

Visual Test-time Scaling via Search

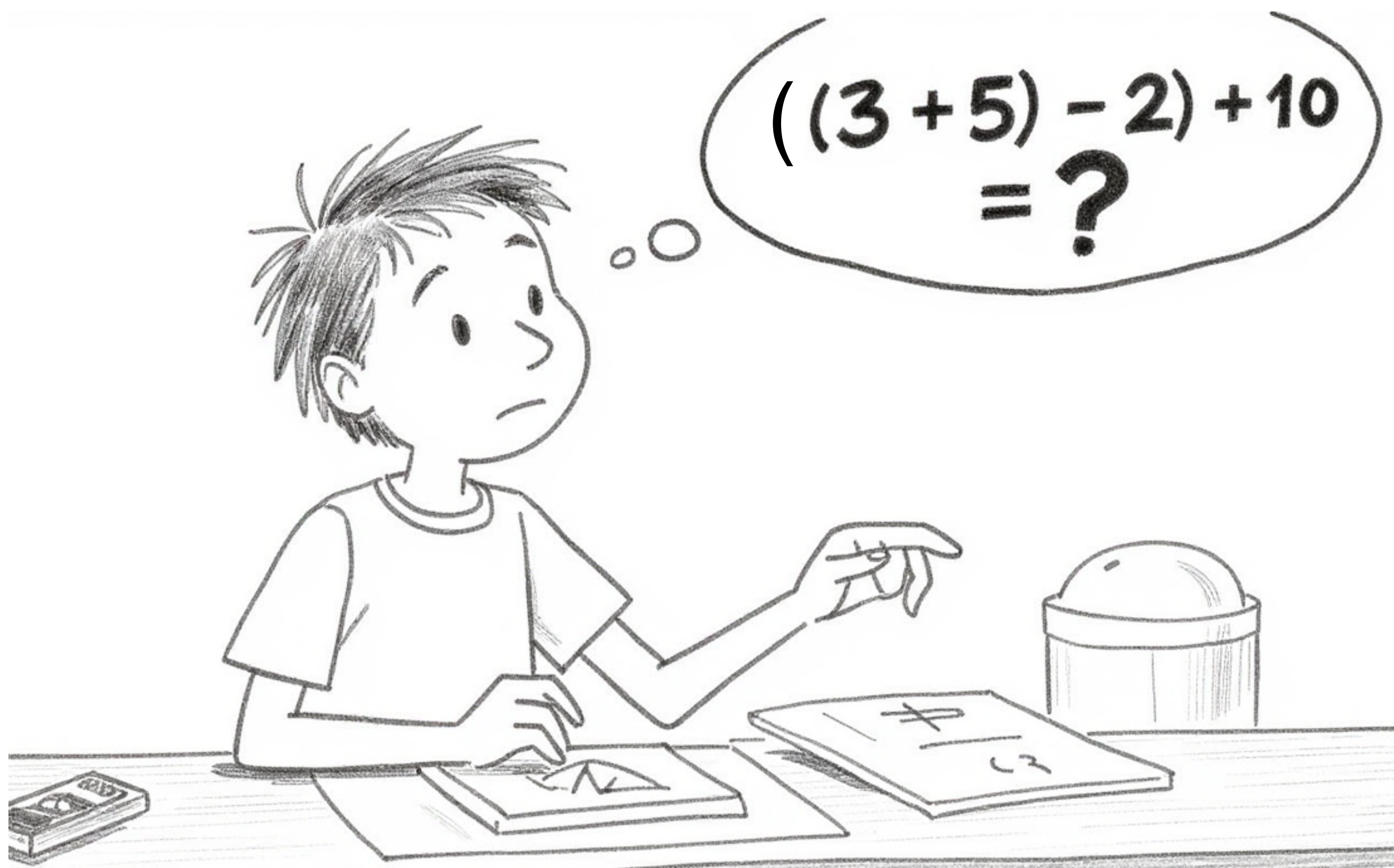
Student: Zhitong Gao

Advisor: Prof. Amir Zamir

EDIC Candidacy Exam

Date: 05.08.2025 | Location: BC 229



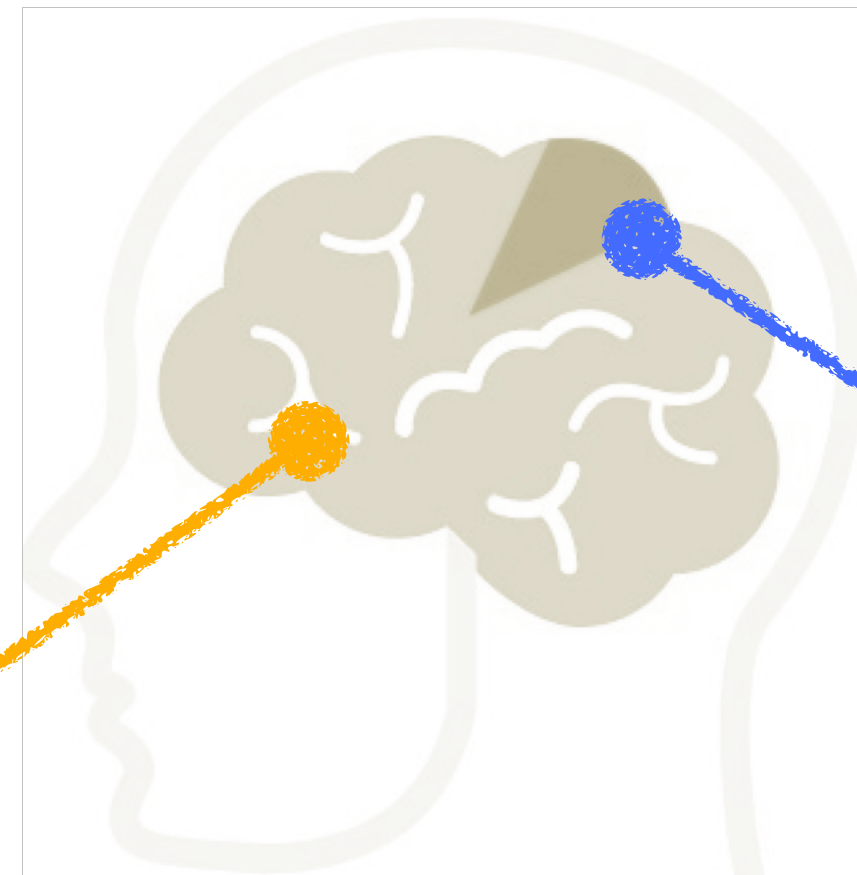
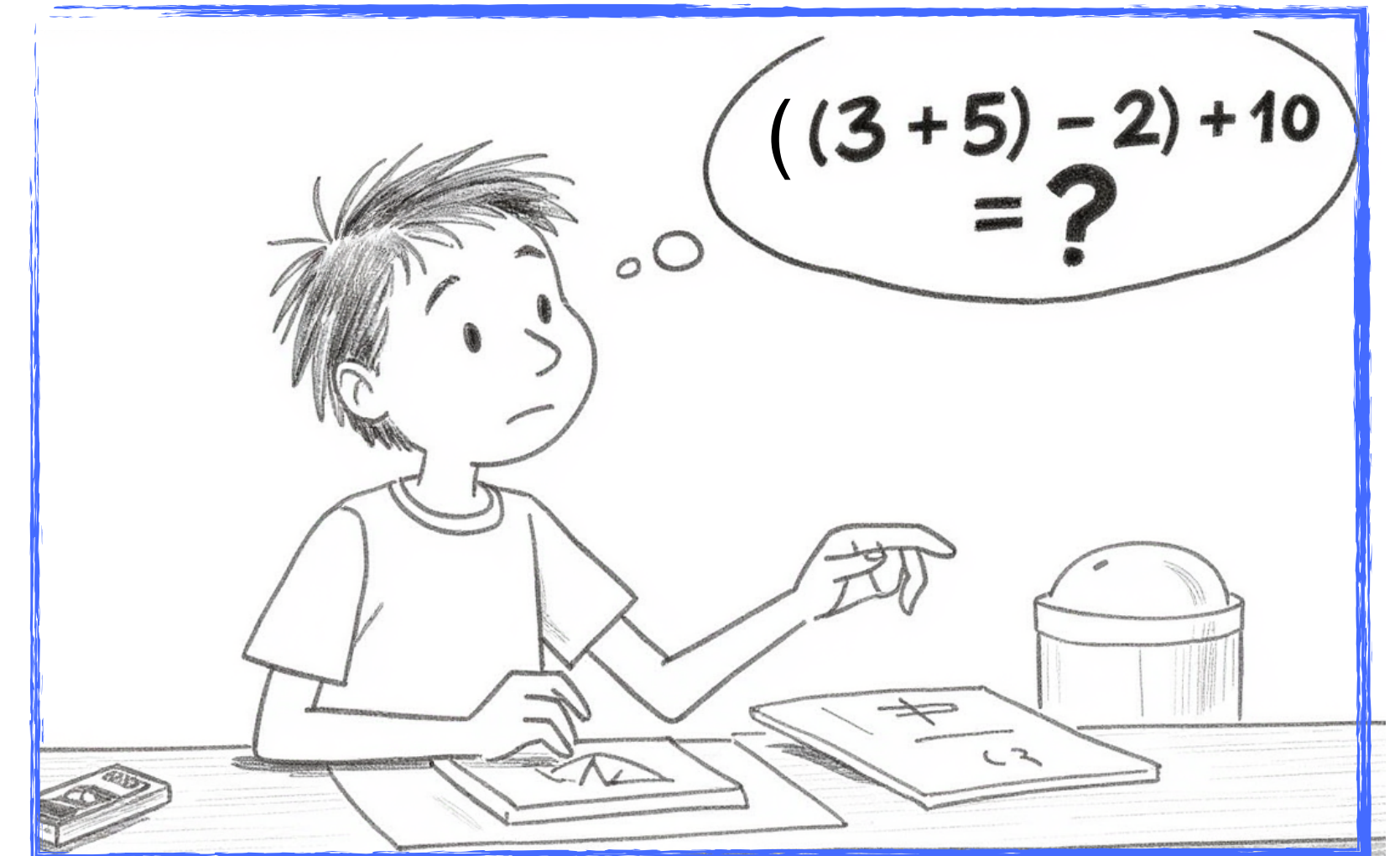


Motivation Two Systems in Human Cognition ^[1]

System 1: Fast Thinking

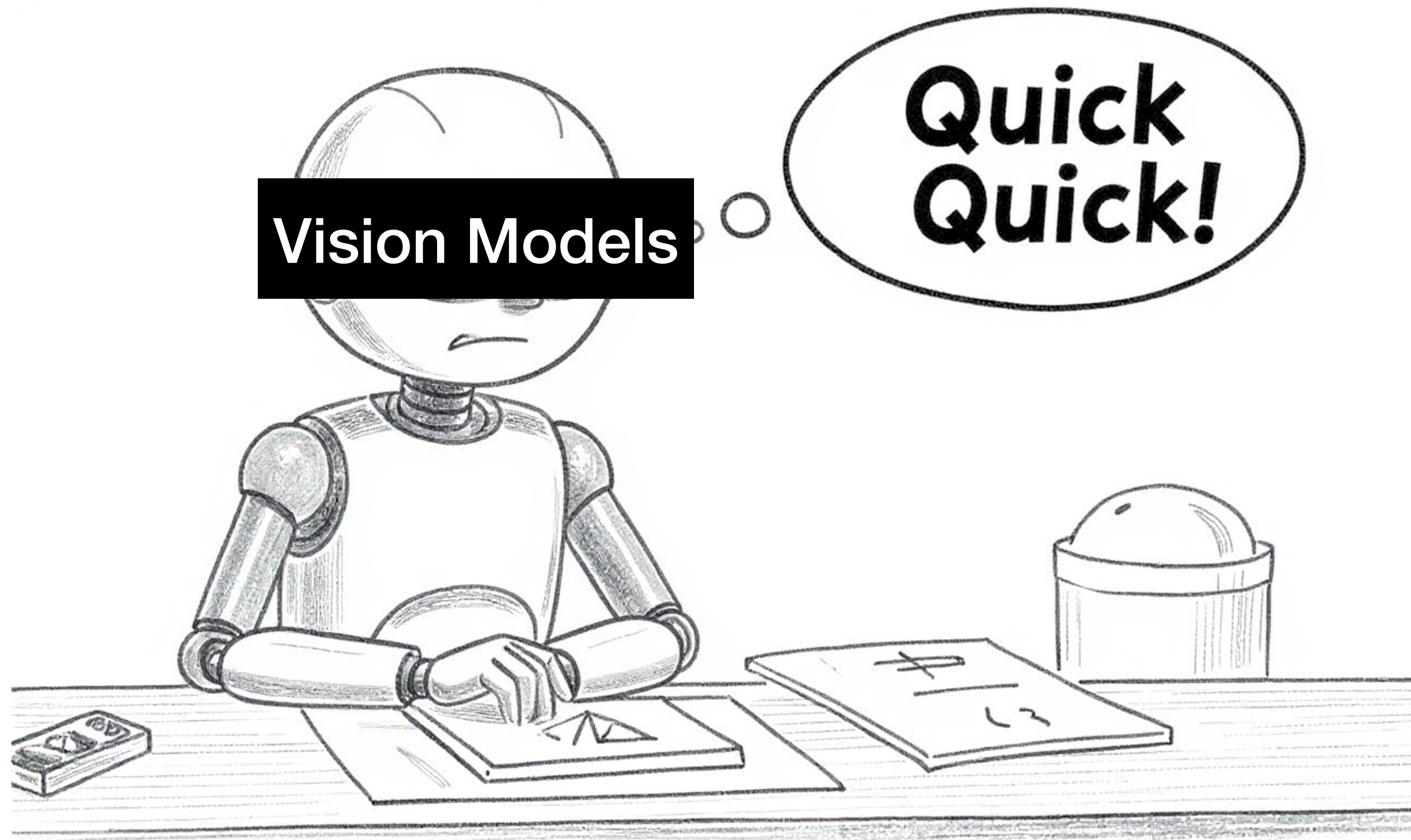


System 2: Slow Thinking



(Both images generated by FLUX.)

