Accessing GPT-4 level Mathematical Olympiad Solutions via Monte Carlo Tree Self-refine with LLaMa-3 8B

ConvAI Reading Group #1

Sumuk Shashidhar

August 29th, 2024

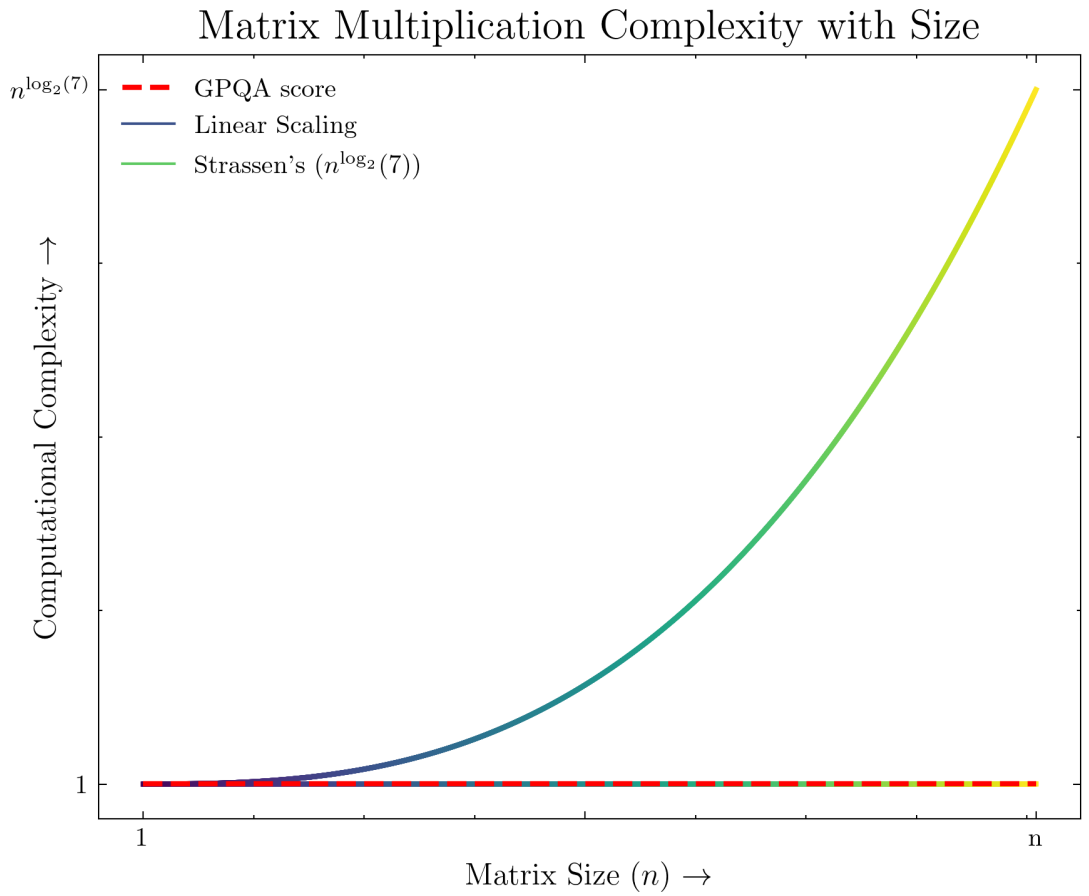
Caveats

1. I'll spend some time talking about why this is a good problem, before I start presenting the paper! 🧐
2. This talk is laden with my personal opinions and ideas, not indicative of the authors' stance. 😊
3. There are graphs, figures, and numbers that are not found in the original paper. 😯

Contents

Why does it make sense to deploy small models?

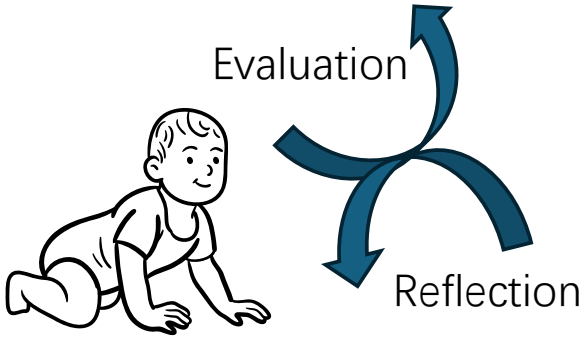
- LLM inference is matrix multiplication.
  - Matrix multiplication complexity: $O(n^{2.807})$ via Strassen's algorithm. [1]
- **Consequence:** Bigger models require exponentially greater amounts of compute to perform.
- However, capabilities are not exponential. They're mostly linear. (e.g., GPQA)
- Smaller GPUs (L4, 4090) are also much cheaper to manufacture, than larger GPUs (H100).

