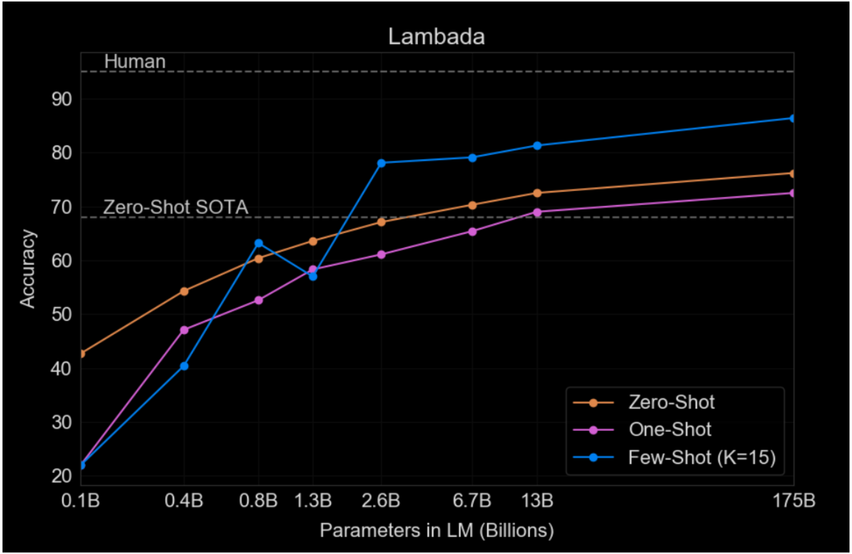
Speculations on Test-Time Scaling

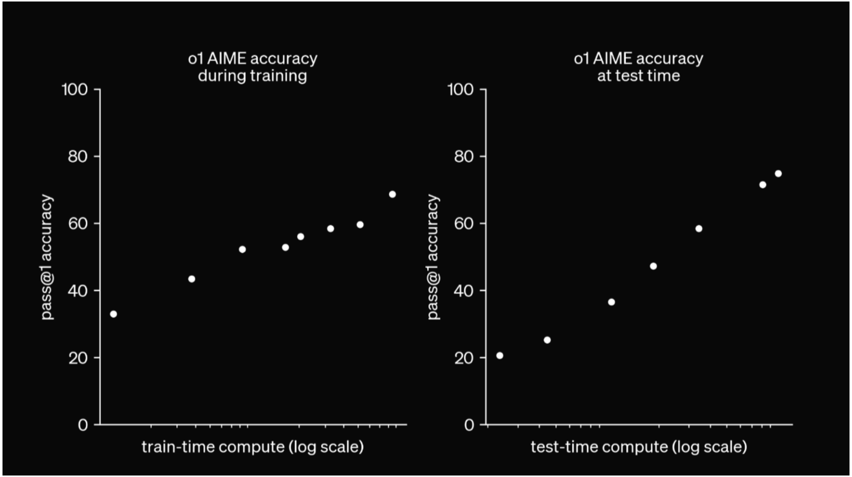
Sasha Rush Daniel Ritter

Cornell

[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

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.

The Bitter Lesson

[Sutton, 2019]

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.

Sources

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

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

Outline

Introduction

The Clues

Technical Background

The Suspects

What do we do now?

o1 Description

[OpenAI, 2024]

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
- 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

o1 Chain of Thought

[OpenAI, 2024]

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.

[OpenAI, 2024]

o1 CoT: Outlining

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).

o1 CoT: Planning

[OpenAI, 2024]

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

Introduction

The Clues

Technical Background

The Suspects

What do we do now?

Technical Background

[Cobbe et al., 2021]

- Formalize sampling of latent reasoning
- Techniques from combinatorial sampling
- No learning yet.

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

[Welleck et al., 2024]

Stepwise CoT Sampling

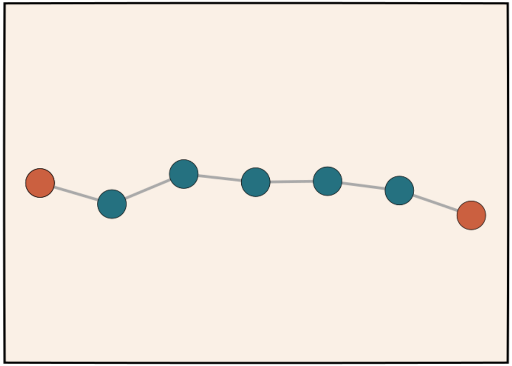
- x; problem specification
- $z_{1:T} \in \mathcal{S}^T$; chain of thought (CoT) steps
- $y \in \mathcal{Y}$; final answer

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

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

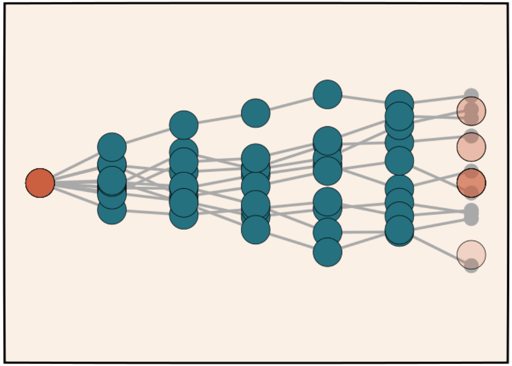
Monte-Carlo Self-Consistency

[Wang et al., 2022]

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 at Training

[Uesato et al., 2022]

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

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]

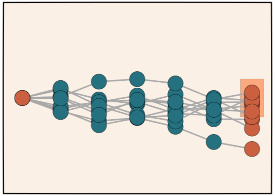
[Nakano et al., 2021, Lightman et al., 2023]

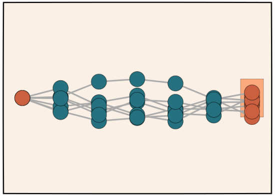
Rejection Sampling
Best-of-N

For $n = 1$ to N:

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

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





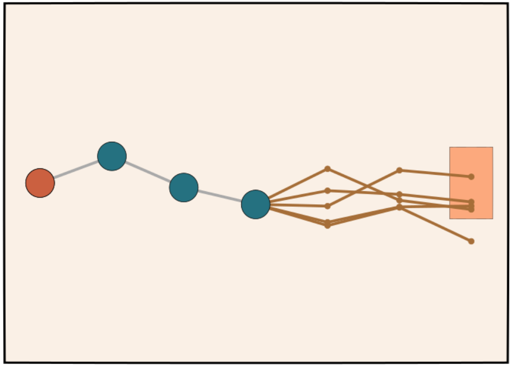
Monte-Carlo Roll-Outs

[Wang et al., 2023]

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

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

Monte Carlo for this expectation.



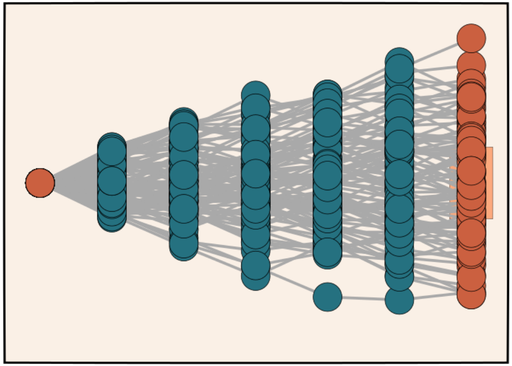
Goal: Learning with Latent CoTs

Maximum likelihood;

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

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

Classic combinatorial expectation



Reinforcement Learning

I will mostly elide RL training question. 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)

Outline

The Suspects

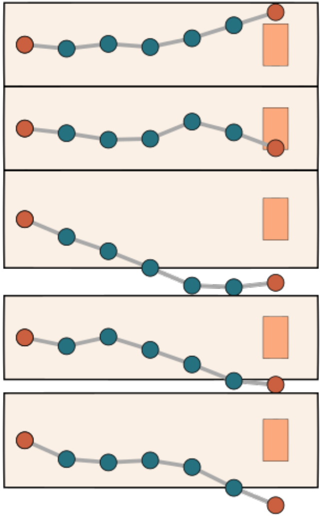
- Guess + Check
- Guided Search
- AlphaZero
- Learning to Correct

The Suspects

- Guess + Check
- Guided 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]

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

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

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

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

- 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]

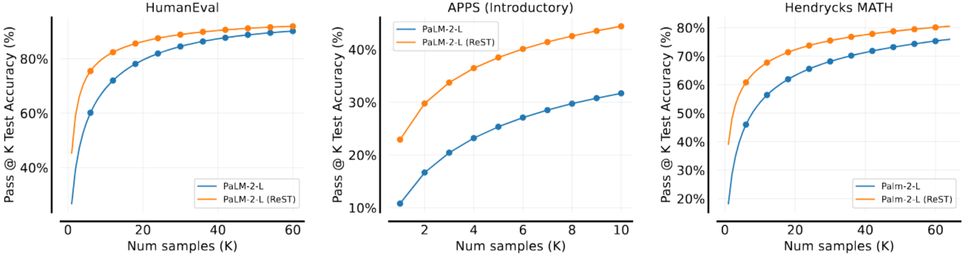


Figure 5 | **Pass@K results** for PaLM-2-L pretrained model as well as model fine-tuned with ReST^{EM}. For a fixed number of samples K, fine-tuning with ReST^{EM} substantially improves Pass@K performance. We set temperature to 1.0 and use nucleus sampling with $p = 0.95$.

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
- Guided Search
- AlphaZero
- Learning to Correct

Suspect 2: Guided Search

- 1) During CoT sampling, use a heuristic to improve trajectories

- 2) Check if final versions are successful

- 3) Train on good ones

Framework: Beam Search with Guide

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

Framework: Beam Search with Guide

[Snell et al., 2024]

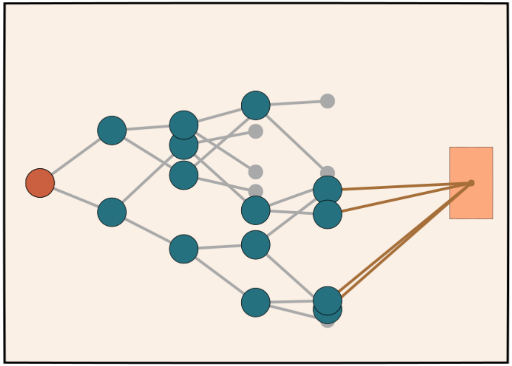
$r : \mathcal{S}^t \to \mathbb{R}$; Guide function

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 $r(z_t)$



Guide Variants

- Monte Carlo Roll-outs
- Learned Value Function (PRM)
- Self-Evaluation Verifier

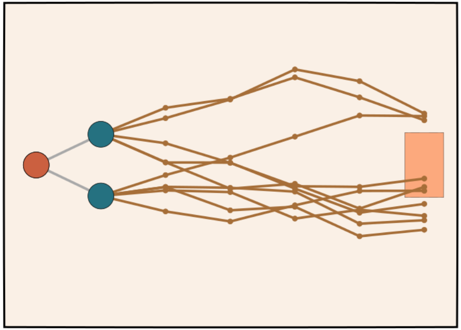
Beam Search with Roll-Outs

[Kazemnejad et al., 2024]

