

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Johns Hopkins University - Fall 2024

CS 601.771 Advances in Self-supervised Models

Elena Kote, Marvin Gao

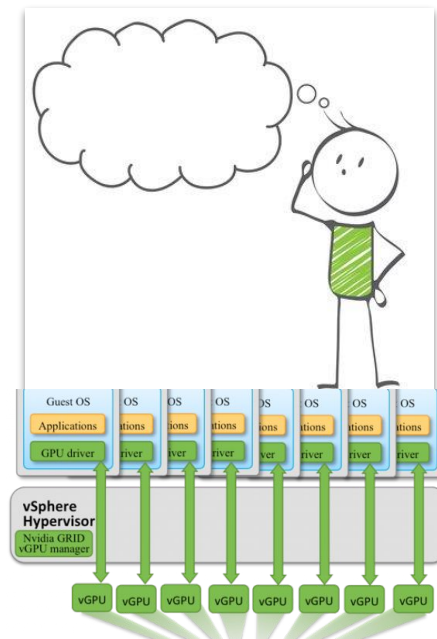
Question Definition

For humans, it's natural to think longer on difficult problems.

How to introduce this capability into LLMs?

Benefits:

- 💪: Improve the performance on the testing phases
- 🕒: Decrease the cost in training and pre-training phases;
- 💰: Is possible to use extra, even spare, computation resource in testing steps to increase overall performance?



Let's Think it Deeper

- 1. **Not only about** better performance on reasoning and inference;
- 2. The model will not need to be very large because of the improvement in testing steps, and we can **deploy smaller size** model in devices but reach the same, even better performance;
- 3. It's a path towards a **general self-improvement** LLMs without / weakly-with human supervision.



Someone has already started

ChatGPT o1-preview ▾

Please mathematically prove that in this world had chicken first and then had eggs



Thought for 8 seconds ^

Mapping the problem

The user seeks a mathematical proof that the chicken precedes the egg. This question traditionally falls under biology and evolution, but a mathematical logical argument might provide a solution.

Revisiting assumptions

I'm assembling a logical map for a mathematical proof to argue that chickens, not eggs, appeared first. Evolution suggests eggs preceded chickens, but I'll define parameters to challenge this idea.

Identifying genetic origins

I'm noting the genetic basis of a chicken, G_C , and how mutation during reproduction from proto-chickens led to the first true chicken, resulting from a non-chicken egg with a chicken embryo.

Taking a closer look

First, I'm proposing that a chicken egg is an egg laid by a chicken. This suggests the chicken came first, then laid the egg. I'm working through this logically and formally to clarify the sequence.

This explanation clarifies that the first chicken emerged due to genetic evolution and was born from an egg, which was laid by a non-chicken, disproving the common chicken-and-egg paradox.

Logical Proof:

- Ancestral Lineage:** Let's denote the immediate ancestors of the first true chicken as *proto-chickens*. These are organisms extremely similar to chickens but not classified as *Gallus gallus domesticus* due to slight genetic differences.
- Mutation Event:** Two proto-chickens mate and produce an offspring. Due to a specific genetic mutation during reproduction, this offspring has the exact genetic makeup to be classified as a chicken (C).
 - Mathematically, if G_P represents the genetic code of proto-chickens and G_C represents that of chickens, then:

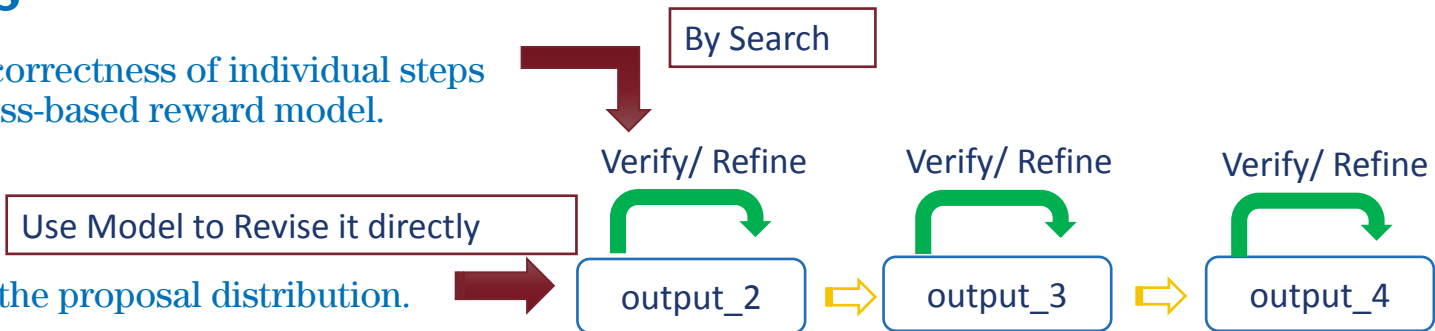
$$G_P + \Delta G = G_C$$

where ΔG is the genetic mutation occurring during reproduction.