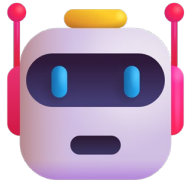


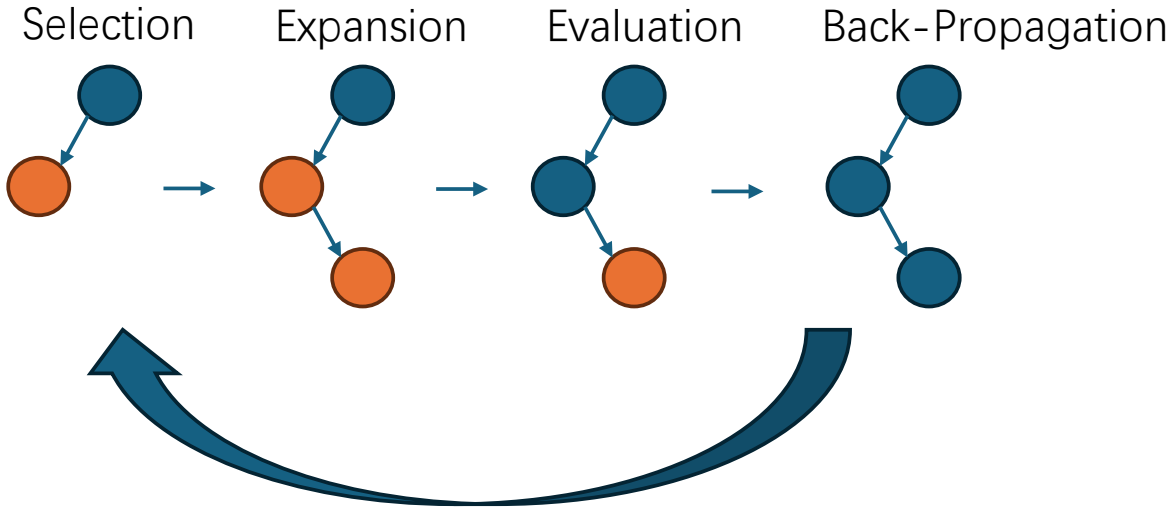
[1] Strassen, Volker. "Gaussian elimination is not optimal." Numerische Mathematik 13 (1969): 354-356.

What is the General Intuition?

- All intelligent life on earth makes good decisions by repeated [action, evaluation, reflection] triples.
- We can use this process with models to elicit better answers.