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

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

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

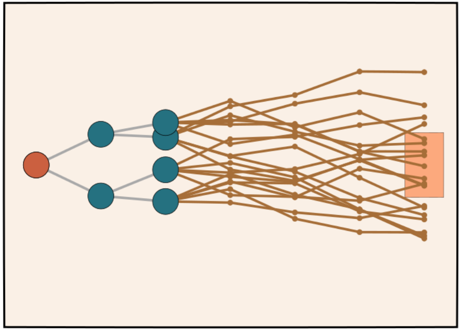


[Kazemnejad et al., 2024]

Beam Search with Roll-Outs

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

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



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

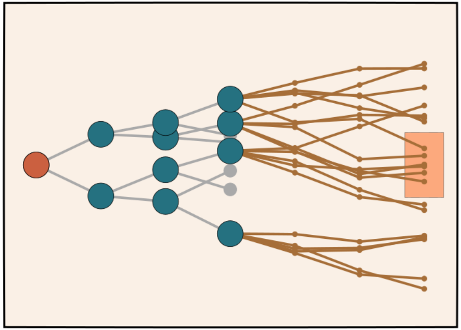
Beam Search with Roll-Outs

[Kazemnejad et al., 2024]

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

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

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



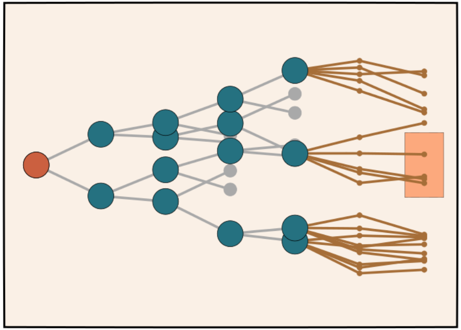
Beam Search with Roll-Outs

[Kazemnejad et al., 2024]

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

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

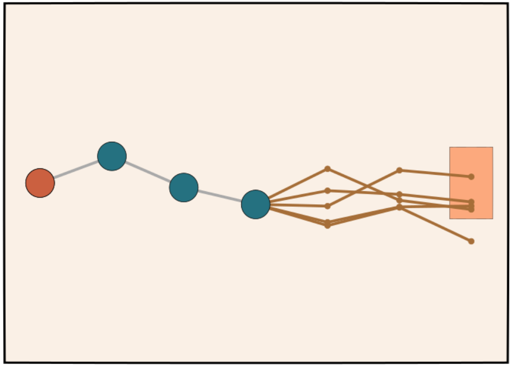
$$r_{MC}(z_t) = \frac{1}{N} \sum_{n=1}^{N} \text{Ver}(y^n)$$



Learned Value Function

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

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



[Xie et al., 2023]

Self-Evaluation

- Evaluation with an LLM
- Prompt LLM to evalute step-level correctness

Variants

- Search Heuristic
- Value Function
- PRM; Process Reward Model
  [Uesato et al., 2022, Lightman et al., 2023]
- PAV; Process Advantage Verifier [Setlur et al., 2024]

[Wang et al., 2023]

Test-time Guides Outperform Self-consistency

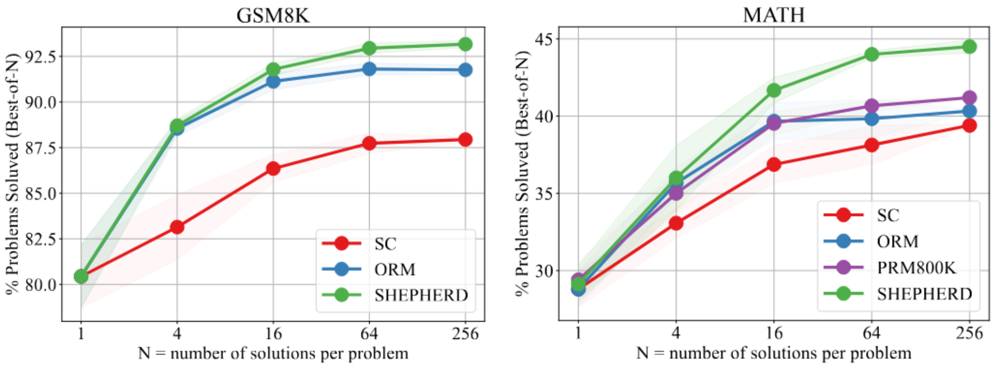


Figure 3: Performance of LLaMA2-70B using different verification strategies across different numbers of solution candidates on GSM8K and MATH.

Is this o1?

- ✓ RS needs to be more efficient.
- ✓ Learned rewards are effective

- × o1 is a single test-time model
- × Not clear if this is enough for planning.

Training Versus Test

- Learned value could be used at test-time
- Alternative can be trained into LLM
- *Generative Verifier* [Zhang et al., 2024]

Let's analyze each option.

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

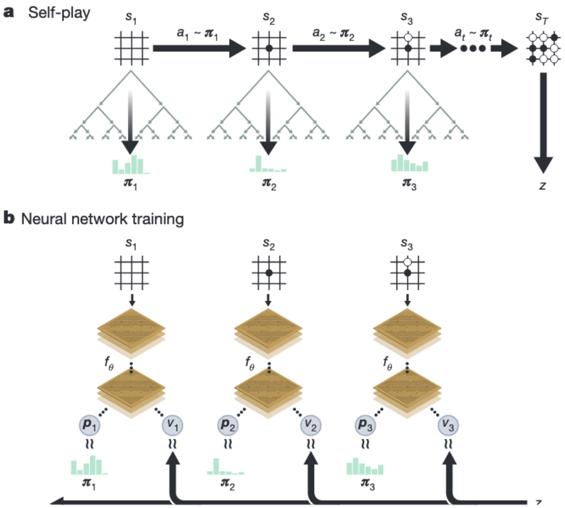
Is that a good explanation? Hmm.