Challenges ML models often lack the ability for “Slow Thinking”

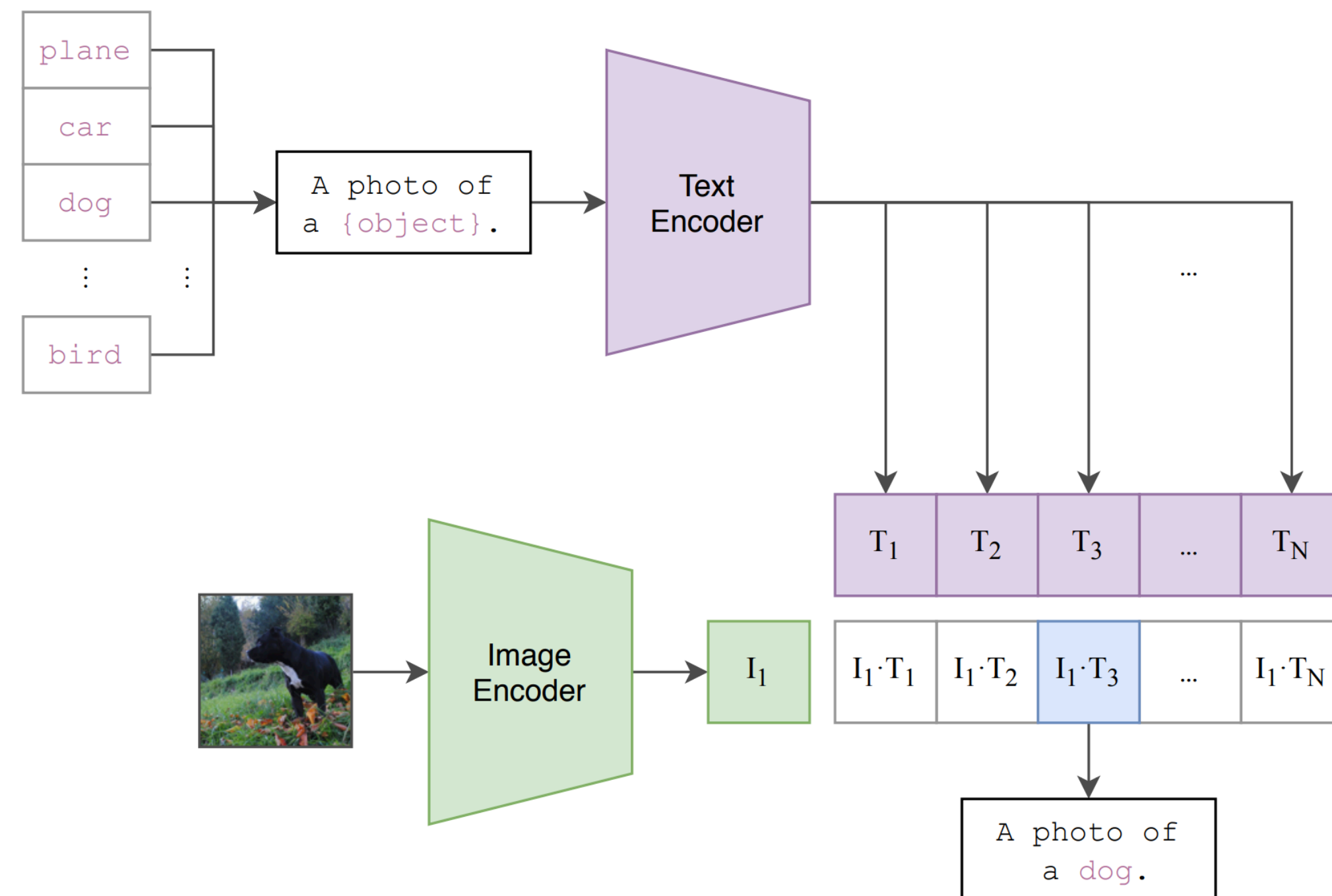


[Generated by FLUX.]

Challenges

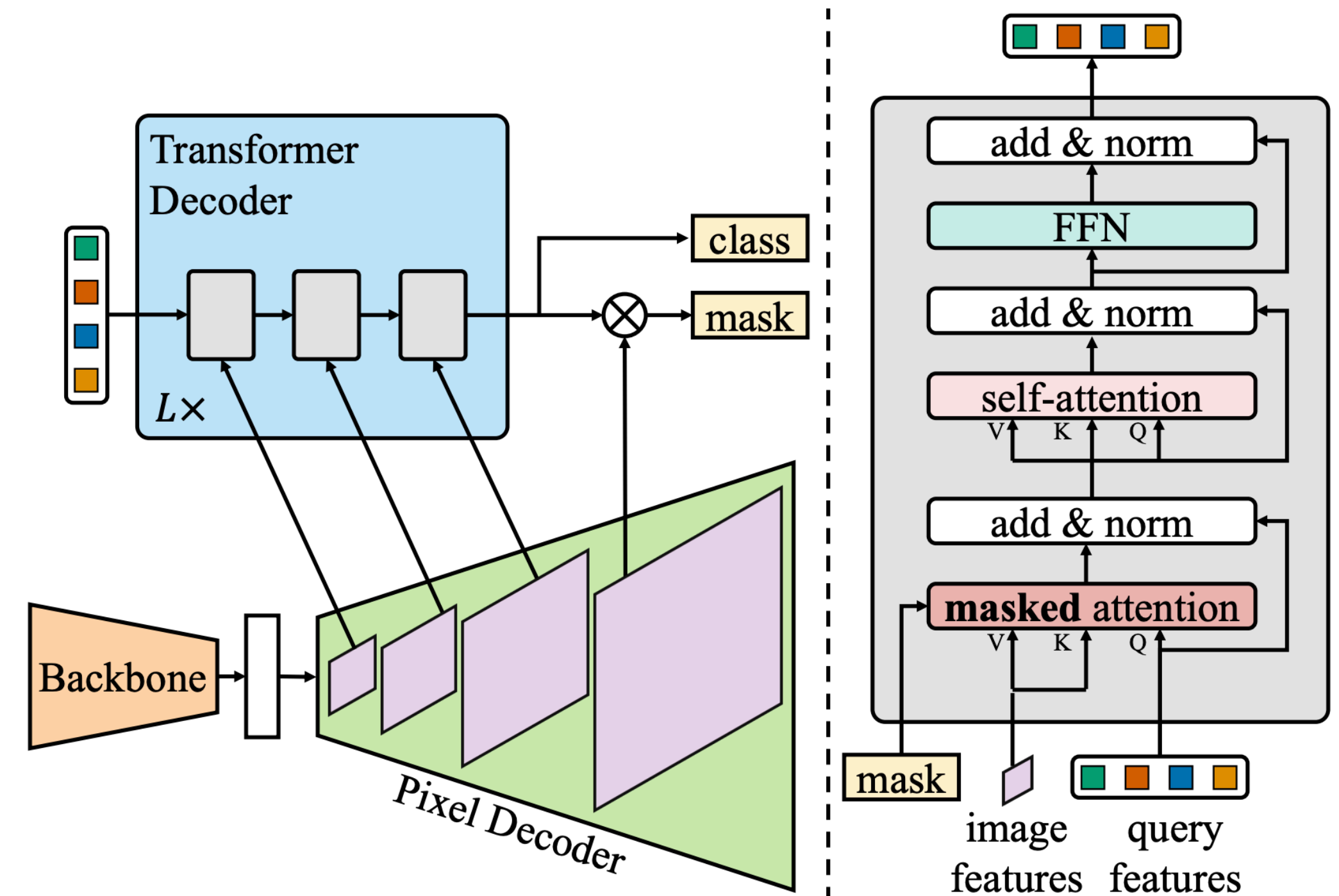
1. Some models have a relatively **fixed inference budget**. Eg.

a. Vision/multimodal encoders



Inference pipeline for CLIP [Radford et al. 2021]

b. Segmentation/detection models

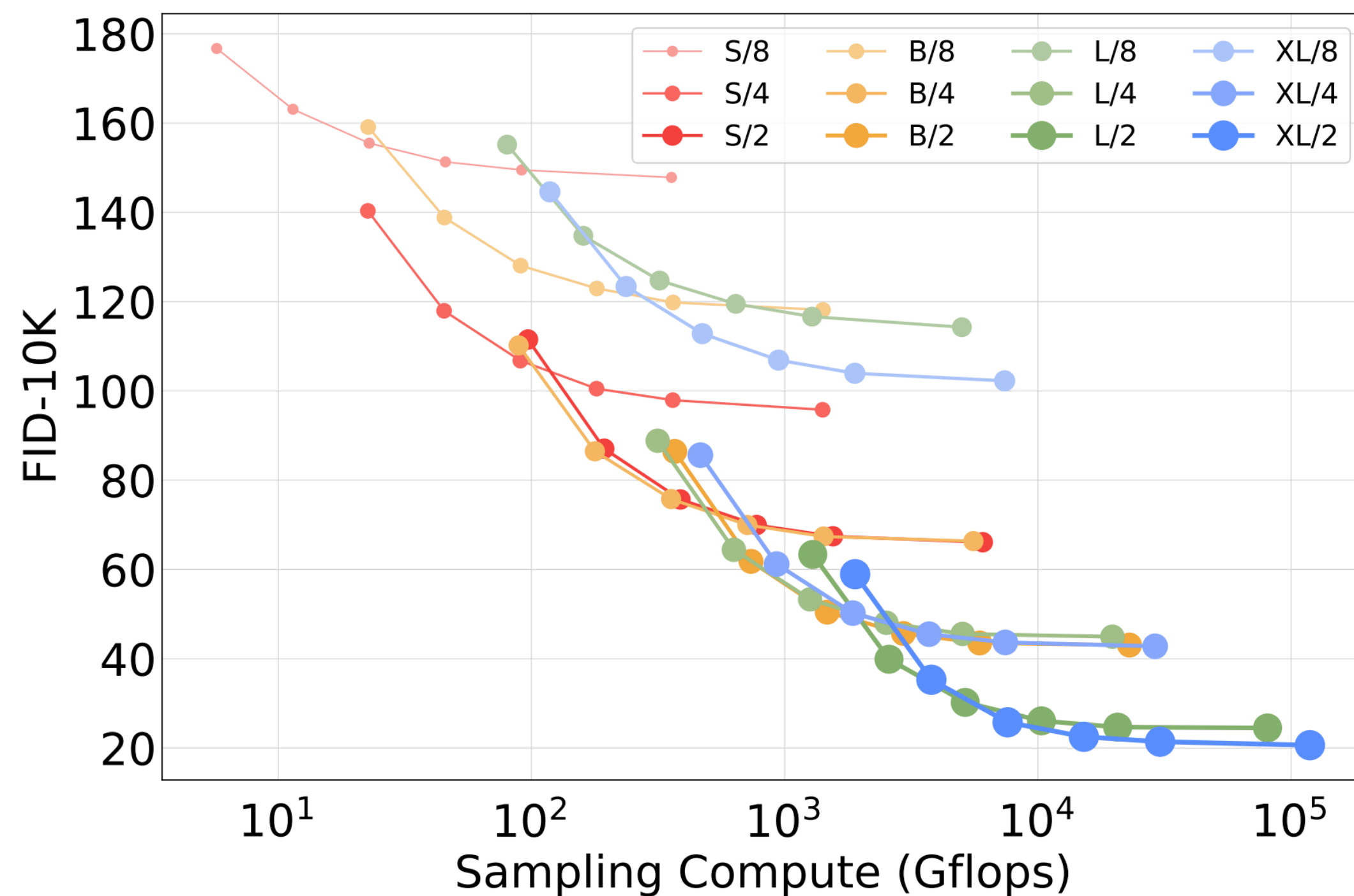


Overview of Mask2Former [Cheng et al. 2021]