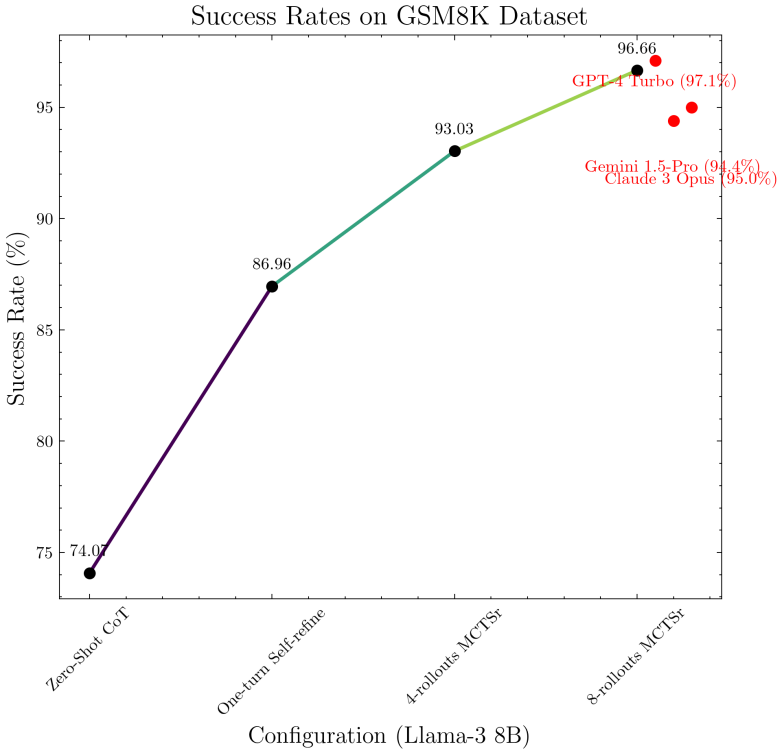


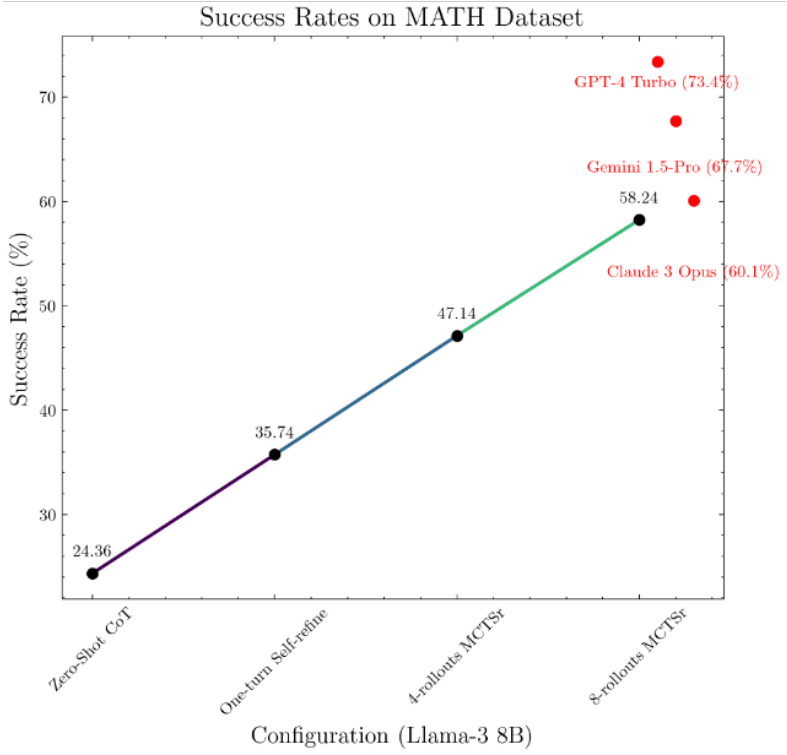
$$UCT_j = \bar{X}_j + C\sqrt{\frac{2\ln N_C}{N_j}}$$

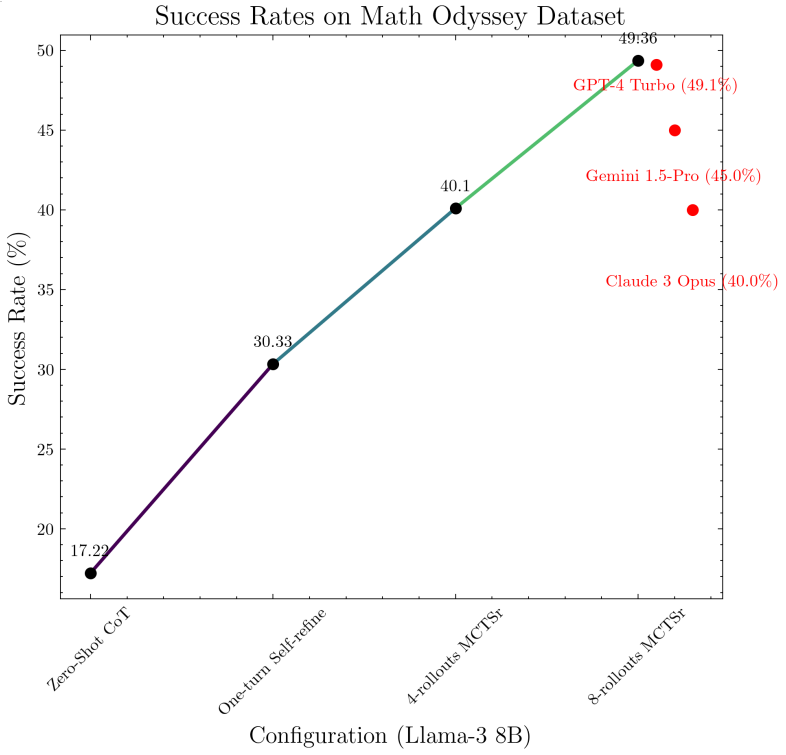
Does this work?