The Suspects

- Guess + Check
- Guided Search
- AlphaZero
- Learning to Correct

[Silver et al., 2017]

Reminder: AlphaZero



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

Suspect 3: AlphaZero

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

Framework: Expert Iteration

[Anthony et al., 2017]

- 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 $\text{Ver}_x(y)$
- Train $p(y, z|x)$, $r(z_t)$ on the labeled samples, and repeat

[Hosseini et al., 2024]

Empirical Results: Expert Iteration

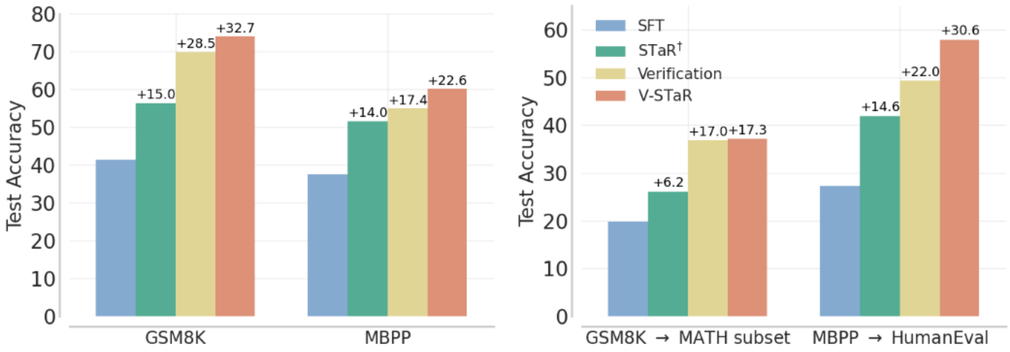
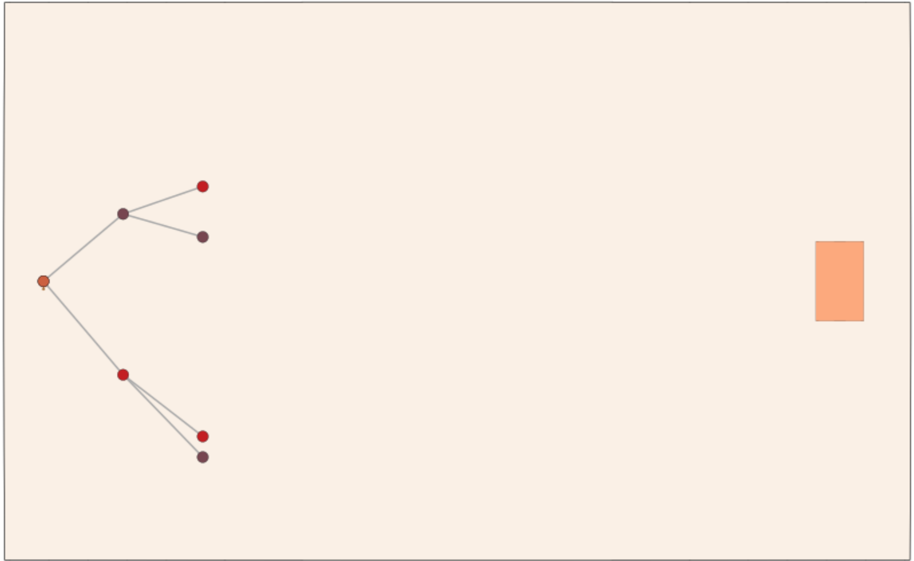


Figure 8: Test accuracy of 13B V-STaR compared to baselines. We report Best-of-64 for verification-based methods and Pass@1 for others. **(Left)** Test accuracy for training tasks. **(Right)** Transfer evaluation of GSM8K and MBPP trained models on MATH subset and HumanEval respectively.

[Kocsis and Szepesvári, 2006]

Generalization: MCTS

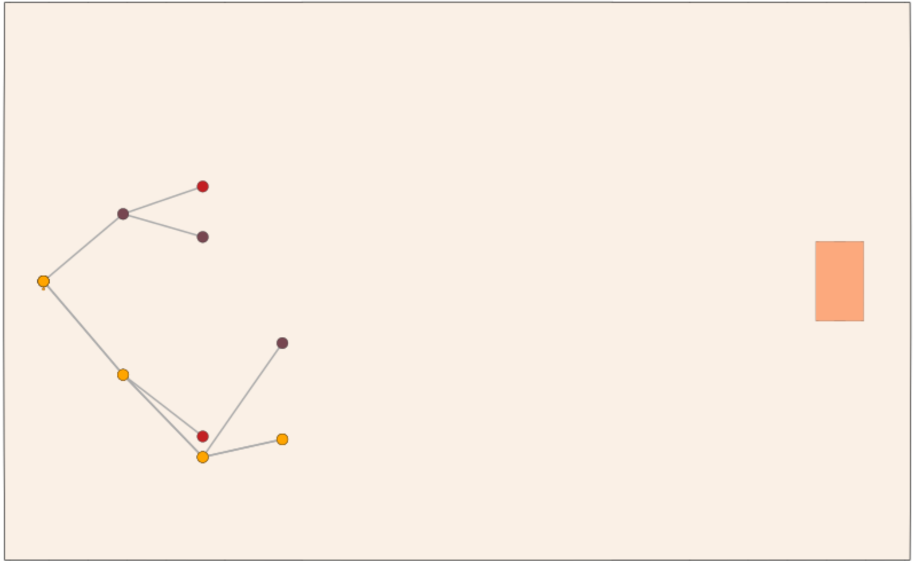


Generalization: MCTS

[Kocsis and Szepesvári, 2006]