Challenges

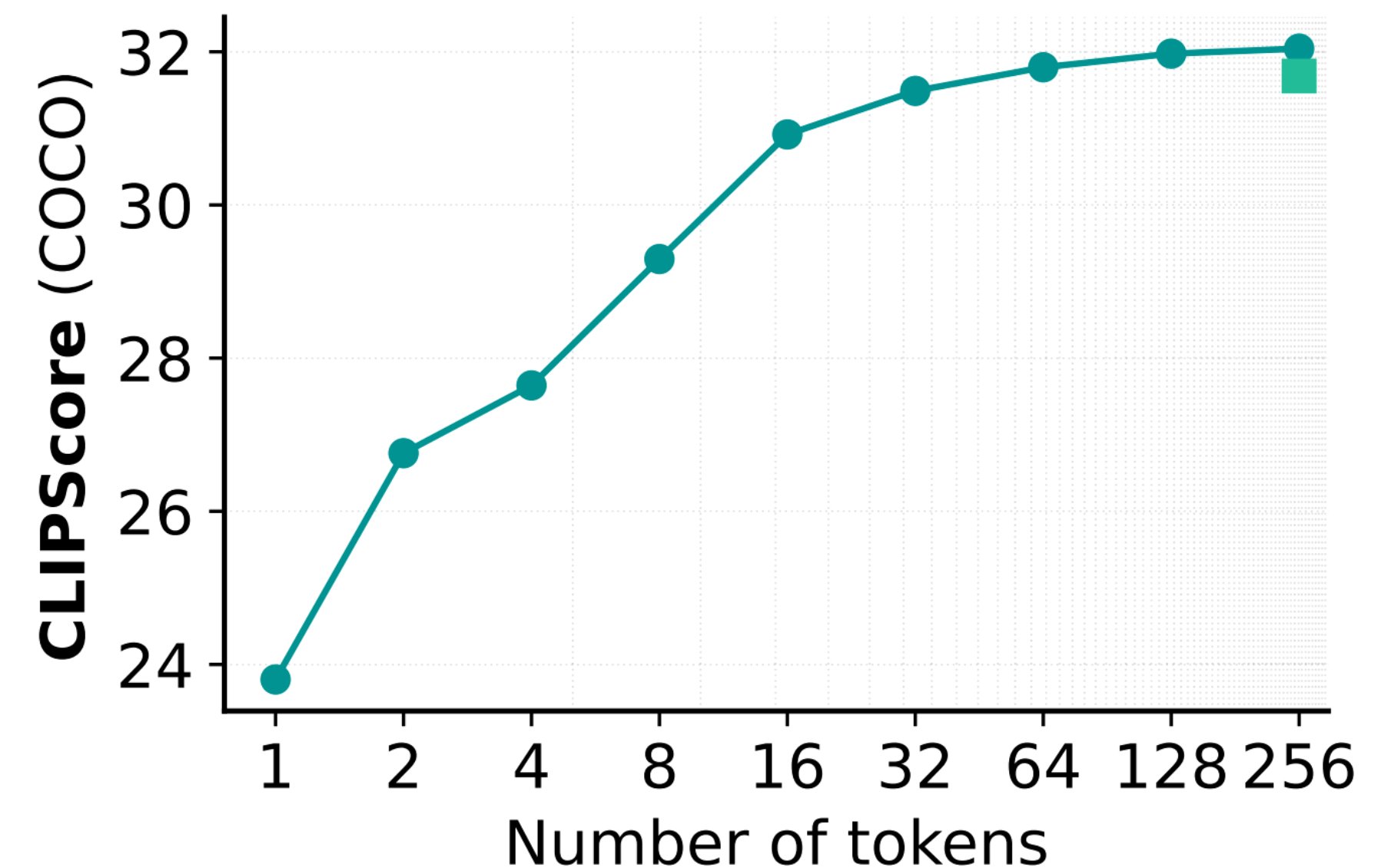
2. Some models support flexible inference budgets, but their performance gains quickly **saturate** with additional compute. Eg.

(a) Scaling denoising steps for diffusion models.



[Peebles et al. Scalable Diffusion Models with Transformers, 2022]

(b) Scaling token numbers for an autoregressive image generation model.



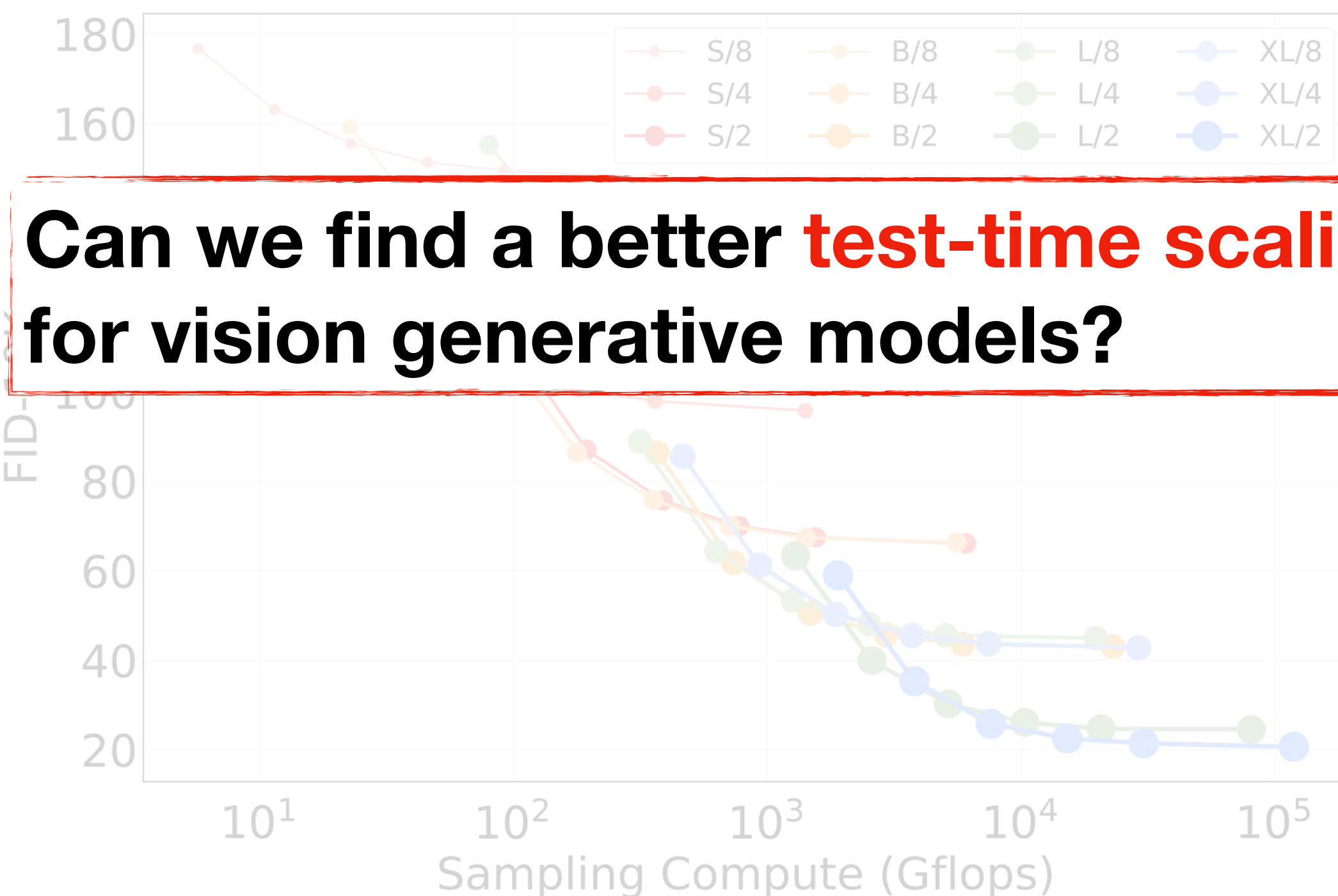
[Bachmann et al. FlexTok: Resampling Images into 1D Token Sequences of Flexible Length, 2022]

Key Question

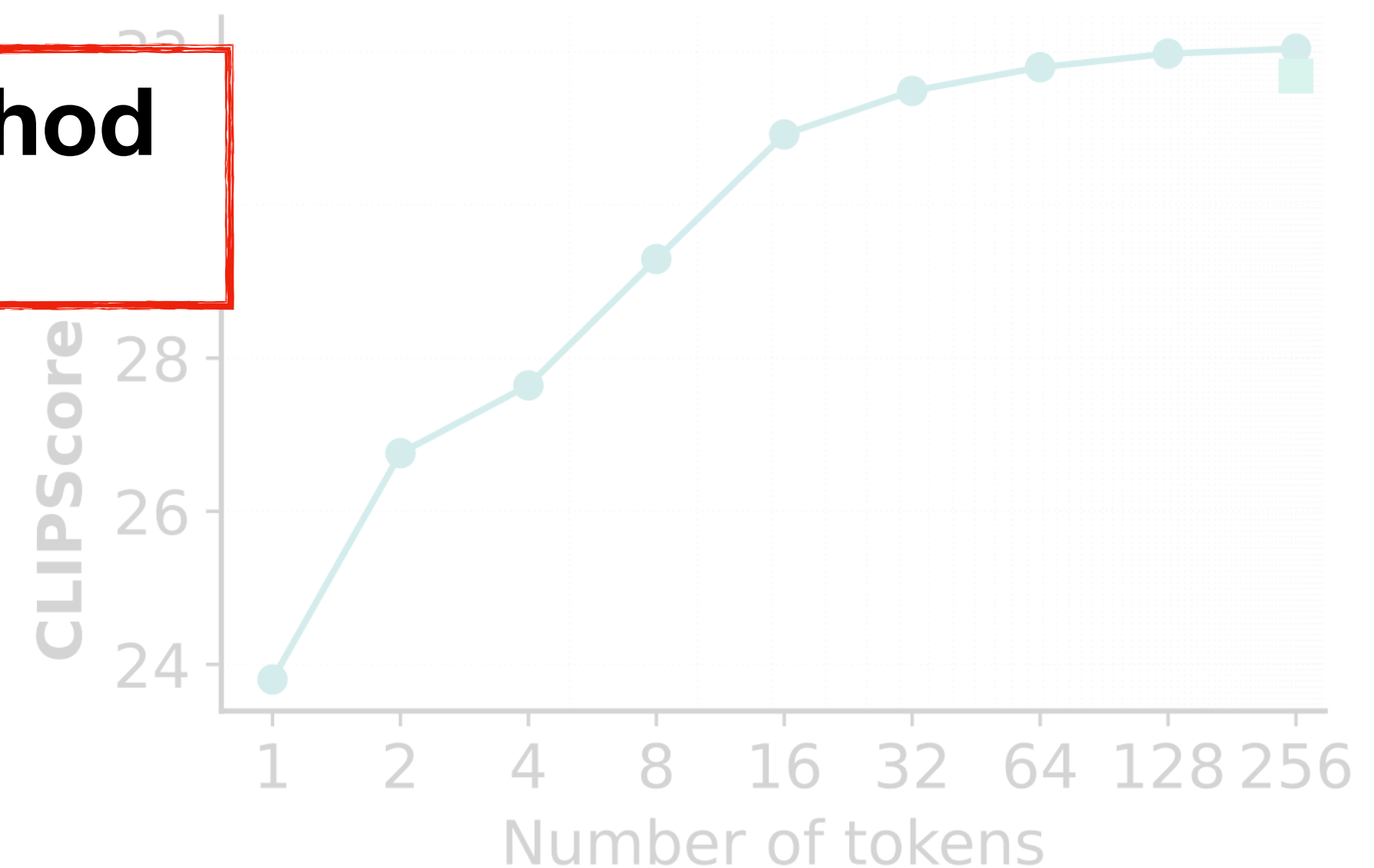
2. Some models support flexible inference budgets, but their performance gains quickly **saturate** with additional compute. Eg.