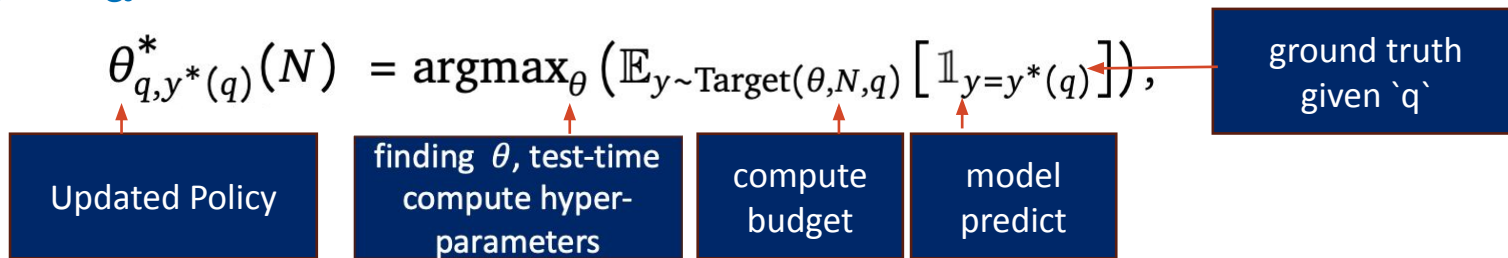
- Egg Origin:** The egg from which this first chicken (C) hatches was produced by proto-chickens. According to our definition, since the egg was laid by proto-chickens (not chickens), it is **not** a "chicken egg" (E) but a "proto-chicken egg."
- First Chicken Egg:** The first true chicken egg (E) is the one laid by the first chicken (C). This egg contains the genetic material of a chicken and is laid by a chicken, satisfying our definition of a chicken egg.
- Conclusion:** Therefore, the sequence of events is:
 - Proto-chickens lay a proto-chicken egg \rightarrow First chicken (C) hatches.
 - First chicken (C) grows and lays the first chicken egg (E).
- Mathematical Representation:**
 - Let $t = 0$ be the time when the first chicken (C) exists.
 - At $t = -1$, only proto-chickens and proto-chicken eggs exist.
 - At $t = 1$, the first chicken egg (E) is produced by the first chicken (C).

Methods

- 1. Verify the correctness of individual steps using a process-based reward model.

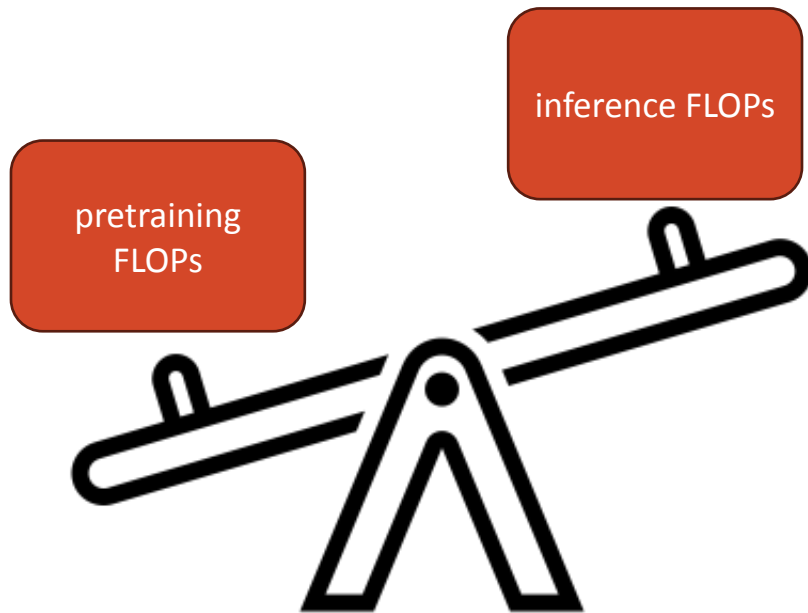


- 2. Improving the proposal distribution.
- What to know: **test-time compute-optimal scaling strategy**



Efficiency Comparison: FLOPs

- comparison between a smaller model with additional test-time compute and pretraining a 14x larger model.
- scaling up test-time computation can be more preferable to scaling up pretraining, but not for all problems;
- For more difficult prompts, it will be less efficient when applying test time scaling up.



Improve Accuracy by a Revisor

ORM: O

Ev

PRM: Pr

Ev

Which is

ORM
Offline Reward
Offline Reward Model

Process
Reward Model

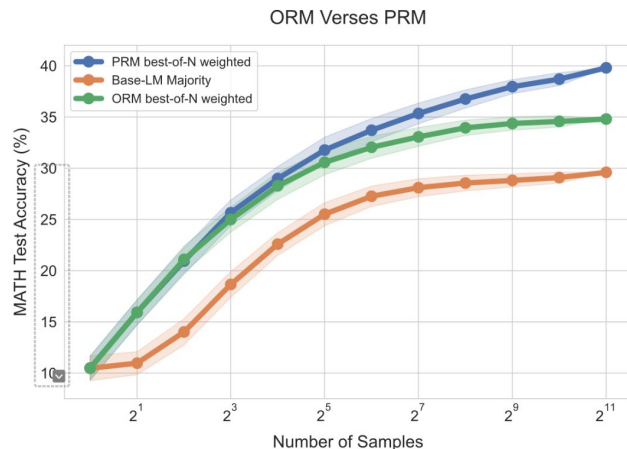


Figure 14 | We compare PRM and ORM models finetuned from PaLM 2-S* in a best-of-N evaluation. We use the PaLM 2-S* base LM to sample outputs, using a few-shot prompt. We see that the PRM greatly outperforms the ORM at a large number of samples.