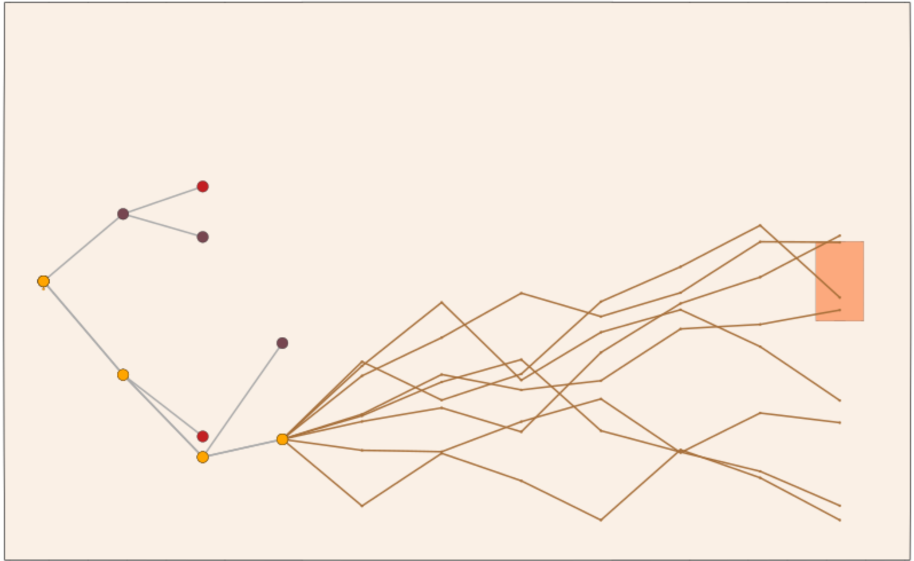
Generalization: MCTS

[Kocsis and Szepesvári, 2006]



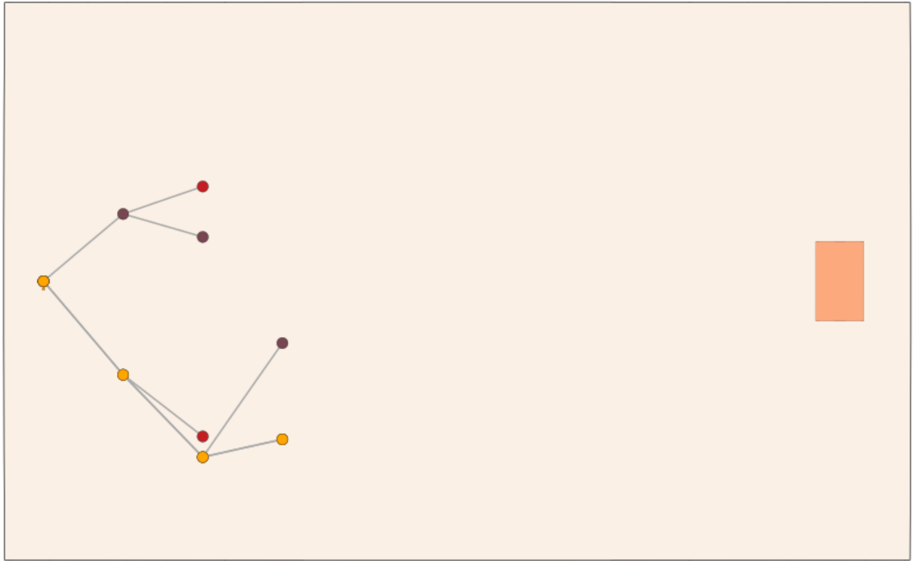
Generalization: MCTS

[Kocsis and Szepesvári, 2006]



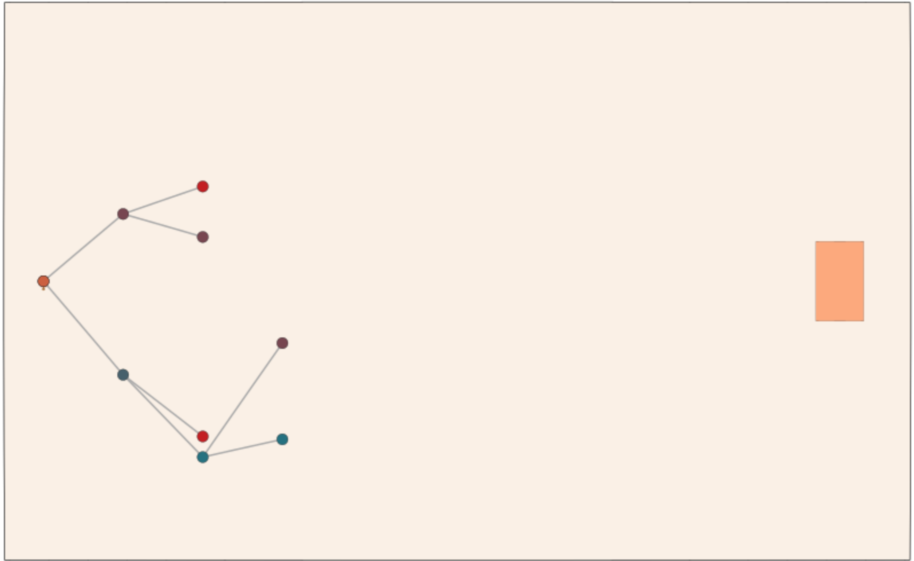
[Kocsis and Szepesvári, 2006]