- Yes!
- Here are some interesting results

	Llama 3 8B	Llama 3 8B ***Monte Carlo***	GPT-4 Turbo
GSM8K	74.1	**96.6**	97.1
MATH	24.36	**58.24**	73.4
Math Odyssey	17.2	**49.36**	49.1

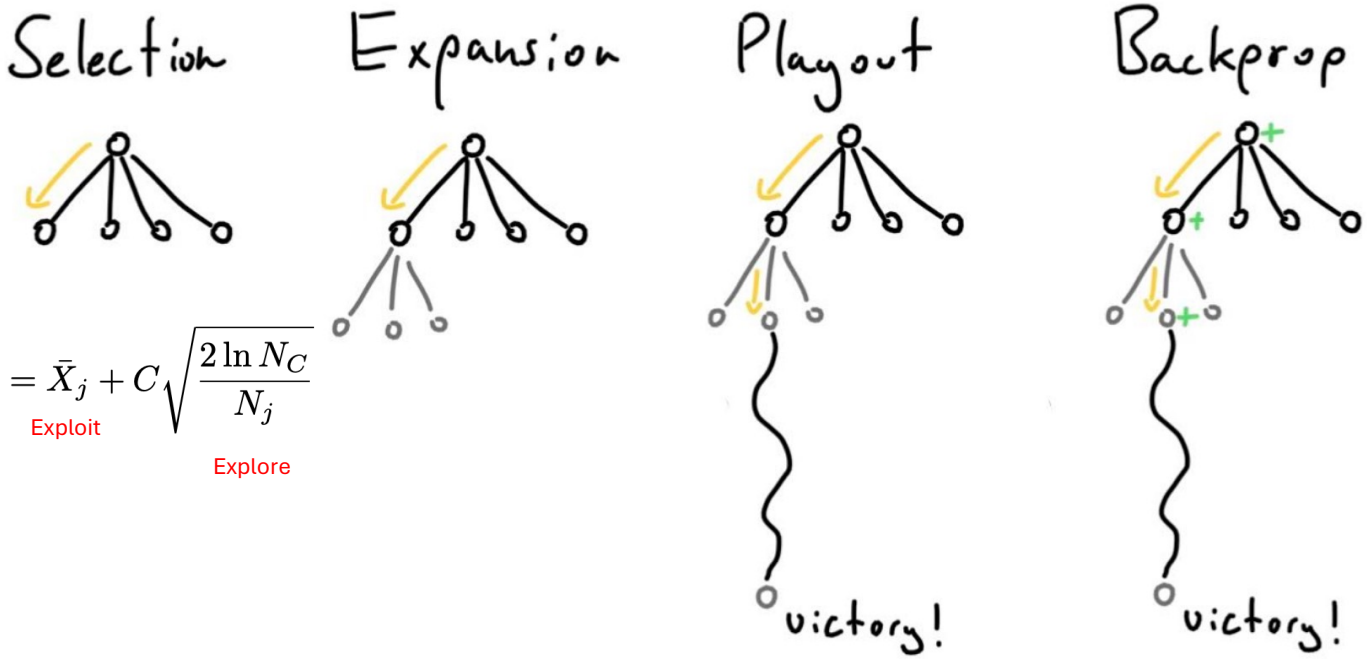






Monte Carlo Tree Search

1. Selection
   - Choose the most promising node / nodes
2. Expansion
   - Add one or more child nodes
3. Playout
   - Simulate a random playthrough
4. Backprop
   - Update the results



$$UCT_j = \bar{X}_j + C\sqrt{\frac{2\ln N_C}{N_j}}$$

Exploit

Explore

Used in the real world to great success (AlphaGo)!

Monte Carlo Tree Search Self Refine (MCTSr)

There are some changes proposed, over the regular MCTS methodology

- *UCT Modification:*
  - **Purpose:** Counteract the tendency of the self reward function.
  - Averages the Reward minimum and the mean, balancing worst case and average outcomes.