a single result-based answer

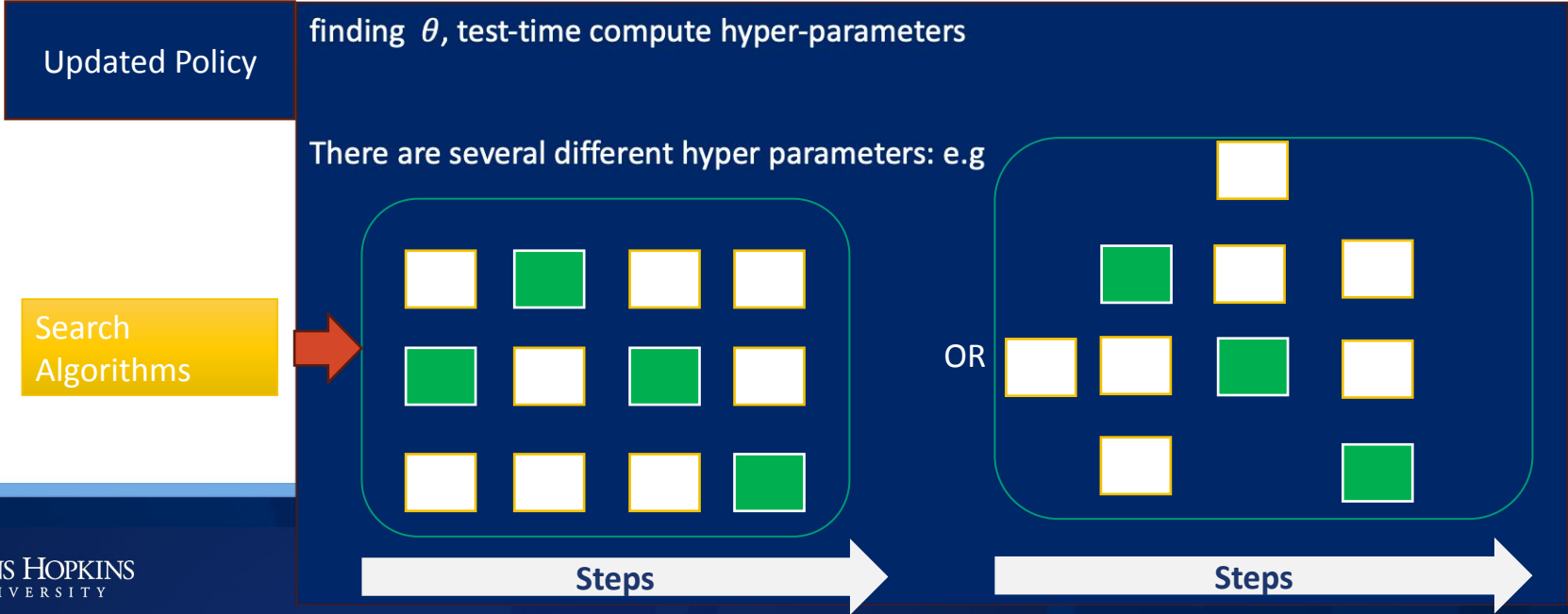


Experiment Settings and Methods

Difficulty of q : assign q to one of five difficulty levels. Use this as input to get optimal parameters settings.
Getting from a LLM

$$\theta_{q, y^*(q)}^*(N) = \operatorname{argmax}_{\theta} \left(\mathbb{E}_{y \sim \text{Target}(\theta, N, q)} \left[\mathbb{1}_{y=y^*(q)} \right] \right),$$

ground truth given `q`



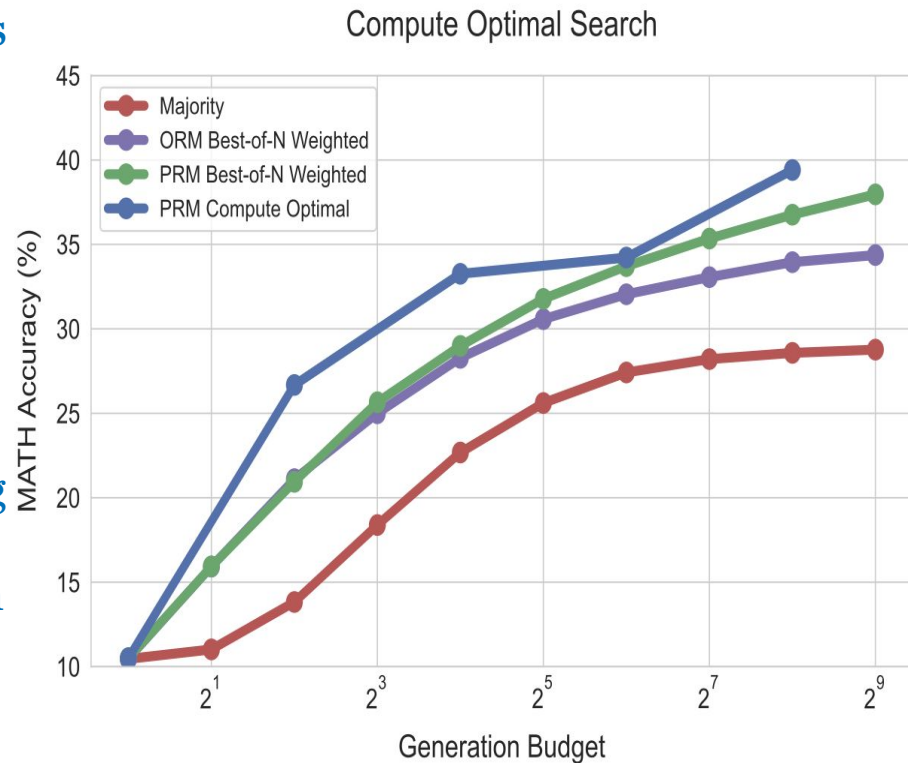
- **Dataset:** MATH benchmark, which consists of high-school competition level math problems.