(a) Scaling denoising steps for diffusion models.

(b) Scaling token numbers for an autoregressive image generation model.



Can we find a better **test-time scaling** method for vision generative models?



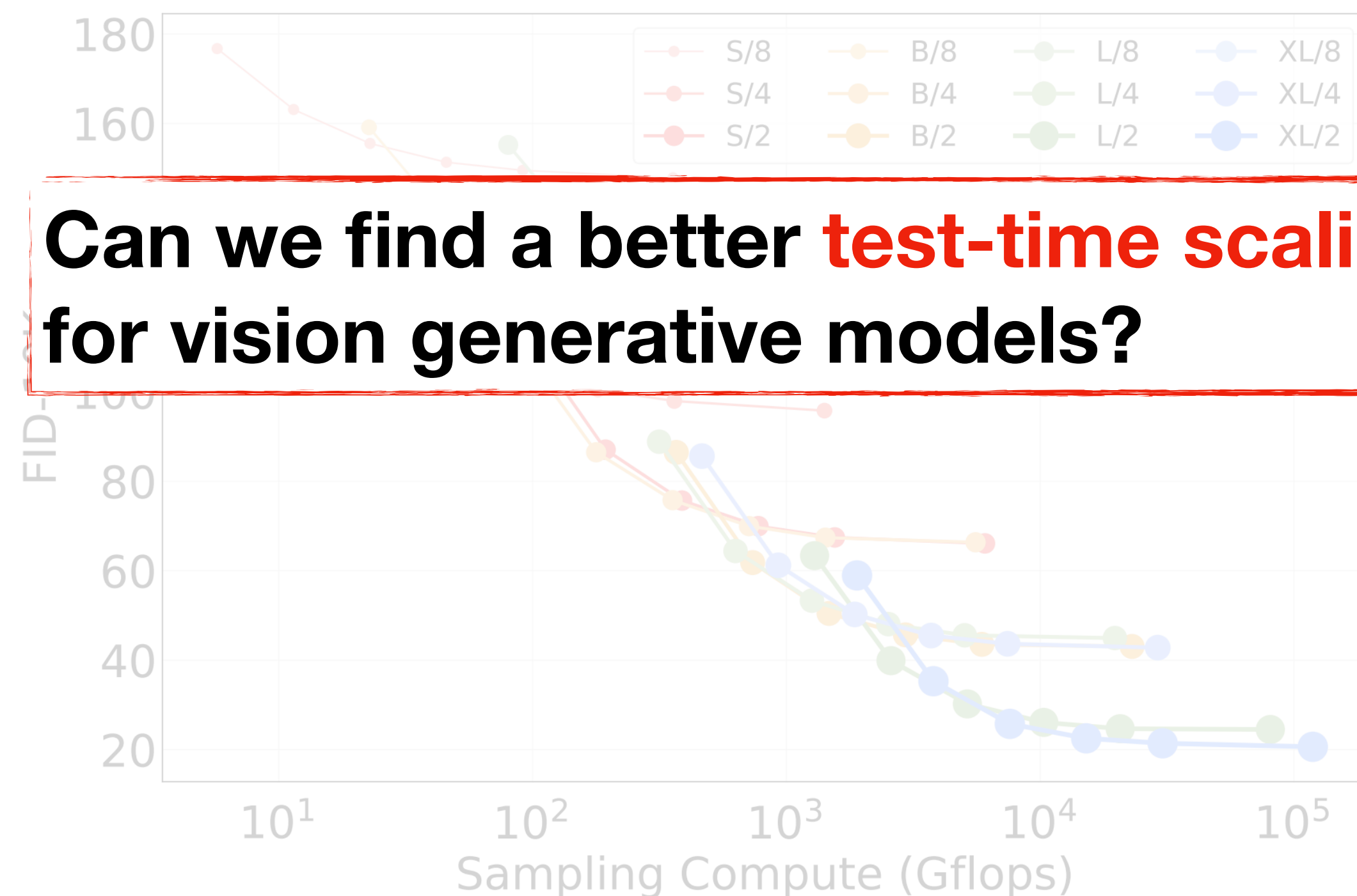
[Peebles et al. Scalable Diffusion Models with Transformers, 2022]

[Bachmann et al. FlexTok: Resampling Images into 1D Token Sequences of Flexible Length, 2022]

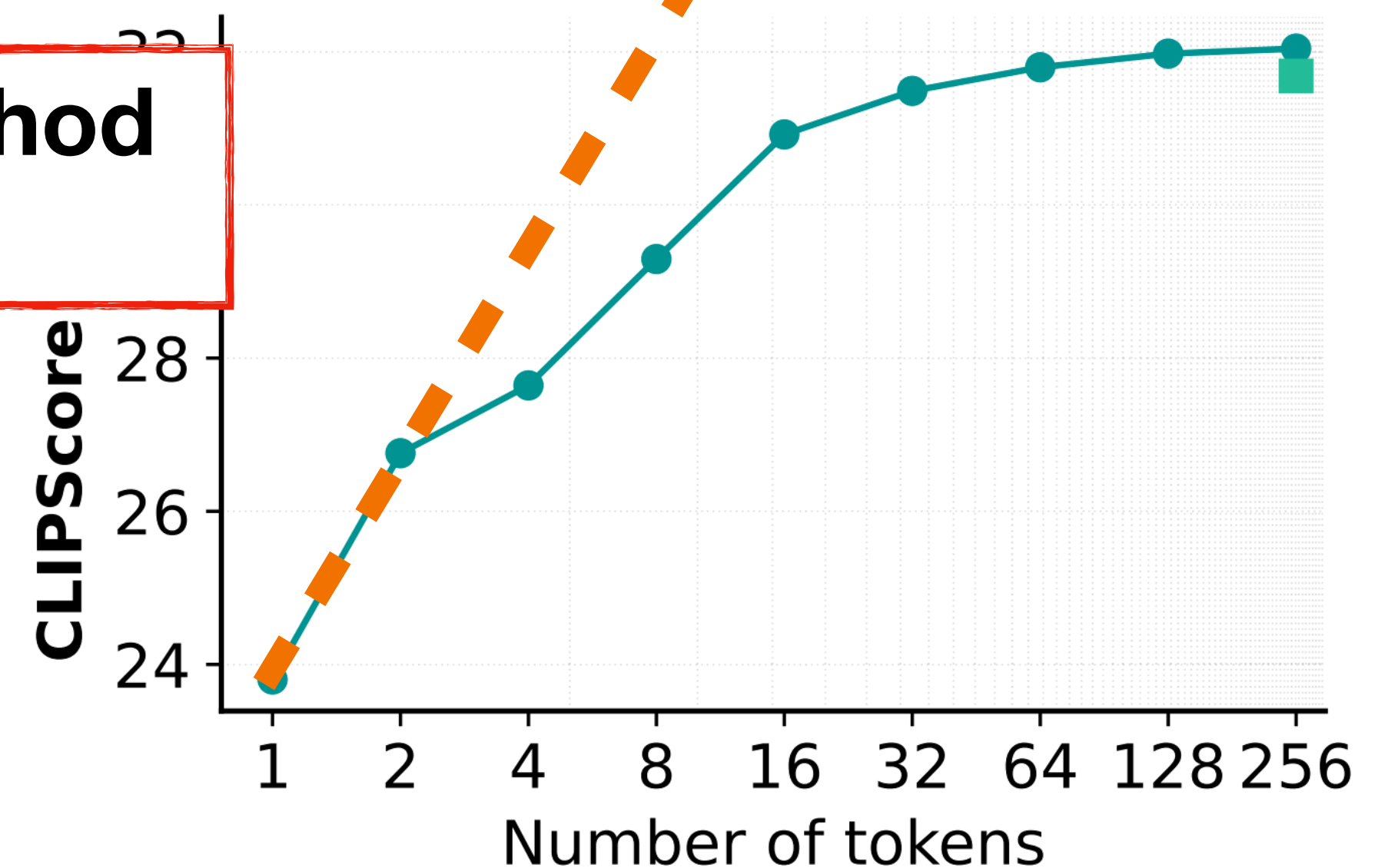
Key Question

2. Some models support flexible inference budgets, but their performance gains quickly **saturate** with additional compute. Eg.

(a) Scaling denoising steps for diffusion models.



(b) Scaling token numbers for an autoregressive image generation model.



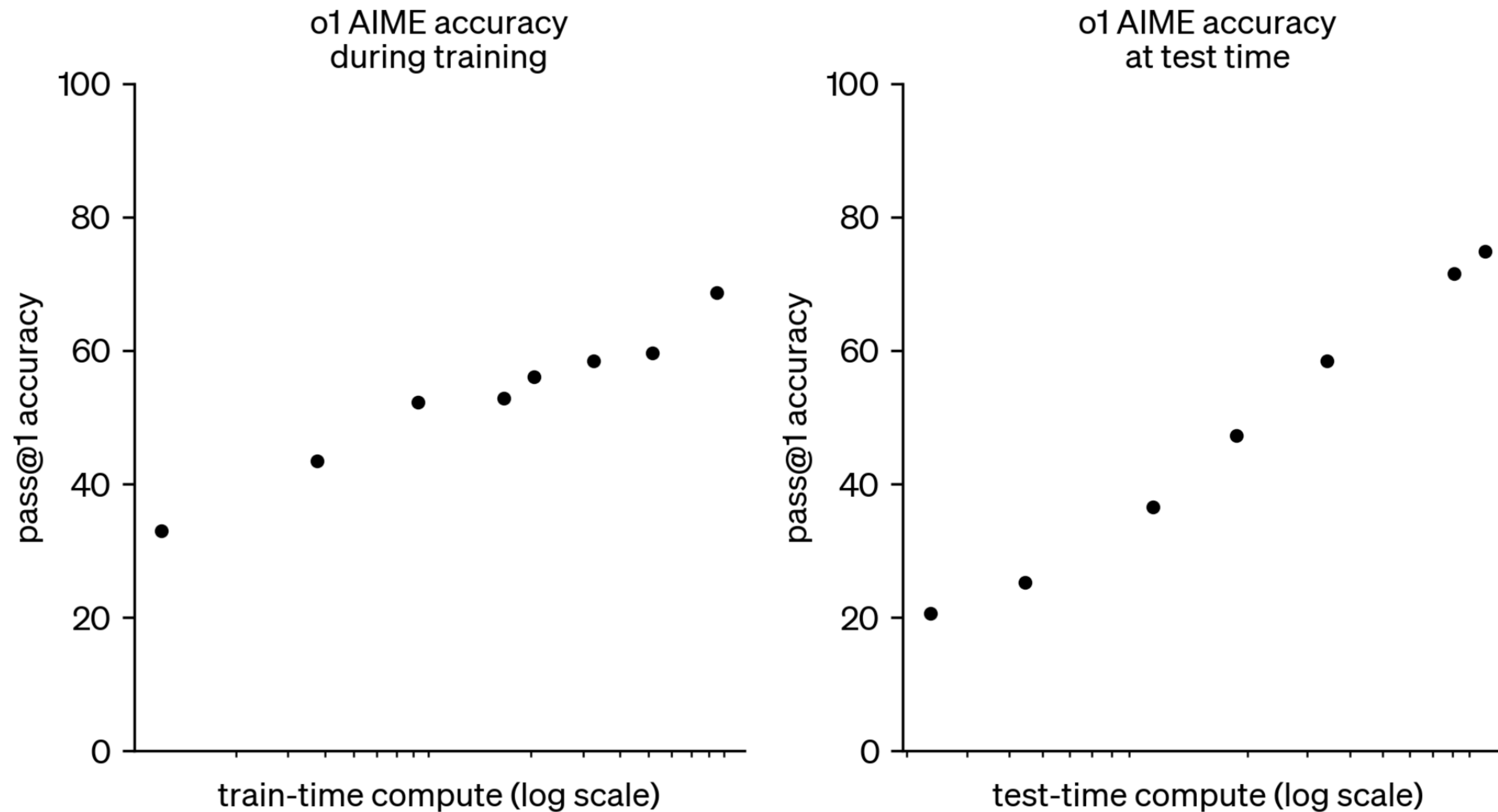
[Bachmann et al. FlexTok: Resampling Images into 1D Token Sequences of Flexible Length, 2022]