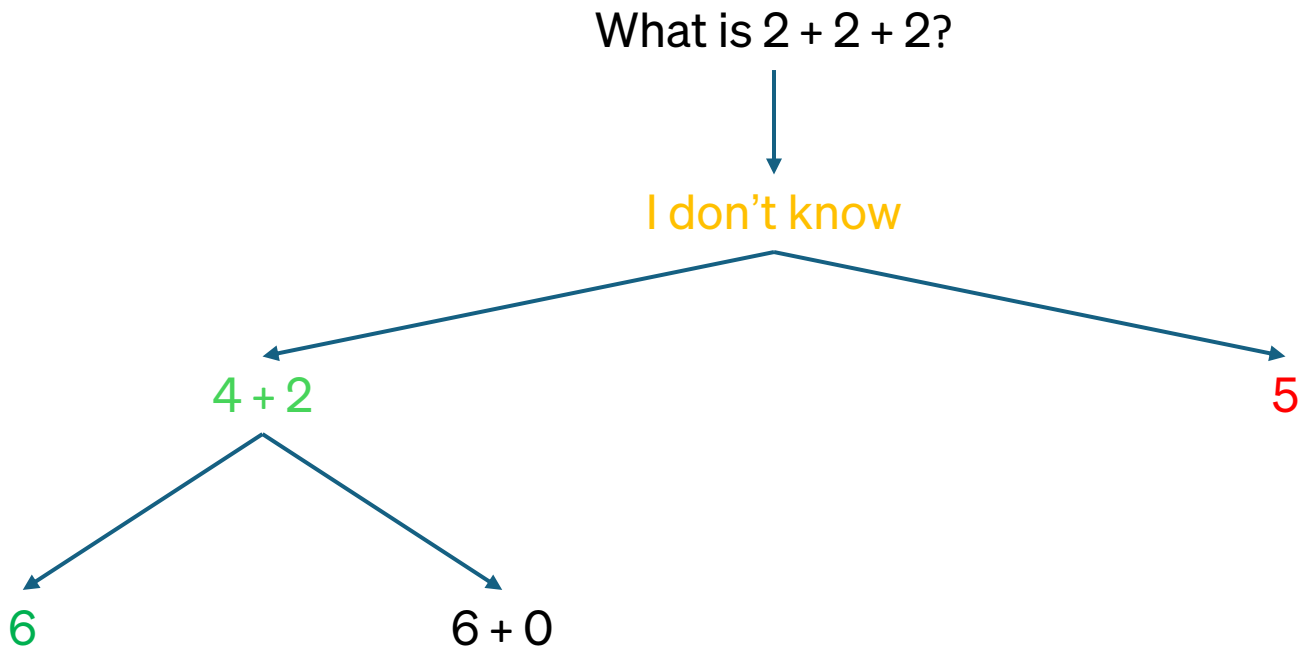
- *Dynamic Pruning:*
  - **Purpose:** Limit Search Space
  - Stop if we reach a maximum number of child nodes.
  - At least one child node has a higher score than the parent.

- *Non-Random Rollout*
  - **Purpose:** Converge to an answer quicker
  - Not random, does more a traditional search, generating new answers.

$$Q(a) = \frac{1}{2}\left(\min R_a + \frac{1}{|R_a|}\Sigma_{i=1}^{|R_a|} R_a^i\right)$$

Here's a short demo about how this works! →

Monte Carlo Tree Search Illustration



- Start with a baseline answer.
- Sample different answers.
- Evaluate sampled answers. Score with the worst case with new Q function.
- Propagate and re-explore. Stop exploring parent nodes if child nodes have higher score (Efficiency)

Don’t models like their own outputs? Doesn’t Self-Refine Overfit?

- The paper proposes three mitigations:
  - Starting with a neutral answer. e.g., “I don’t know” or “I can’t answer that”.
    - Helps against overfitting
  - Hard cap scores at 95, when asking it to evaluate from (-100, 100)
  - Sample evaluations multiple times, then average.

- How do we know that we’re not just retrieving more information from a contaminated benchmark?
  - Math Odyssey was released April 2024, after the release of Llama-3 and other models. Contains new questions.

```
504
505     @retry()
506     def cal_reward(question,ans):
507         query = f'Question: {question}\nAnswer:{ans}\nAnaly
508         ret = generate(query)
509         score = ret[0].split('Score')[-1]
510         scores = pattern.findall(score)
511         if not scores:
512             raise Exception('no')
513         else:
514             ret = float(scores[-1])
515             # if abs(ret - 100.0) < 1e-5:
516             #     ret = 50.0
517             if ret >= 95:
518                 ret = 50
519             # elif ret <= -100:
520             #     ret = -50
521             return ret
```

Limitations

Technique Limitations:

- Slow.
- Expensive.
- Unreliable.

Paper Limitations:

- Lack of result discussion.
  - Where does MCTS not work? Where does it work? Example cases?
  - Why do the different datasets vary so much?
  - Overlap analysis: i.e., are there problems that GPT-4 gets right that MCTS doesn't, and why?
- Descriptions are too abstract about implementation.
- Why did they not choose more rollouts? Is there a plateau?

Future Work

- Prompts in the paper are extremely under optimized!
- Only math datasets have been tested, what about general decision making?
- No discussion about plateau or degradation after multiple rollouts.

A Prompts in Experiment

A.1 Self-Refine

Get Feedback:

USER: Since we have a weak Answer, could you provide me with a relection or feedback to correct this answer better? Analyze this Answer Strictly and Critic, point out every flaw for ervery possible imperfect to minus every possible score!
Let's think step by step.

Get Refined Answer:

USER: Please refine the your answer according to your Reflection or Feedback. The response should begin with [reasoning process]...[Verification]... and end with end with "[Final Answer] The answer is [answer formula]"
Let's think step by step.