- **Models:** PaLM 2-S*

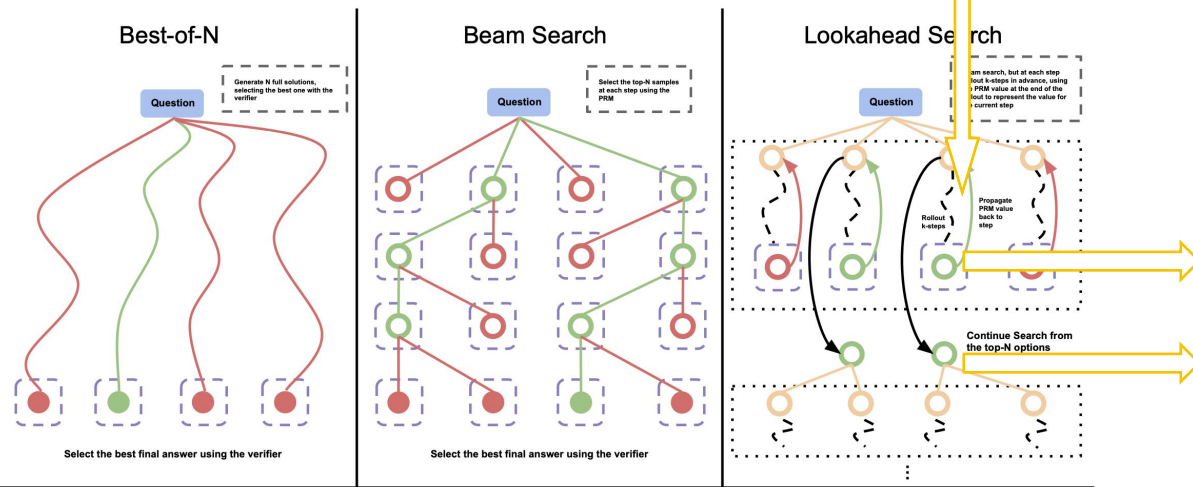
- **Method ½: Scaling Test-Time Compute Via Verifiers**

Step1: Training Process Reward Model: supervise PRMs without human labels, using estimates of per-step correctness obtained from running Monte Carlo rollouts from each step in the solution, corresponding to value estimated.

Step2: Score a set of solutions using PRM.



PRM Search Methods



Key:

- = Apply Verifier
- = Full Solution
- = Intermediate solution step
- = Selected by verifier
- = Rejected by verifier