Generalization: MCTS



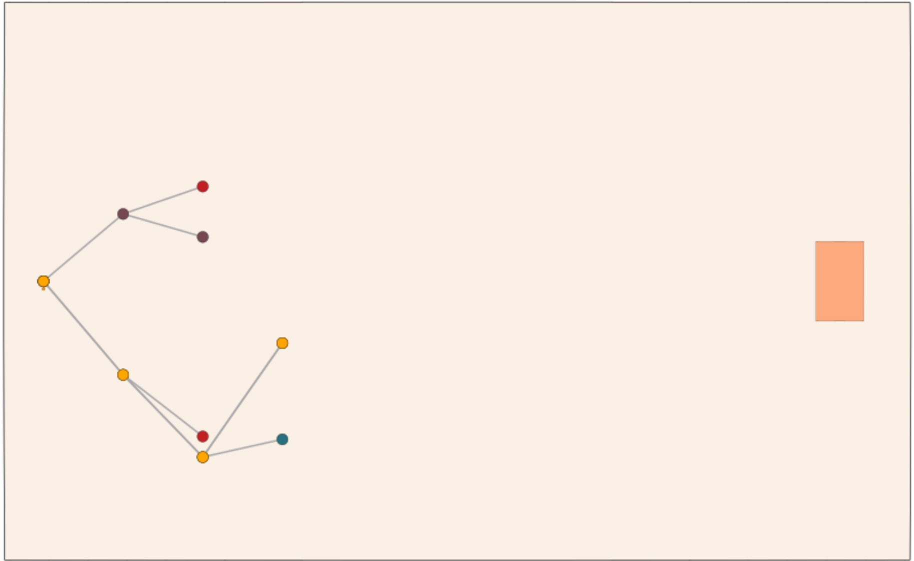
[Kocsis and Szepesvári, 2006]

Generalization: MCTS



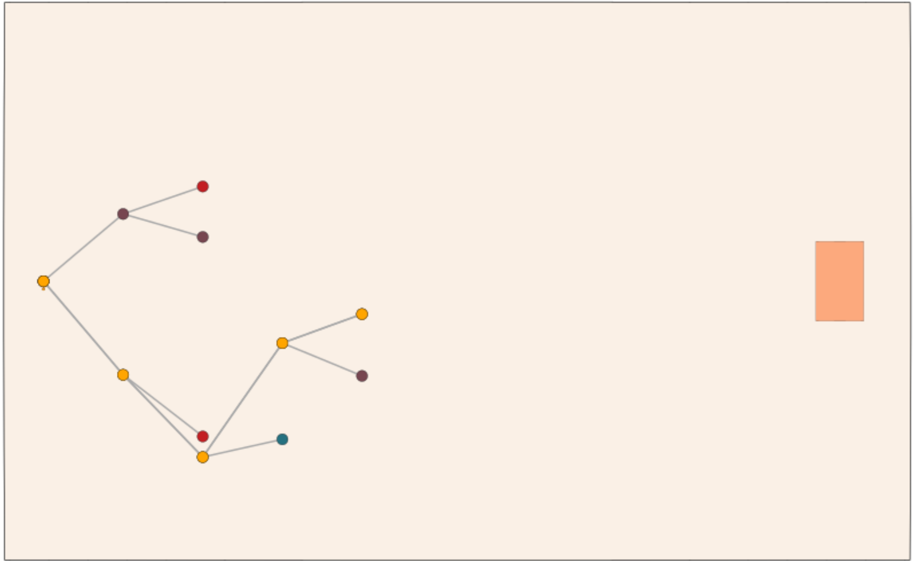
[Kocsis and Szepesvári, 2006]

Generalization: MCTS



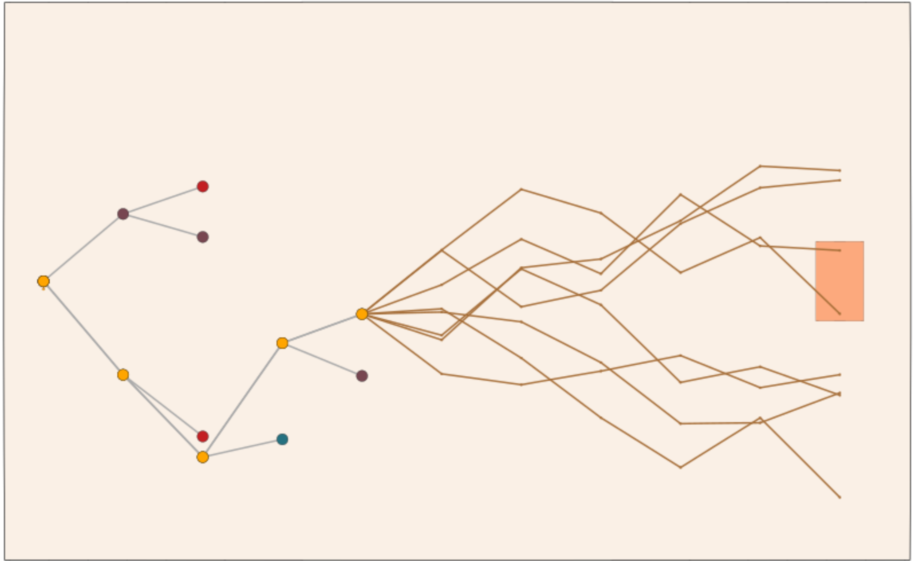
Generalization: MCTS

[Kocsis and Szepesvári, 2006]



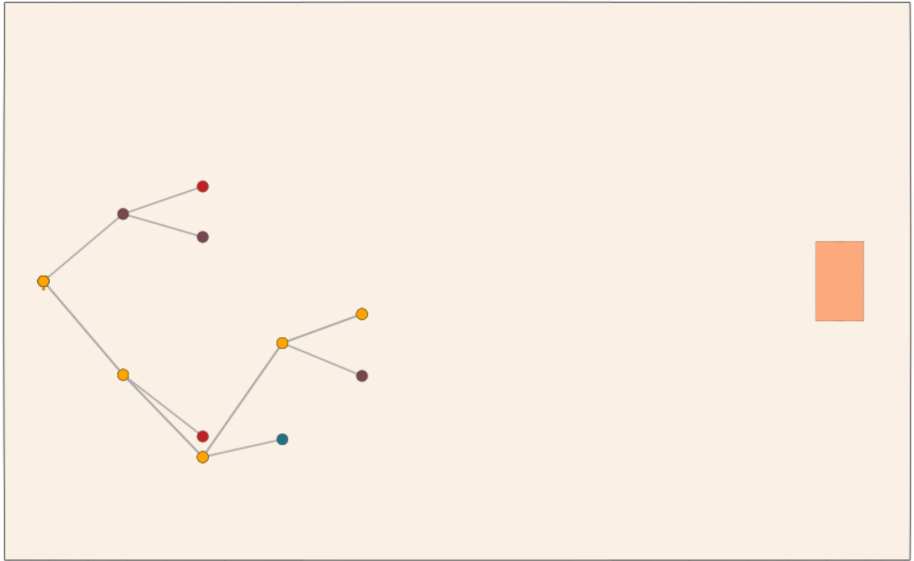
Generalization: MCTS

[Kocsis and Szepesvári, 2006]