[Peebles et al. Scalable Diffusion Models with Transformers, 2022]

Background Successful cases in other domains

Background Successful cases in other domains

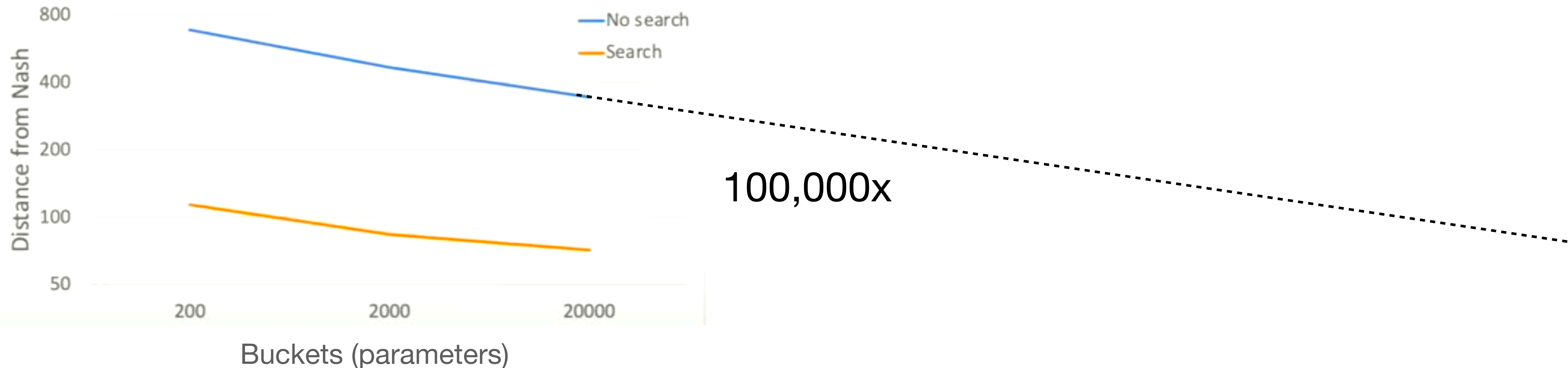
1. **Reasoning LLMs:** Increasing test-time compute leads to continual improvements in accuracy on complex reasoning tasks.



Background Successful cases in other domains

2. Board Game: Test-time compute is essential for success.

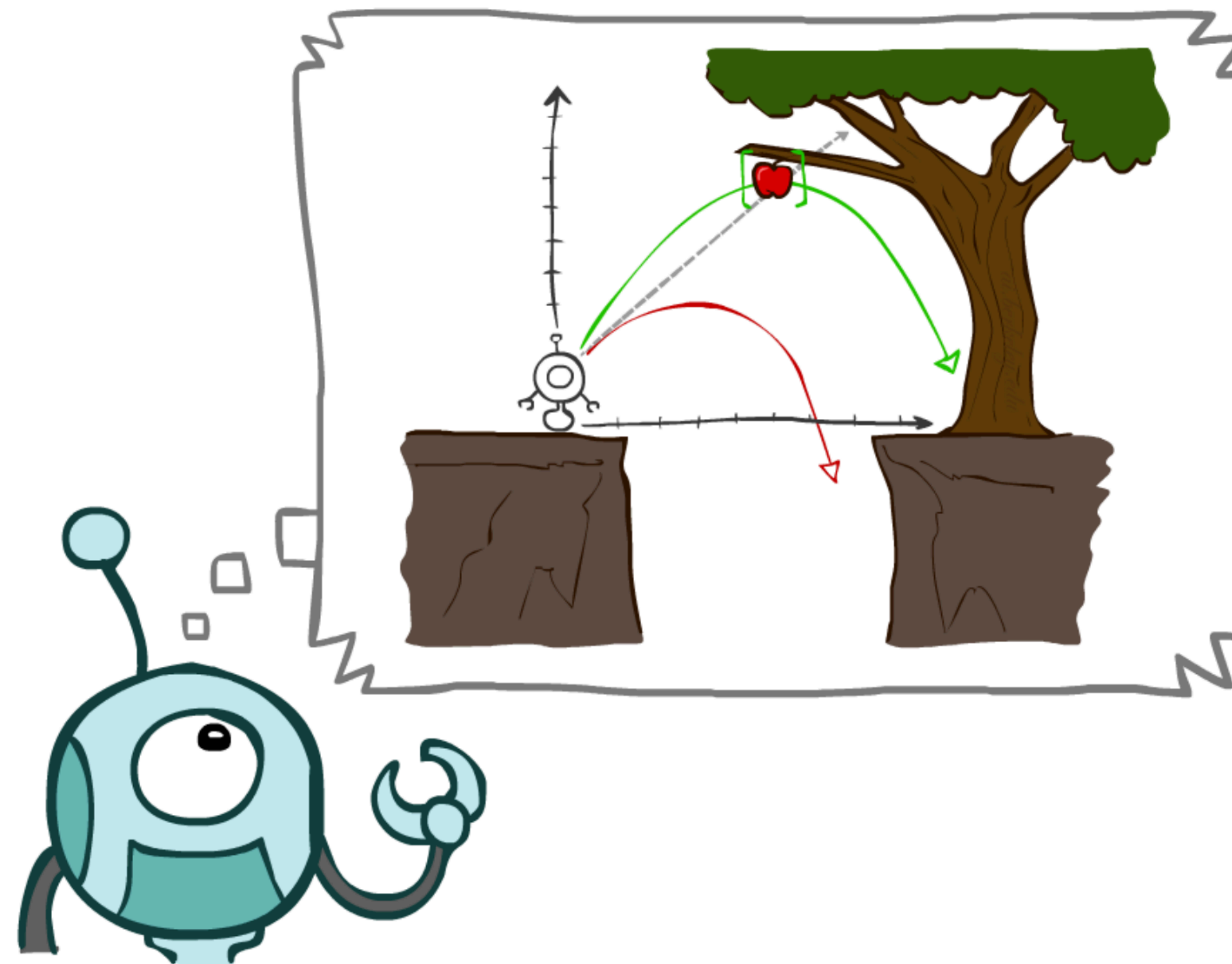
Poker game [2017]



[Brown et al. Safe and Nested Subgame Solving for Imperfect-Information Games. 2017]

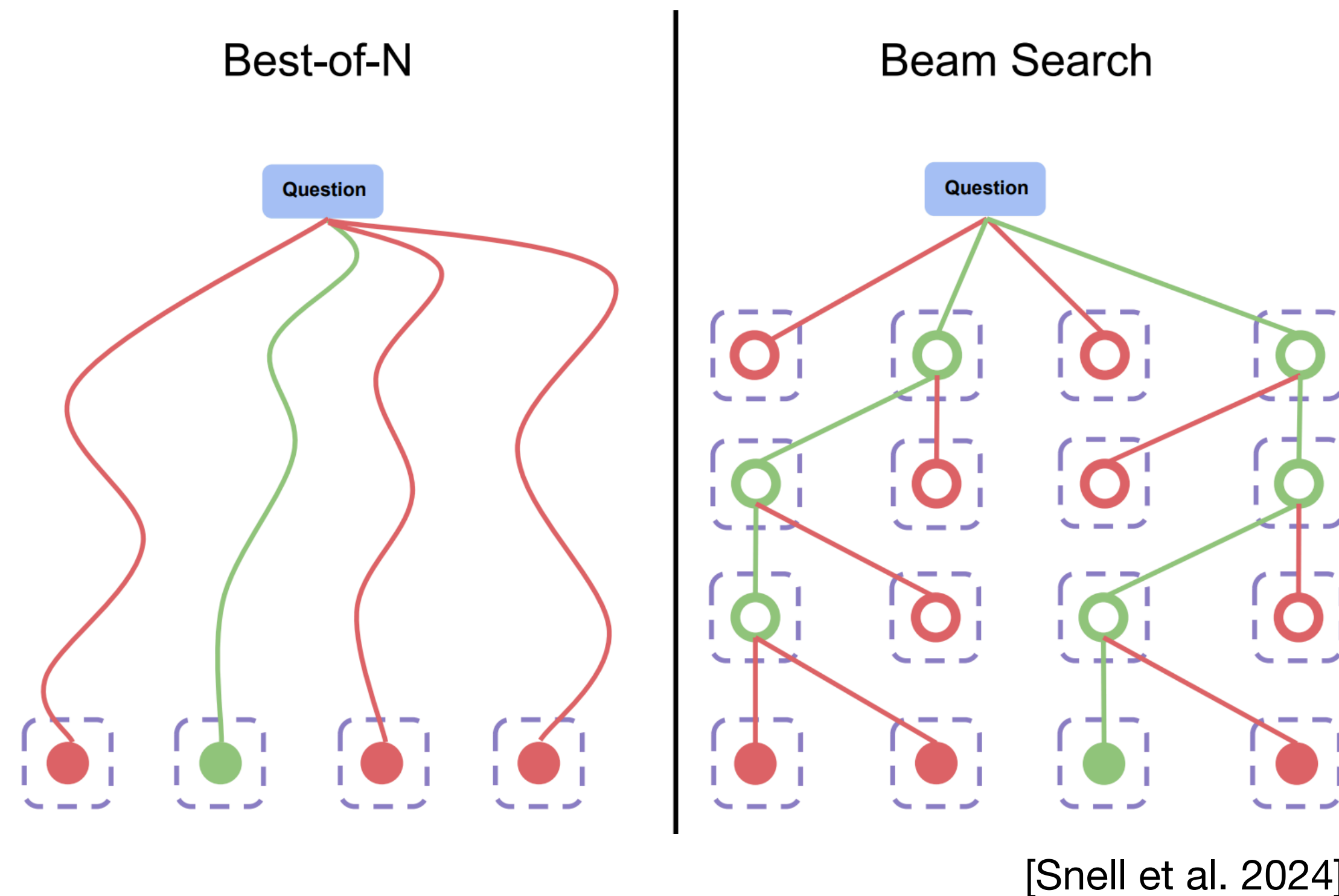
The Idea behind: Search

"Search" means exploring actions or solutions before making a final decision.



Test-time Scaling via Search

"**Search**" means exploring actions or solutions before making a final decision.



(a) Explicit Search (eg. Best of N, Beam Search, MCTS)

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?

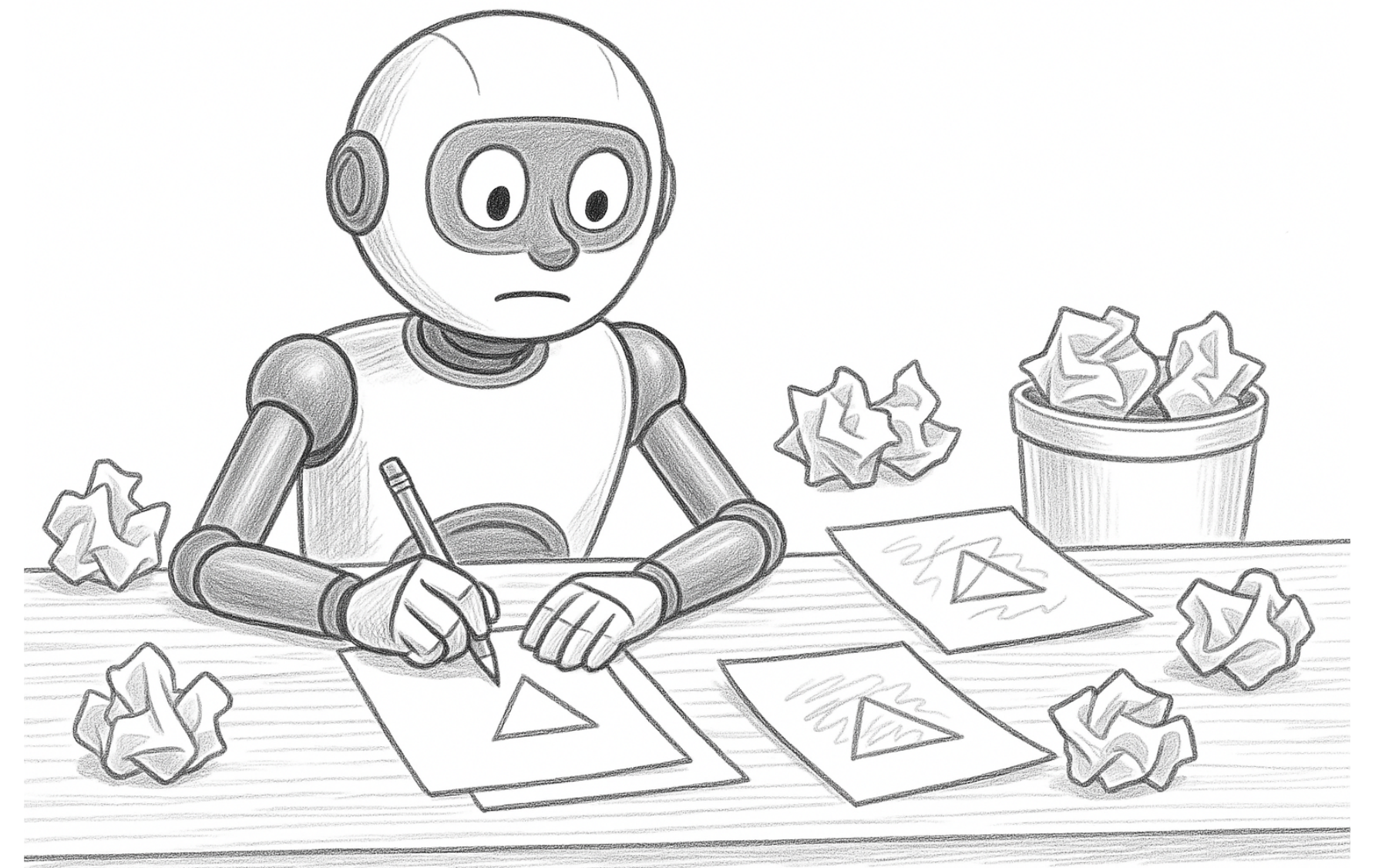
[OpenAI o1, 2024]

(b) Implicit Search (eg, CoT)

Main Idea: Visual Test-time Scaling via Search

How about Search?