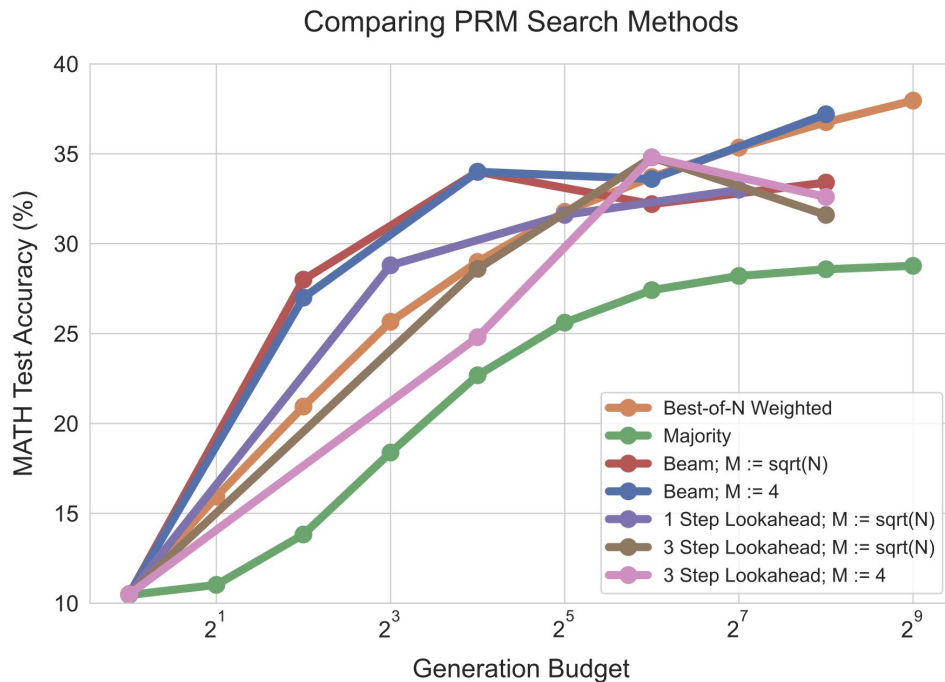
Rollout

Process Reward Model

Select By Verifier and Proceed

Test-time Scaling for Search with Verifiers

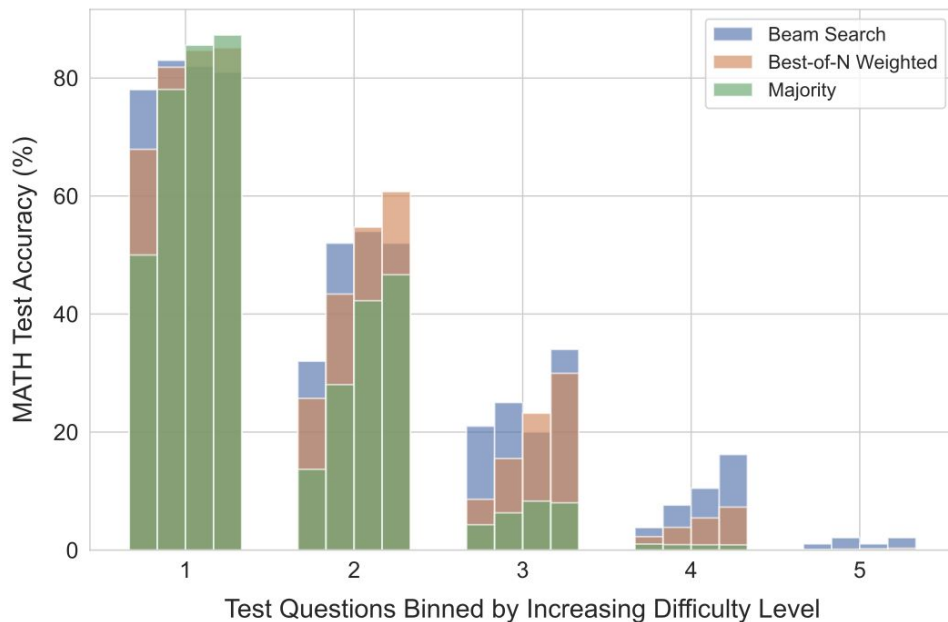
- Sweep various search settings and hyperparameters
- Results
 - Smaller budget: beam search better
 - Greater budget: best-of-n
 - Lookahead underperforms for all budgets



Comparing search methods by question difficulty

- Four bars are increasing test-time compute
- Easy (1-2) : best of N better
 - Beam over-optimizes, exploits PRM signal
- Medium (3-4): beam better
- Intuition
 - On easy questions verifier will mostly be correct, by further optimizing we amplify spurious features thus degrading
 - On difficult questions model less likely to sample correct answer, search helps find it

Comparing Beam Search and Best-of-N by Difficulty Level



Most powerful search methods underperform with high budget on easy questions

If 4 daps = 7 yaps, and 5 yaps = 3 baps, how many daps equal 42 baps?

So 4 daps is equal to 7 yaps, so that means that 1 dap is equal to $7/4$ yaps or 1.75 yaps. [score = 0.21]

And 5 yaps is equal to 3 baps, so that means that 1 yap is equal to $3/5$ baps or 0.6 baps. [score = 0.32]

And we can plug these values in to find the answer. [score = 0.31]

So 42 baps is equal to $42/0.6$ yaps or 70 yaps. [score = 0.60]

And 70 yaps is equal to $70/1.75$ daps or 40 daps. [score = 0.87]

So the answer is $\boxed{40}$. [score = 0.92]

And we're done! [score = 0.91]

And that is our final answer. [score = 0.90]

[score = 0.91]

So 40 daps is equal to 42 baps. [score = 0.90]

And we're done! [score = 0.91]

And that is our final answer. [score = 0.90]

[score = 0.64]

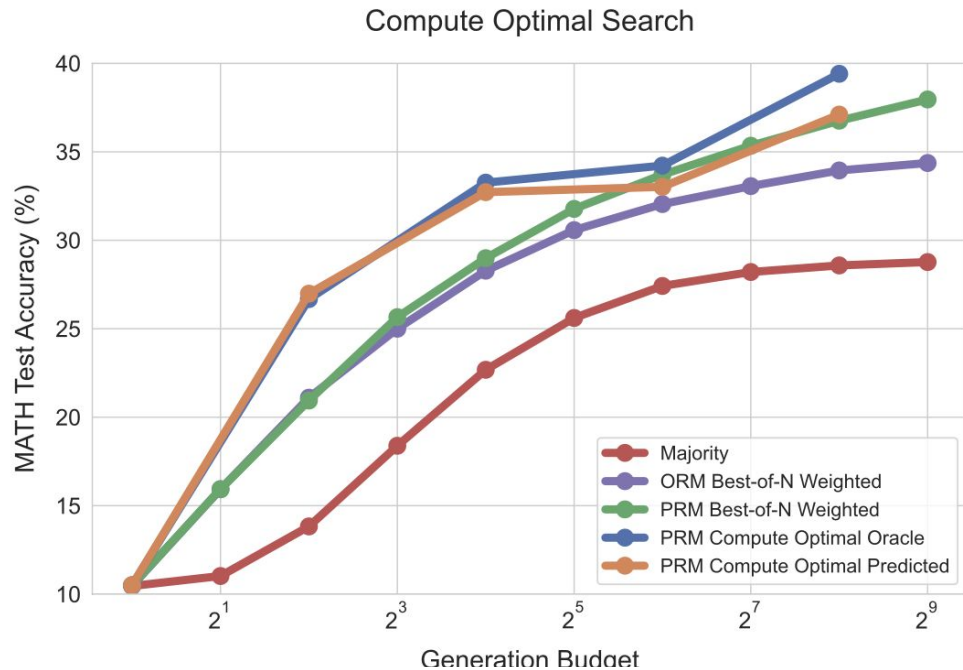
[score = 0.78]

40 [score = 0.83]



- Search causes model to generate low-information repetitive steps at end of solution
- Over-optimizing search can result in overly-short 1 or 2 step solutions

Comparing against baselines

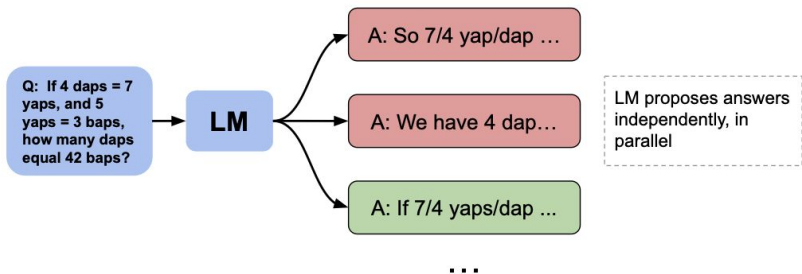


- In both high and low generation budgets compute-optimal scaling can outperform Best-of-N by using **4x less compute**
- Curves for oracle and PRM-predicted difficulty bins generally overlap