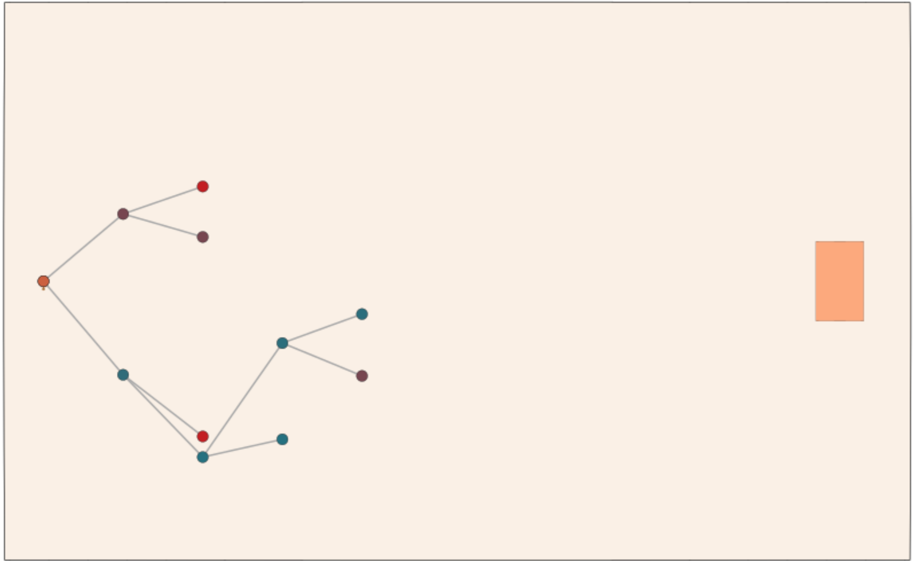
Generalization: MCTS

[Kocsis and Szepesvári, 2006]



Generalization: MCTS

[Kocsis and Szepesvári, 2006]



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

UCB 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 $N(z_{1:t})$ based on results

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

Exploration

- MCTS-UCB explores states

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

- Less strict search process

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

The Suspects

- Guess + Check
- Guided Search
- AlphaZero
- Learning to Correct

What does exploration look like?

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

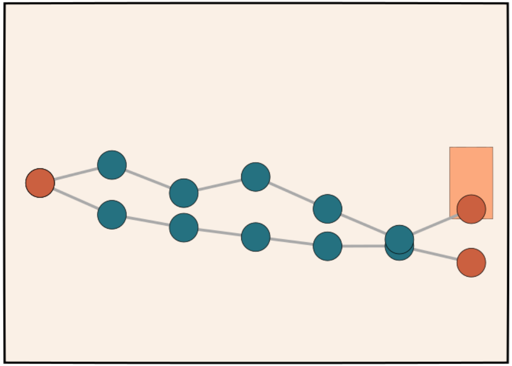
Suspect 4: Learning to Correct

- 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]

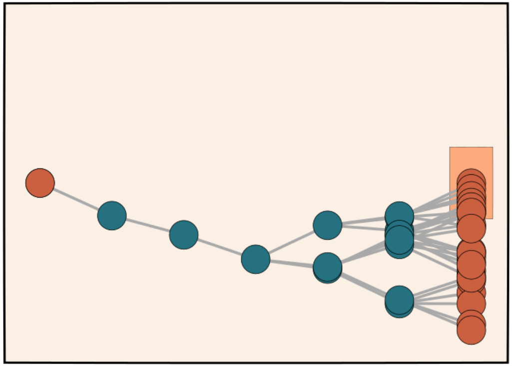
Challenges: Learning Correction

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

RL from Mistakes

[Gandhi et al., 2024]

- Start with z'
- Learn to correct from verifier



Empirical Results

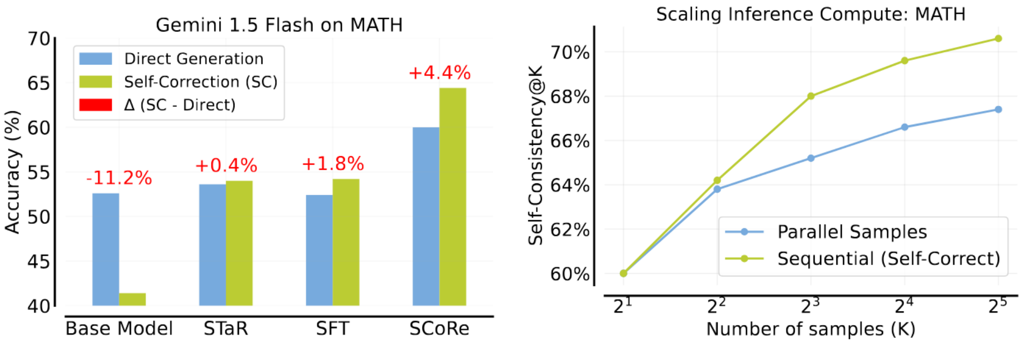
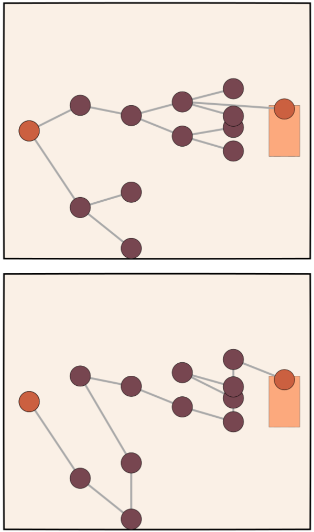


Figure 1 | **Left:** ***SCoRe*** achieves state-of-the-art self-correction performance on MATH; **Right:** ***SCoRe*** inference-time scaling: spending samples on *sequential* self-correction becomes more effective than only on *parallel* direct samples (Section 6.2).