Can we find a better test-time scaling method for vision generative models?



[Generated by ChatGPT 4o.]

Overview

Provide background for test-time search techniques.

What has been done for the task that we considered?

1st paper

—
“Mastering the game of Go with deep neural networks and tree search” (AlphaGo)
—

Search in Game Play

2nd paper

—
“DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning”
—

Search in Language Models

3rd paper

—
“Inference-Time Scaling for Diffusion Models beyond Scaling Denoising Steps”
—

Search in Vision

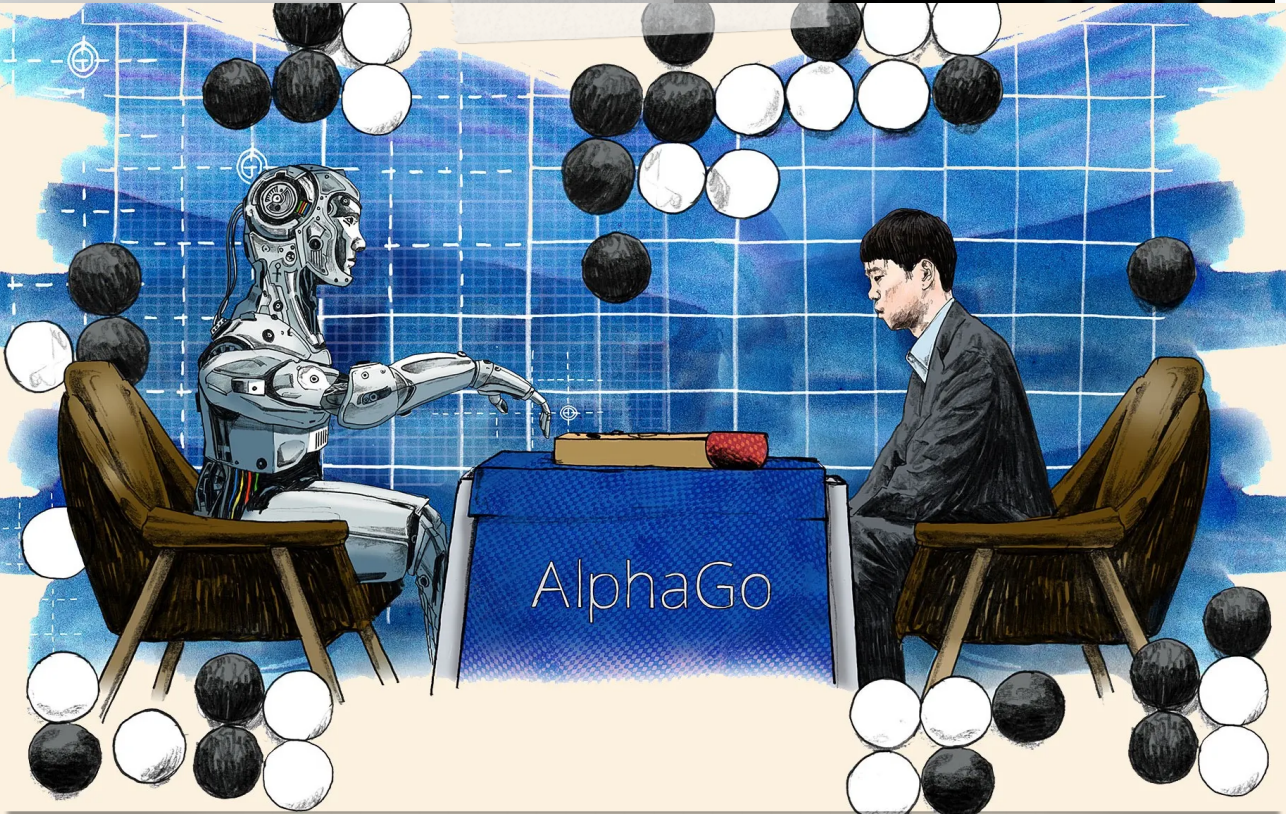
Discussion & Research Proposal

—
Ongoing explorations and future work

1st paper

Mastering the game of Go with *deep neural networks* and tree search

Google DeepMind. 2016



ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver^{1*}, Aja Huang^{1*}, Chris J. Maddison¹, Arthur Guez¹, Laurent Sifre¹, George van den Driessche¹, Julian Schrittwieser¹, Ioannis Antonoglou¹, Veda Panneershelvam¹, Marc Lanctot¹, Sander Dieleman¹, Dominik Grewe¹, John Nham², Nal Kalchbrenner¹, Ilya Sutskever², Timothy Lillicrap¹, Madeleine Leach¹, Koray Kavukcuoglu¹, Thore Graepel¹ & Demis Hassabis¹

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses ‘value networks’ to evaluate board positions and ‘policy networks’ to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

All games of perfect information have an optimal value function, $v^*(s)$, which determines the outcome of the game, from every board position or state s , under perfect play by all players. These games may be solved by recursively computing the optimal value function in a search tree containing approximately b^d possible sequences of moves, where b is the game’s breadth (number of legal moves per position) and d is its depth (game length). In large games, such as chess ($b \approx 35$, $d \approx 80$)¹ and especially Go ($b \approx 250$, $d \approx 150$)¹, exhaustive search is infeasible^{2,3}, but the effective search space can be reduced by two general principles. First, the depth of the search may be reduced by position evaluation: truncating the search tree at state s and replacing the subtree below s by an approximate value function $v(s) \approx v^*(s)$ that predicts the outcome from state s . This approach has led to superhuman performance in chess⁴, checkers⁵ and othello⁶, but it was believed to be intractable in Go due to the complexity of the game⁷. Second, the breadth of the search may be reduced by sampling actions from a policy $p(a|s)$ that is a probability distribution over possible moves a in position s . For example, Monte Carlo rollouts⁸ search to maximum depth without branching at all, by sampling long sequences of actions for both players from a policy p . Averaging over such rollouts can provide an effective position evaluation, achieving superhuman performance in backgammon⁸ and Scrabble⁹, and weak amateur level play in Go¹⁰. Monte Carlo tree search (MCTS)^{11,12} uses Monte Carlo rollouts to estimate the value of each state in a search tree. As more simulations are executed, the search tree grows larger and the relevant values become more accurate. The policy used to select actions during search is also improved over time, by selecting children with higher values. Asymptotically, this policy converges to optimal play, and the evaluations converge to the optimal value function¹². The strongest current Go programs are based on MCTS, enhanced by policies that are trained to predict human expert moves¹³. These policies are used to narrow the search to a beam of high-probability actions, and to sample actions during rollouts. This approach has achieved strong amateur play^{13–15}. However, prior work has been limited to shallow

policies^{13–15} or value functions¹⁶ based on a linear combination of input features. Recently, deep convolutional neural networks have achieved unprecedented performance in visual domains: for example, image classification¹⁷, face recognition¹⁸, and playing Atari games¹⁹. They use many layers of neurons, each arranged in overlapping tiles, to construct increasingly abstract, localized representations of an image²⁰. We employ a similar architecture for the game of Go. We pass in the board position as a 19×19 image and use convolutional layers to construct a representation of the position. We use these neural networks to reduce the effective depth and breadth of the search tree: evaluating positions using a value network, and sampling actions using a policy network. We train the neural networks using a pipeline consisting of several stages of machine learning (Fig. 1). We begin by training a supervised learning (SL) policy network p_s directly from expert human moves. This provides fast, efficient learning updates with immediate feedback and high-quality gradients. Similar to prior work^{13,15}, we also train a fast policy p_r that can rapidly sample actions during rollouts. Next, we train a reinforcement learning (RL) policy network p_l that improves the SL policy network by optimizing the final outcome of games of self-play. This adjusts the policy towards the correct goal of winning games, rather than maximizing predictive accuracy. Finally, we train a value network v_θ that predicts the winner of games played by the RL policy network against itself. Our program AlphaGo efficiently combines the policy and value networks with MCTS.

Supervised learning of policy networks

For the first stage of the training pipeline, we build on prior work on predicting expert moves in the game of Go using supervised learning^{13,21–23}. The SL policy network $p_s(a|s)$ alternates between convolutional layers with weights σ , and rectifier nonlinearities. A final softmax layer outputs a probability distribution over all legal moves a . The input s to the policy network is a simple representation of the board state (see Extended Data Table 2). The policy network is trained on randomly

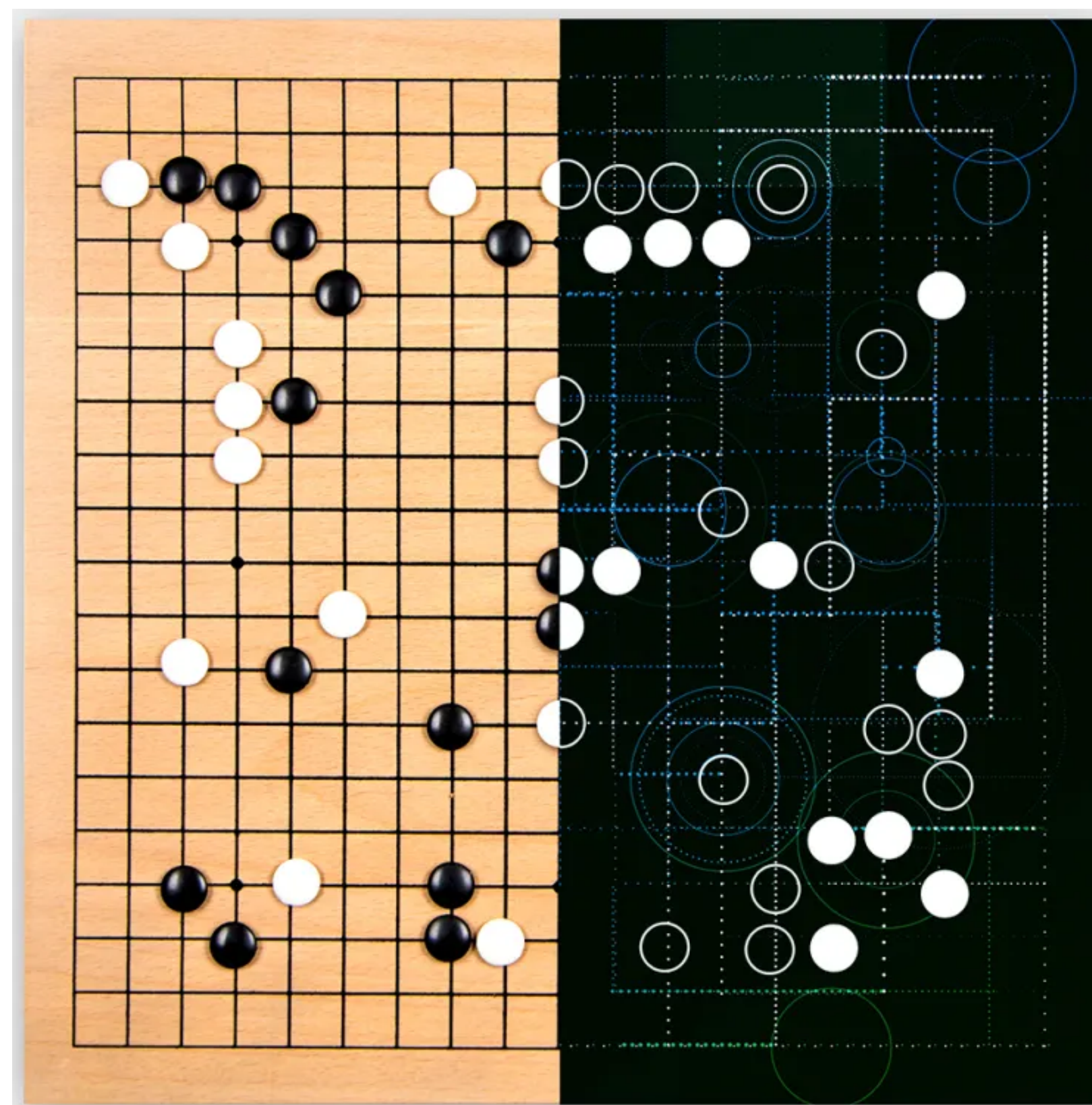
¹Google DeepMind, 5 New Street Square, London EC4A 3TW, UK. ²Google, 1600 Amphitheatre Parkway, Mountain View, California 94043, USA. *These authors contributed equally to this work.