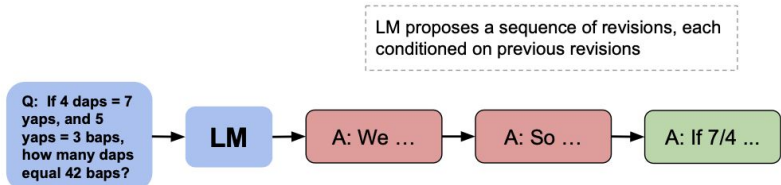
Modifying the Proposal Distribution

- Parallel Sampling
- Sequential Revision
 - Correct answer correlated to incorrect answers
 - Rollout generation
 - Sample in parallel at high temperature
 - Pair up post-hoc
 - Use character edit distance as correlation metric

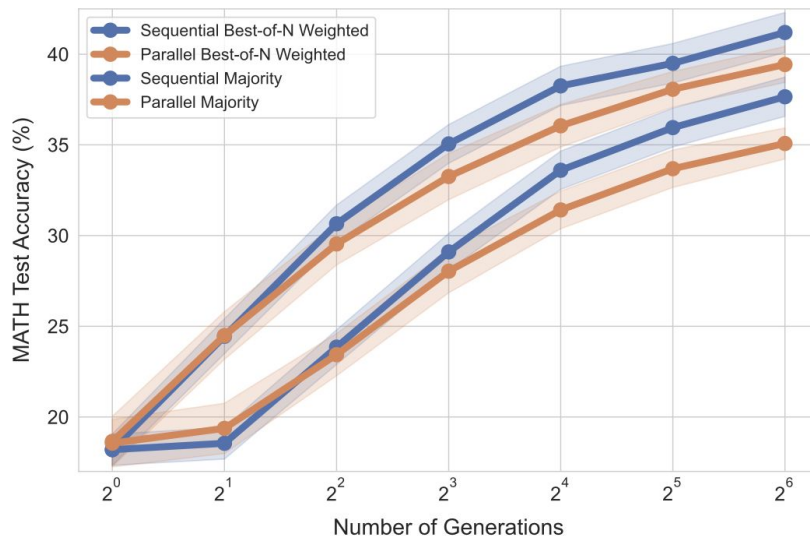
Parallel Sampling



Sequential Revisions



Revision Model Parallel Verses Sequential



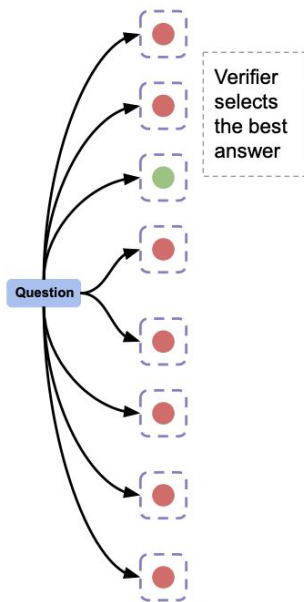
Combining the two strategies

- Each has its pros and cons so why not find a **balance**
- **Parallel**
 - A global search process
 - Give different approaches of solving/answering something
- **Sequential**
 - Local refinement process

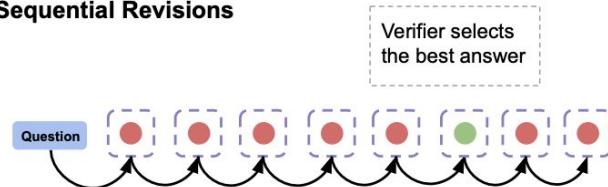
Using Revision Model + Verifier at Inference Time