[Gandhi et al., 2024]

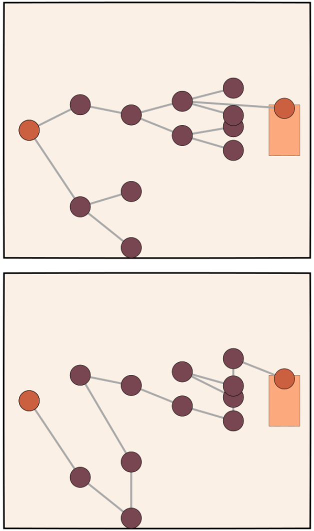
Generalization: Stream of Search

- 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

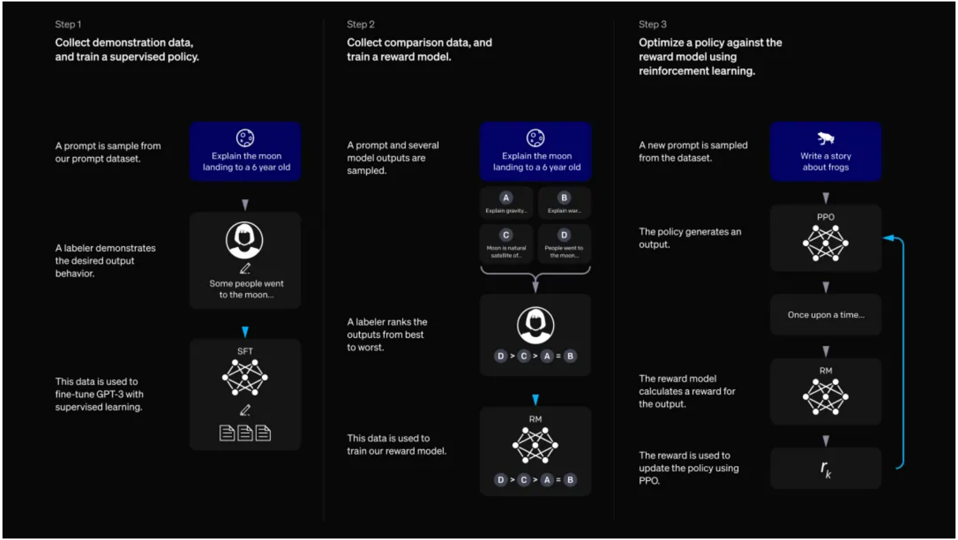
Technical Background

The Suspects

What do we do now?

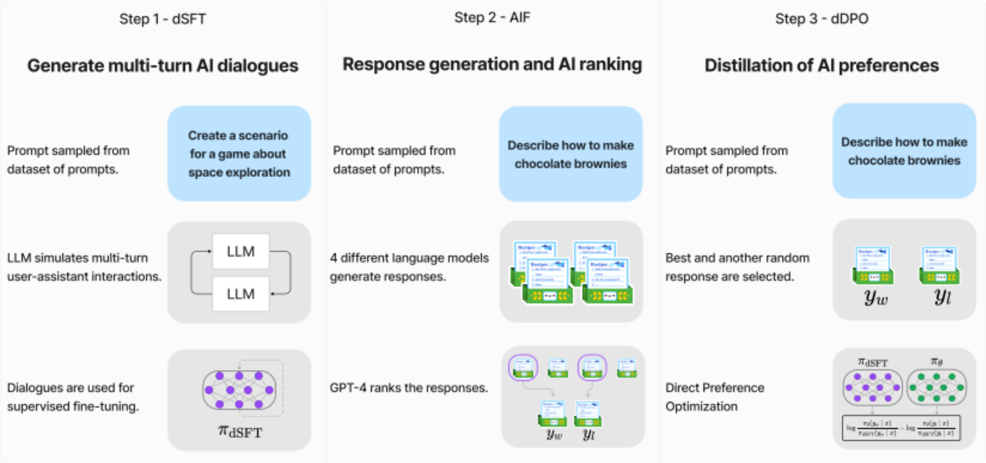
Does it need to be the same?

[Ouyang et al., 2022]



[Tunstall et al., 2023]

Open-Source Models



Let’s Build

- Once result is established there should be a easier path
- Open-Source tools need to be improved to scale these pipelines

Let’s Build

Thank You

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

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 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.

[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].

Reference II

[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].

[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].

Reference III

[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].

[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.

Reference IV

[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].

[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].

Reference V

[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 VI

[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 VII

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

[Ouyang et al., 2022] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022).
Training language models to follow instructions with human feedback.
arXiv [cs.CL], pages 27730–27744.

[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].

Reference VIII

[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].

[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

[Tunstall et al., 2023] Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier, C., Habib, N., Sarrazin, N., Sanseviero, O., Rush, A. M., and Wolf, T. (2023).

Zephyr: Direct distillation of LM alignment.

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.

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].

Reference XI

[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].

[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].

Reference XII

[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].

[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].

[Xie et al., 2023] Xie, Y., Kawaguchi, K., Zhao, Y., Zhao, X., Kan, M.-Y., He, J., and Xie, Q. (2023).

Self-evaluation guided beam search for reasoning.

arXiv [cs.CL].

Reference XIII

[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].

[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].