Problem: GO as a ...

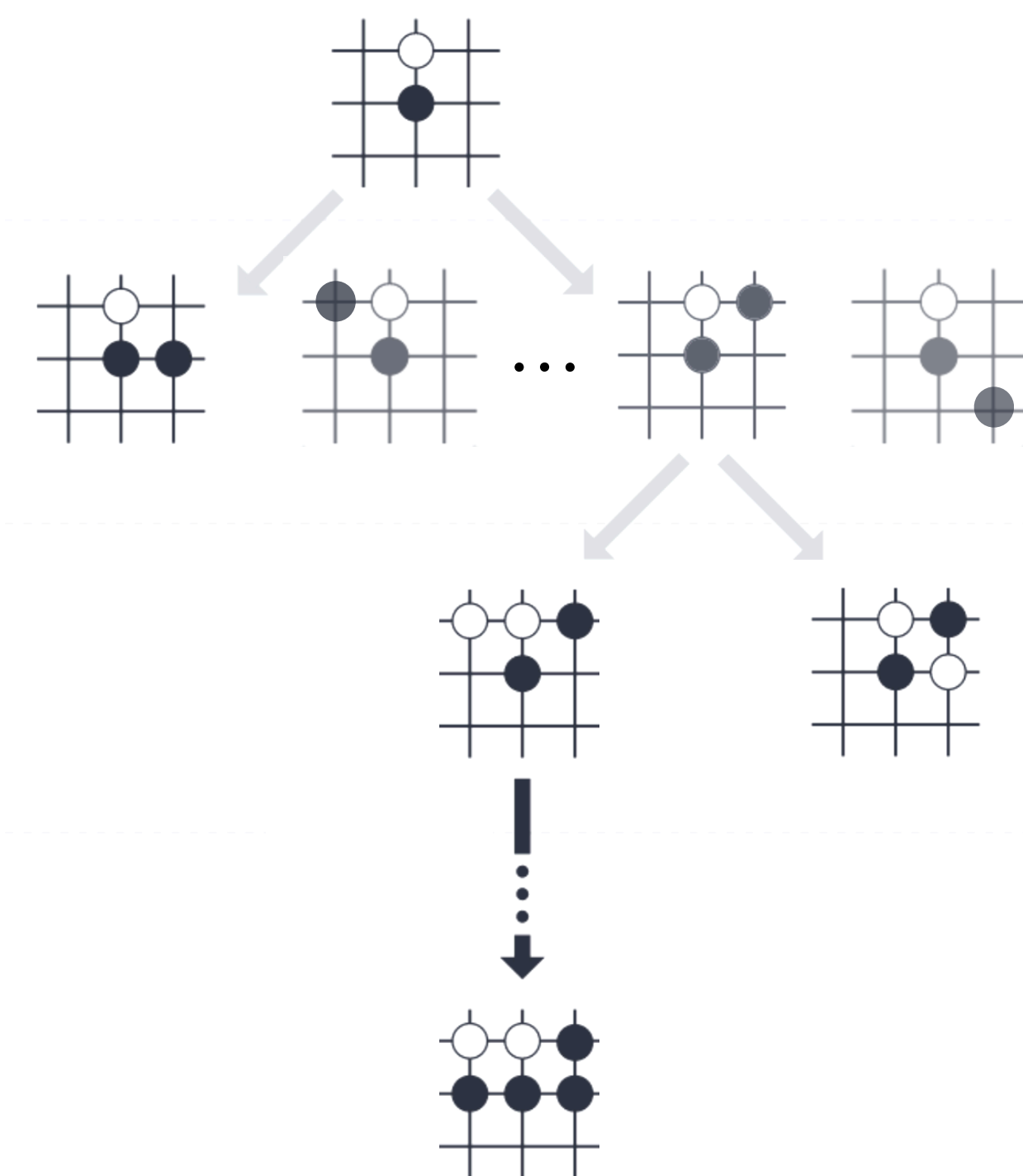
Go is a two-player strategic game.

The goal is to fence off more territory.

Board Game

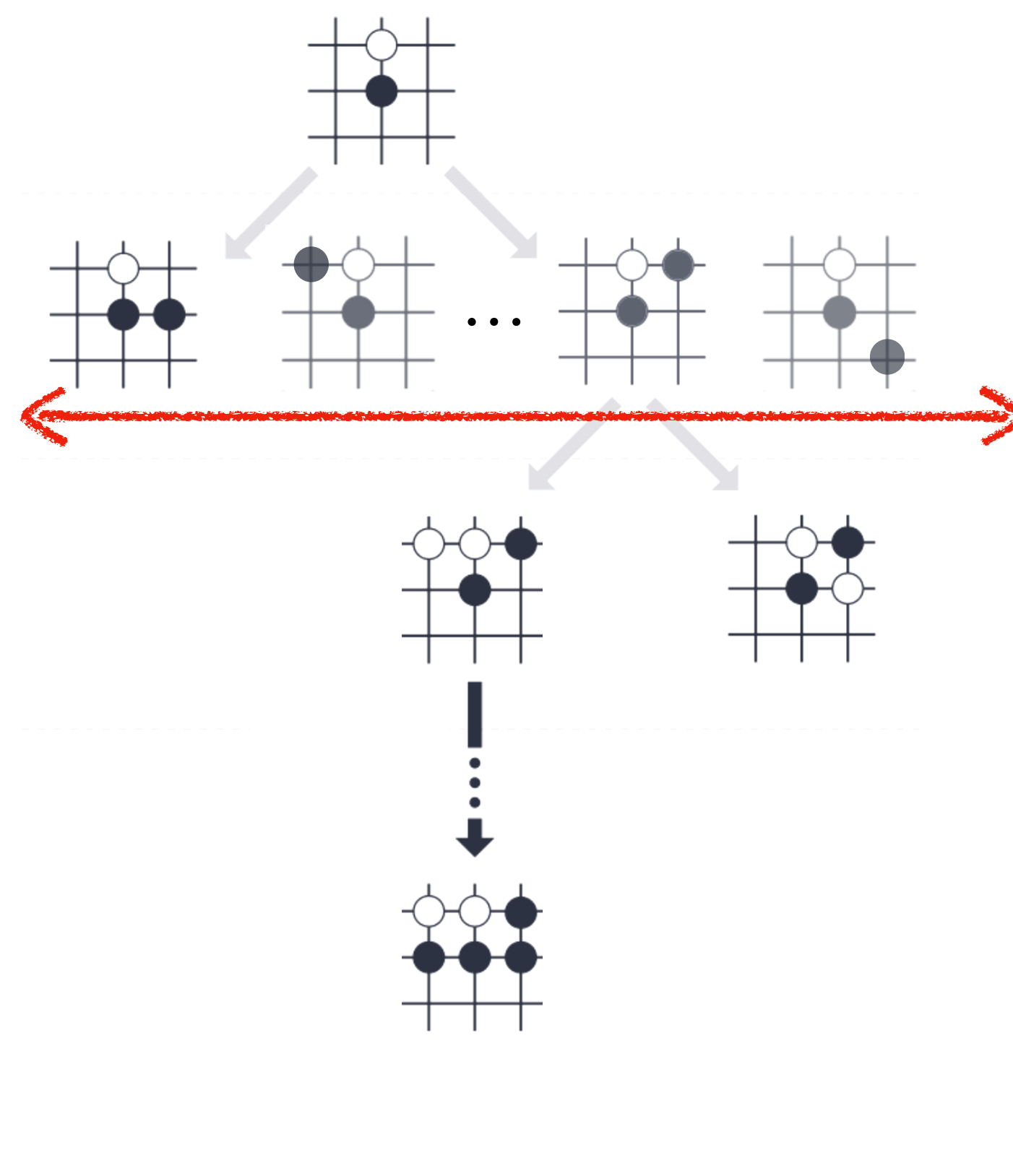
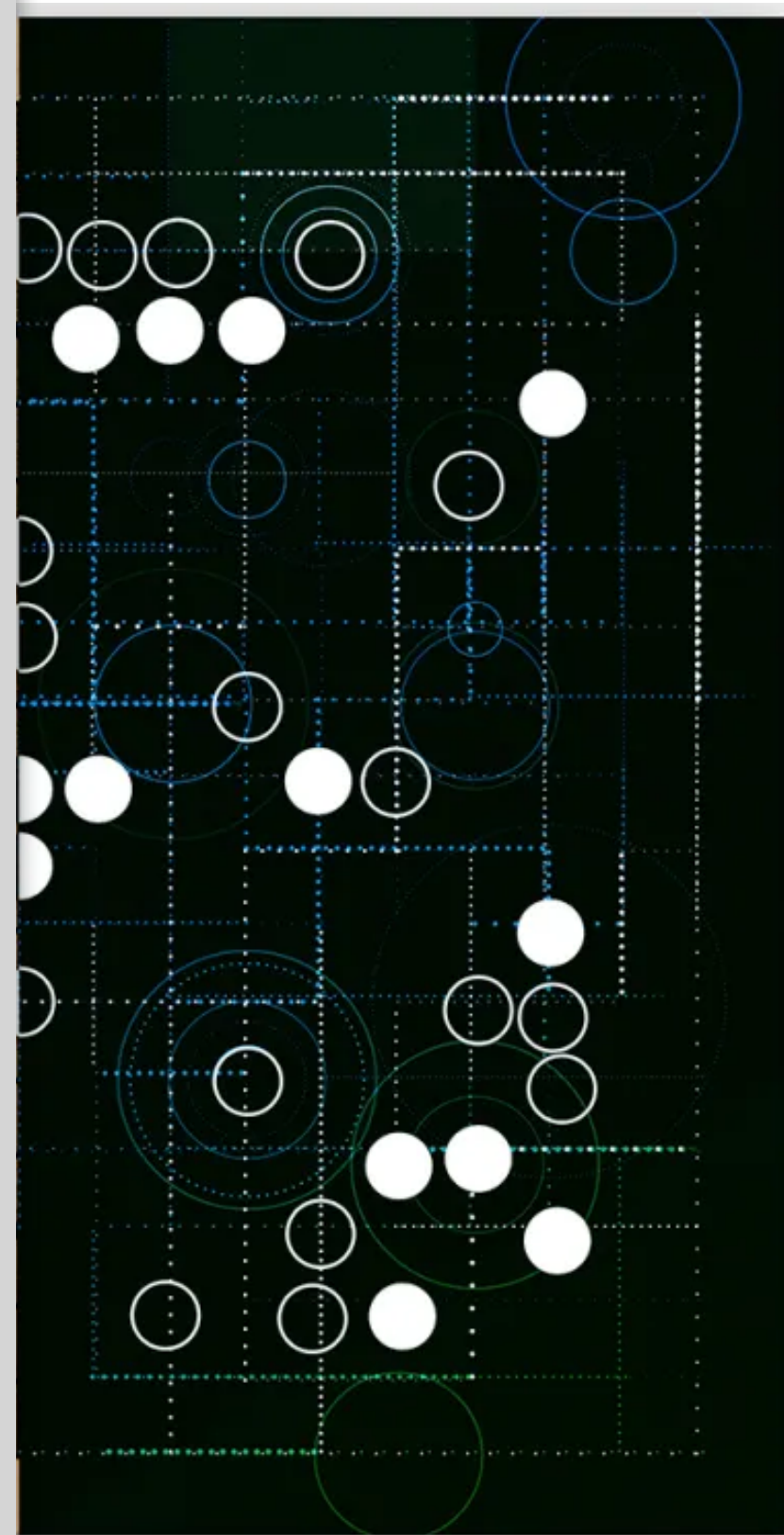


Search Problem



Problem: GO as a ...