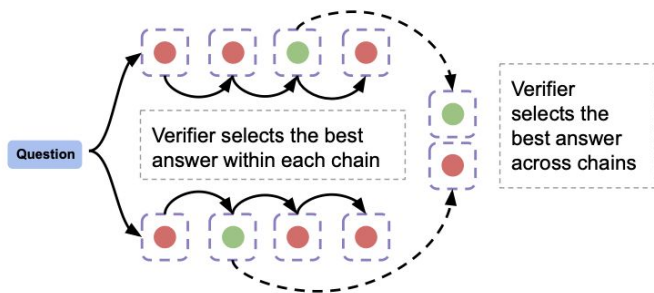
Parallel Best-of-N



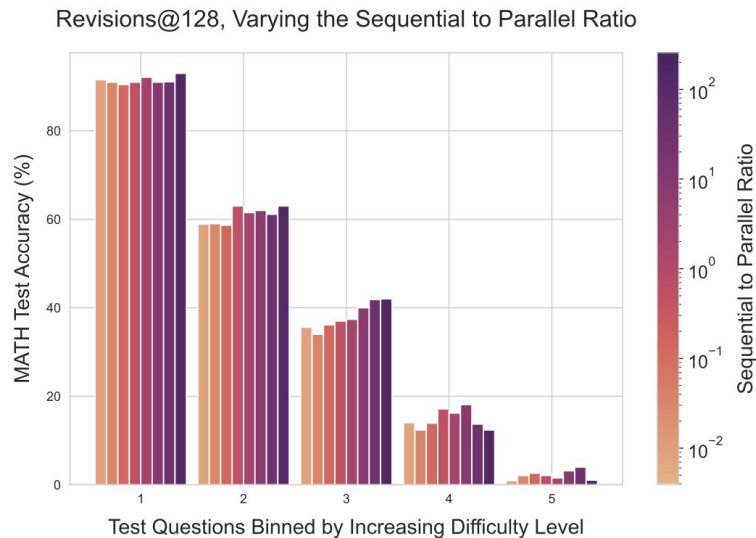
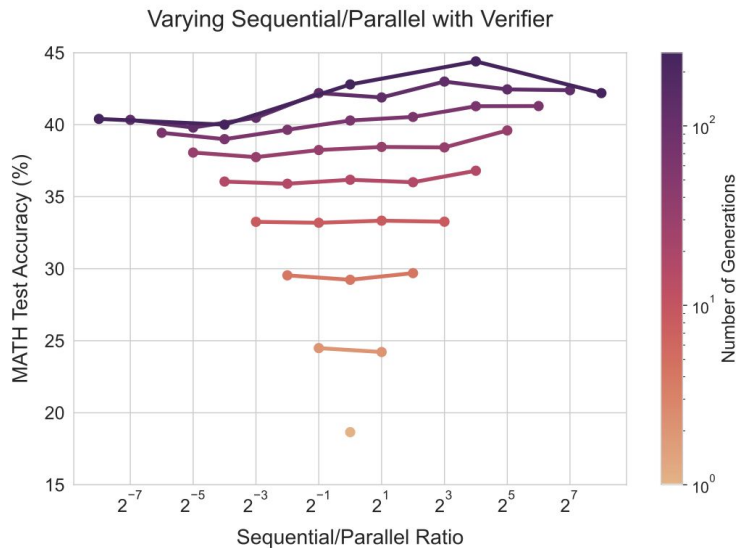
Sequential Revisions



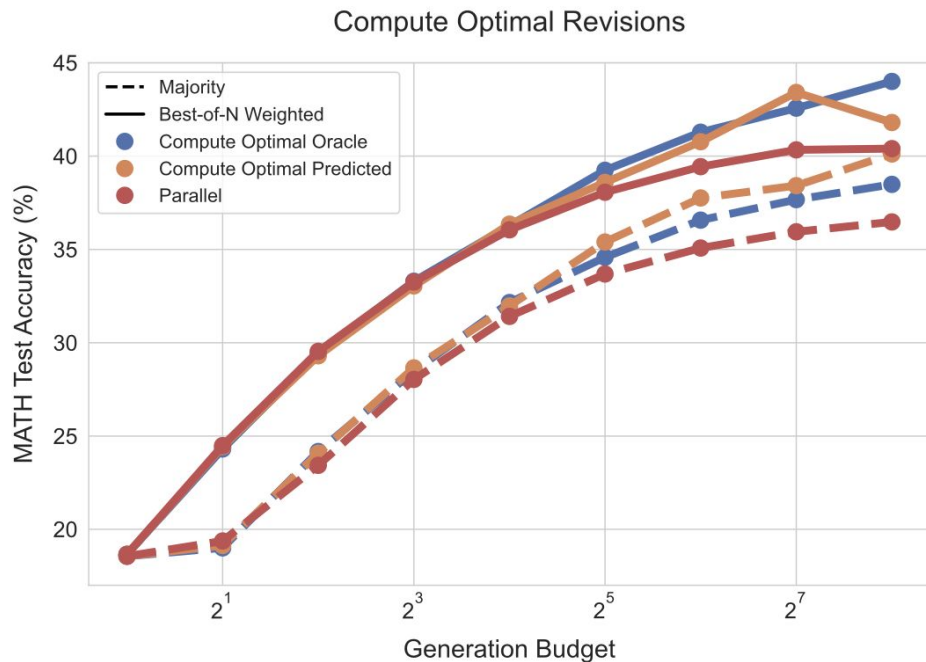
Combining Sequential / Parallel



Easier questions better with full sequential revision Harder questions better with combo of sequential - parallel



Compute-Optimal Revisions vs baseline



- Can still outperform best-of-n by using up to **4x less compute**

Exchanging Pre-Training and Test-Time Compute

Pretrain with X FLOP-s

Run Y FLOP-s on inference

Increase budget by factor of M

N model parameters

D number of tokens used or generated

$$X = 6ND_{\text{pretrain}}$$

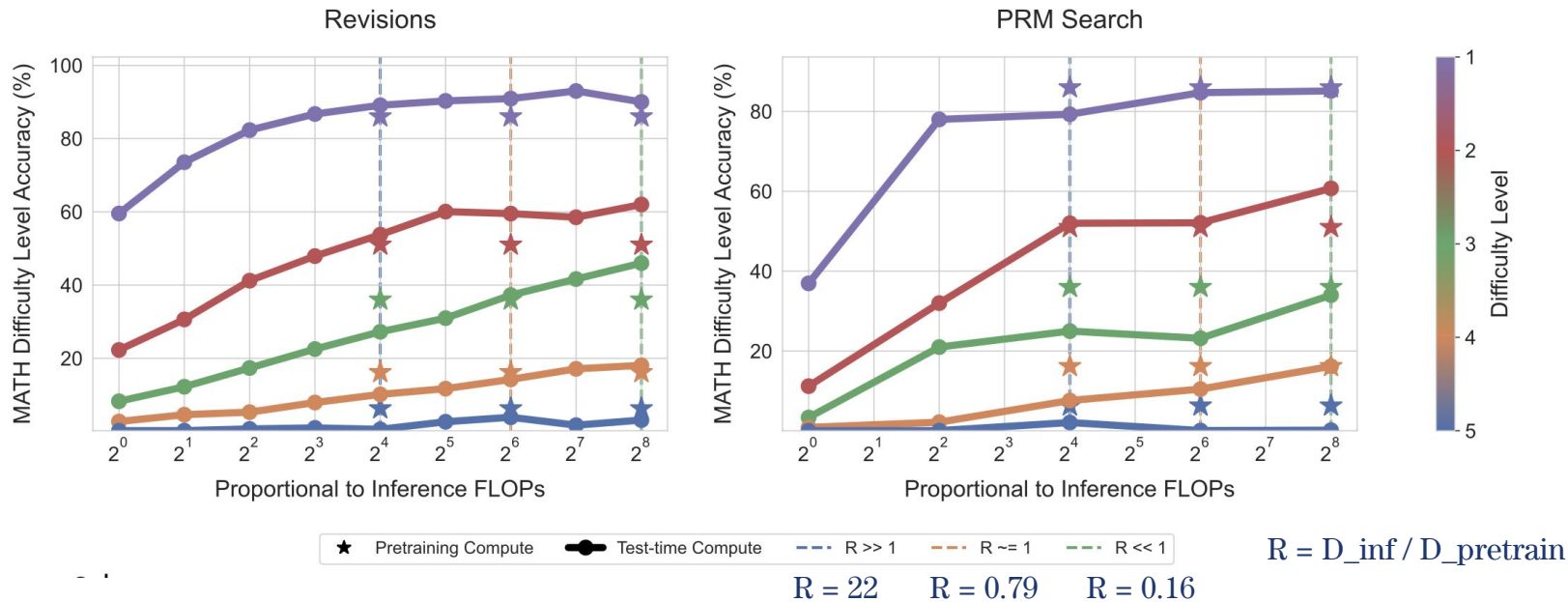
$$Y = 2ND_{\text{inference}}$$

Smaller model's inference compute can match larger pretrained model by a factor of

$$M + 3 (D_{\text{pretrain}} / D_{\text{inference}}) (M-1)$$

$$R = D_{\text{inference}} / D_{\text{pretrain}}$$

Comparing Test-time and Pretraining Compute



- Very difficult questions or high D_{inf} => more budget towards **pretraining**
- Easier questions or lower D_{inf} => more budget towards **test-time compute**

Takeaways



Difficulty of Prompts and Compute Budgets have strong effect on test time optimal efficacy



Simple methods like search and revisions can scale very well on certain types of prompts



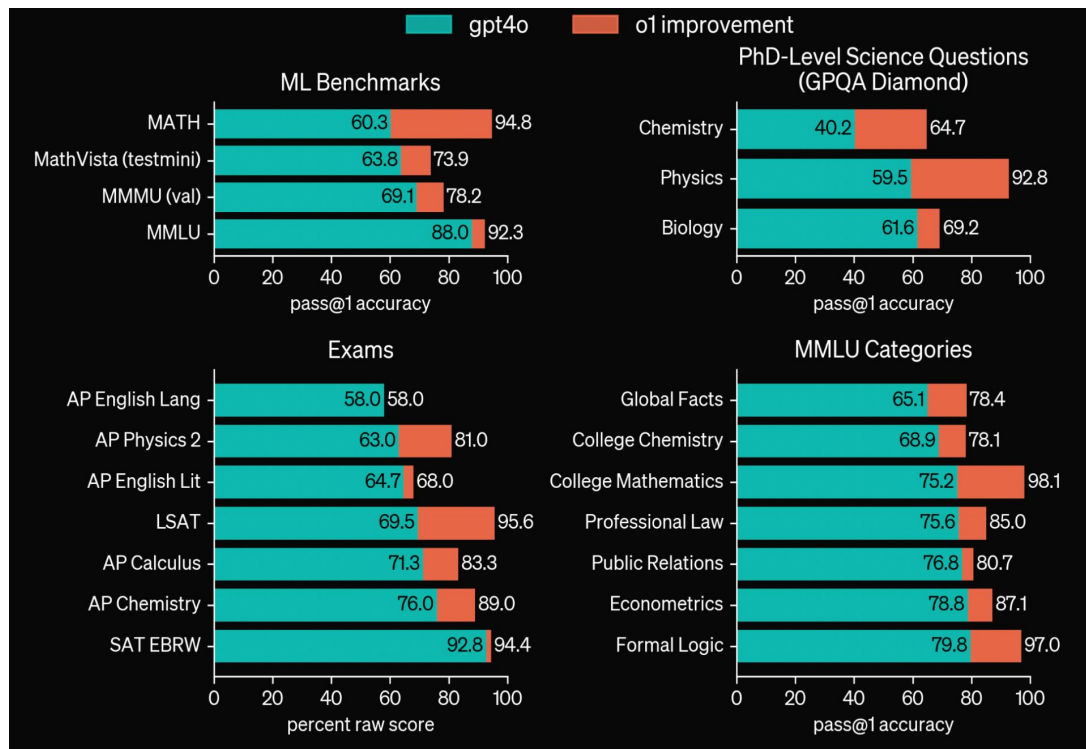
Utilize any resource you can use: By properly using test time computation resources, we can improve the end-2-end performance. (By a factor of 4)



There are a bunch of hyperparameters and variables that need to be accounted for, to get the optimal model

OpenAI o1-preview

- Produces a long internal chain of thought before responding
- Through RL recognizes and corrects its mistakes
- Tries different approach when current isn't working
- Chain of thought reasoning robustly teaches human values and principles



Thanks!

Elena Kote, Marvin Gao