Search Problem



Challenge: Search Space is Huge!

Search Breadth: $b \approx 250$

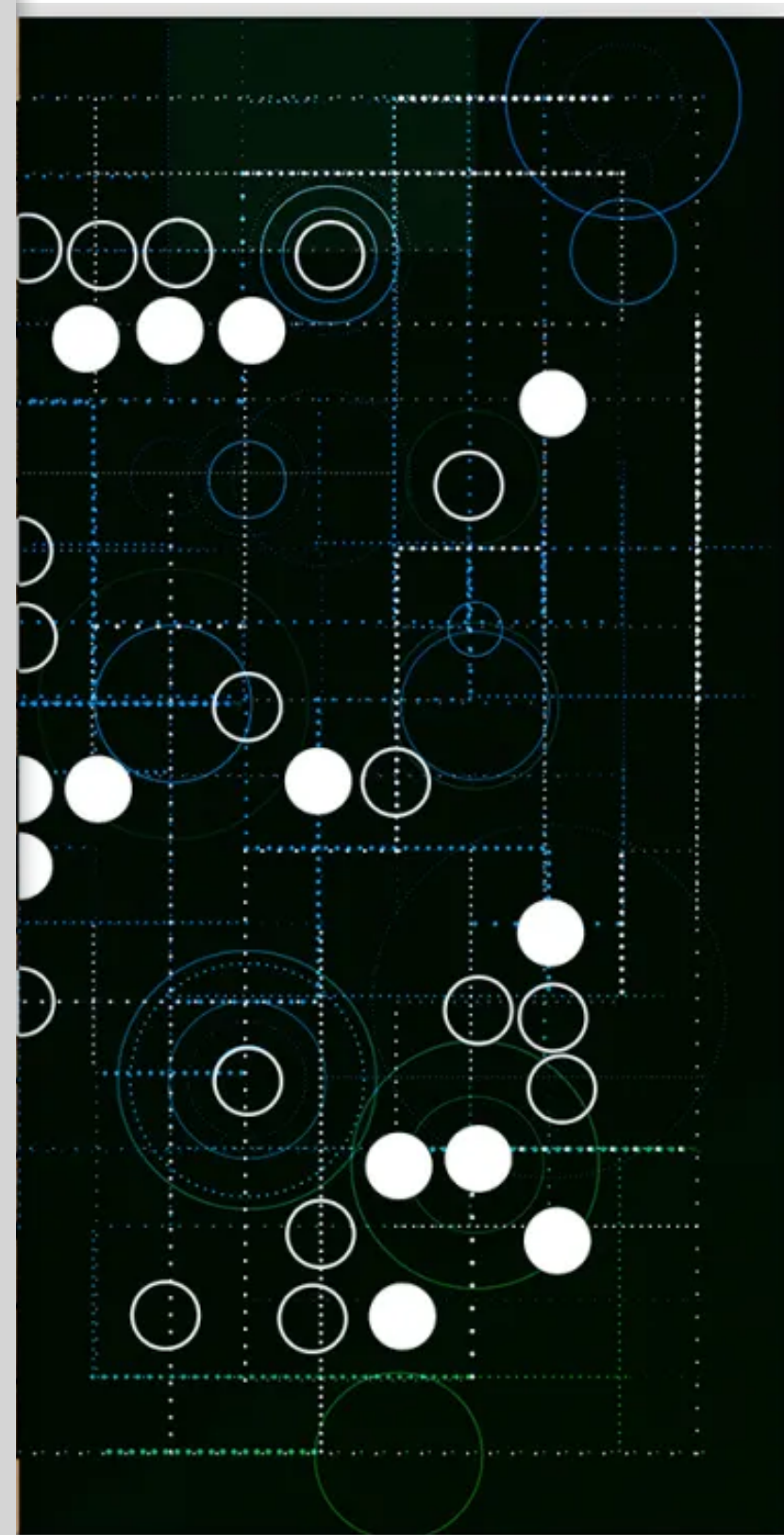
Search Length: $d \approx 150$

Search Space $\approx b^d$

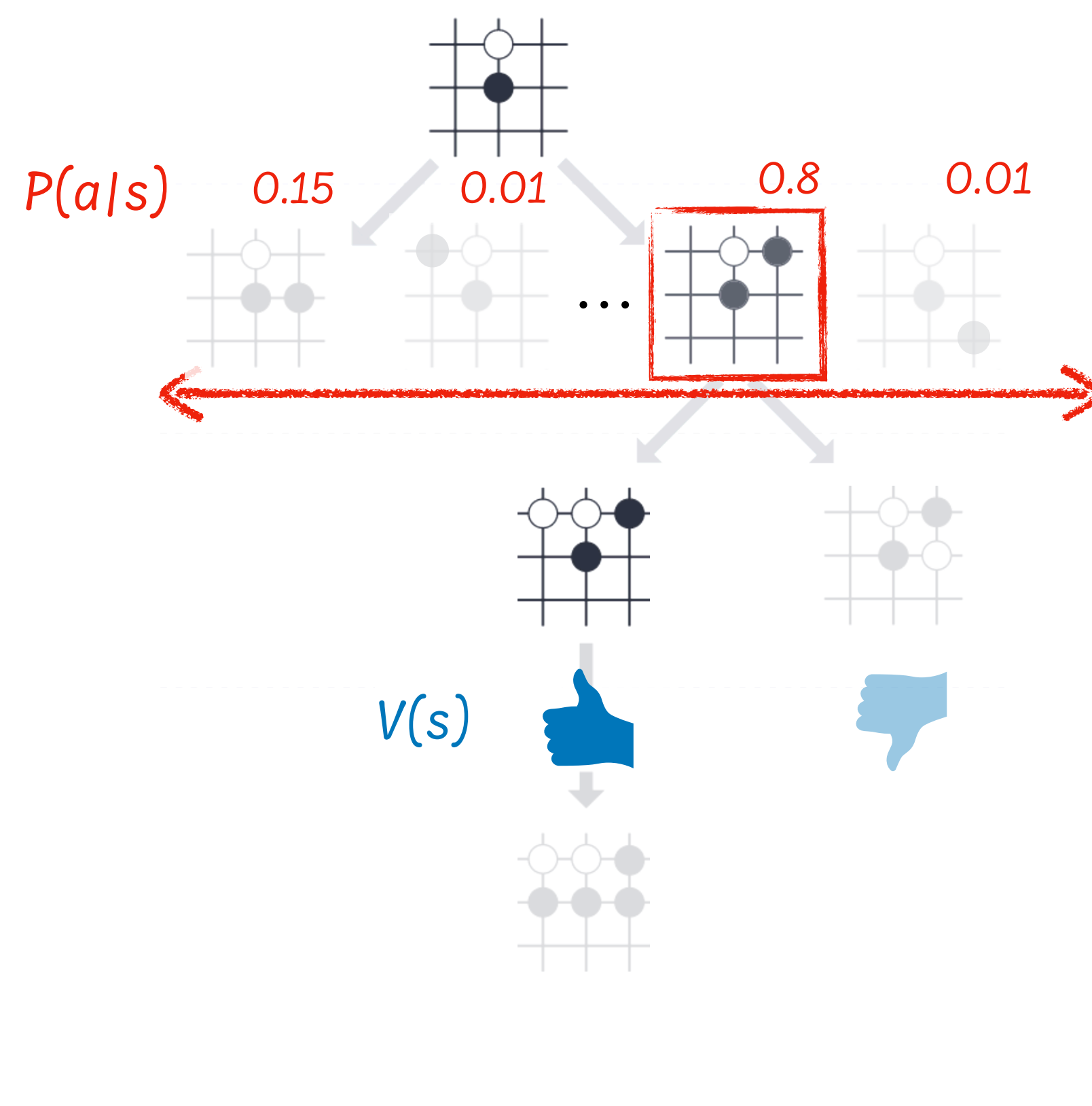
$$250^{150} \approx 4.91 \times 10^{359}$$

Problem: GO as a ...

Search Problem



Challenge: Search Space is Huge!



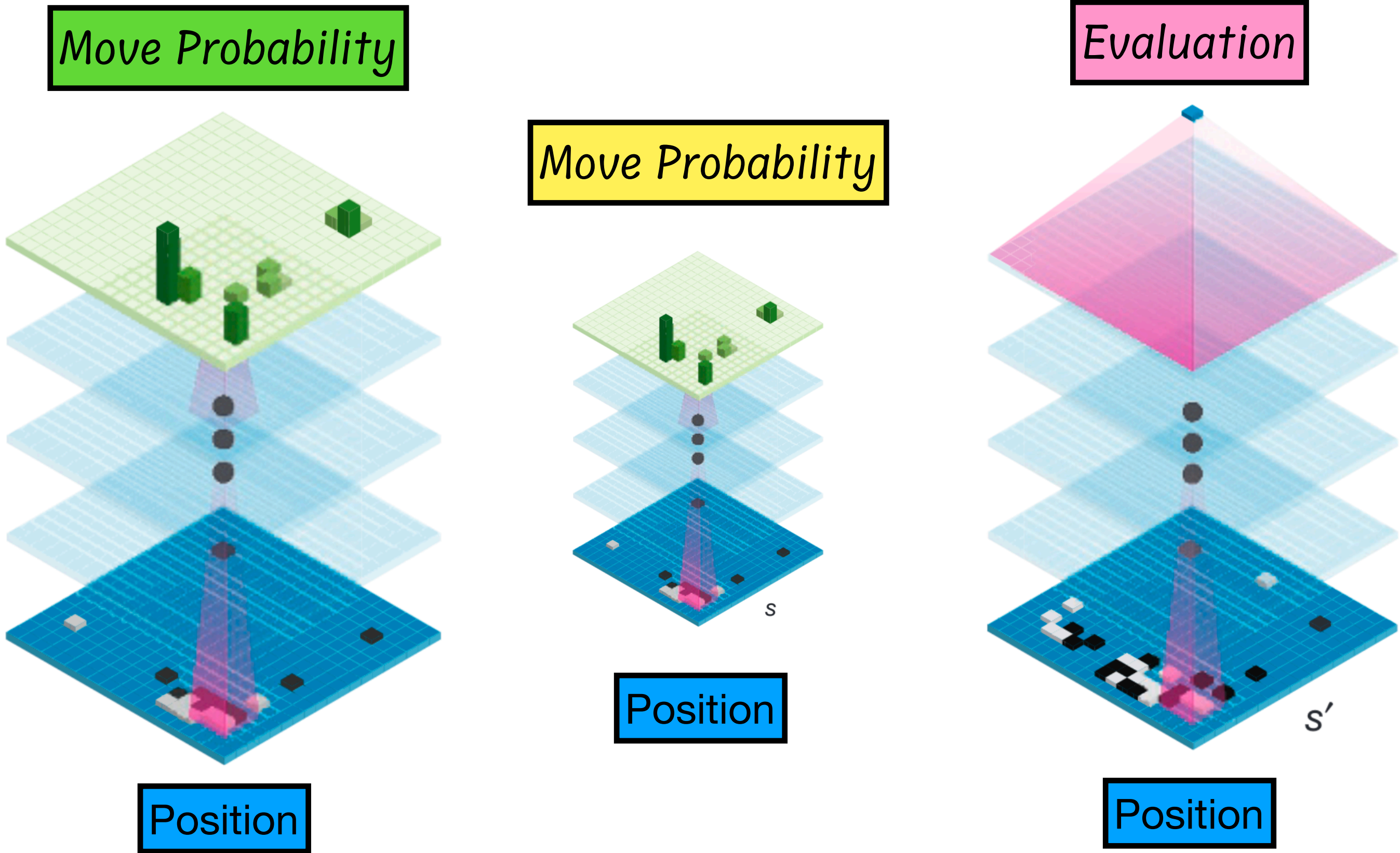
Reduce Breath with a Policy Network

Reduce Depth with a Value Network

Method Overview

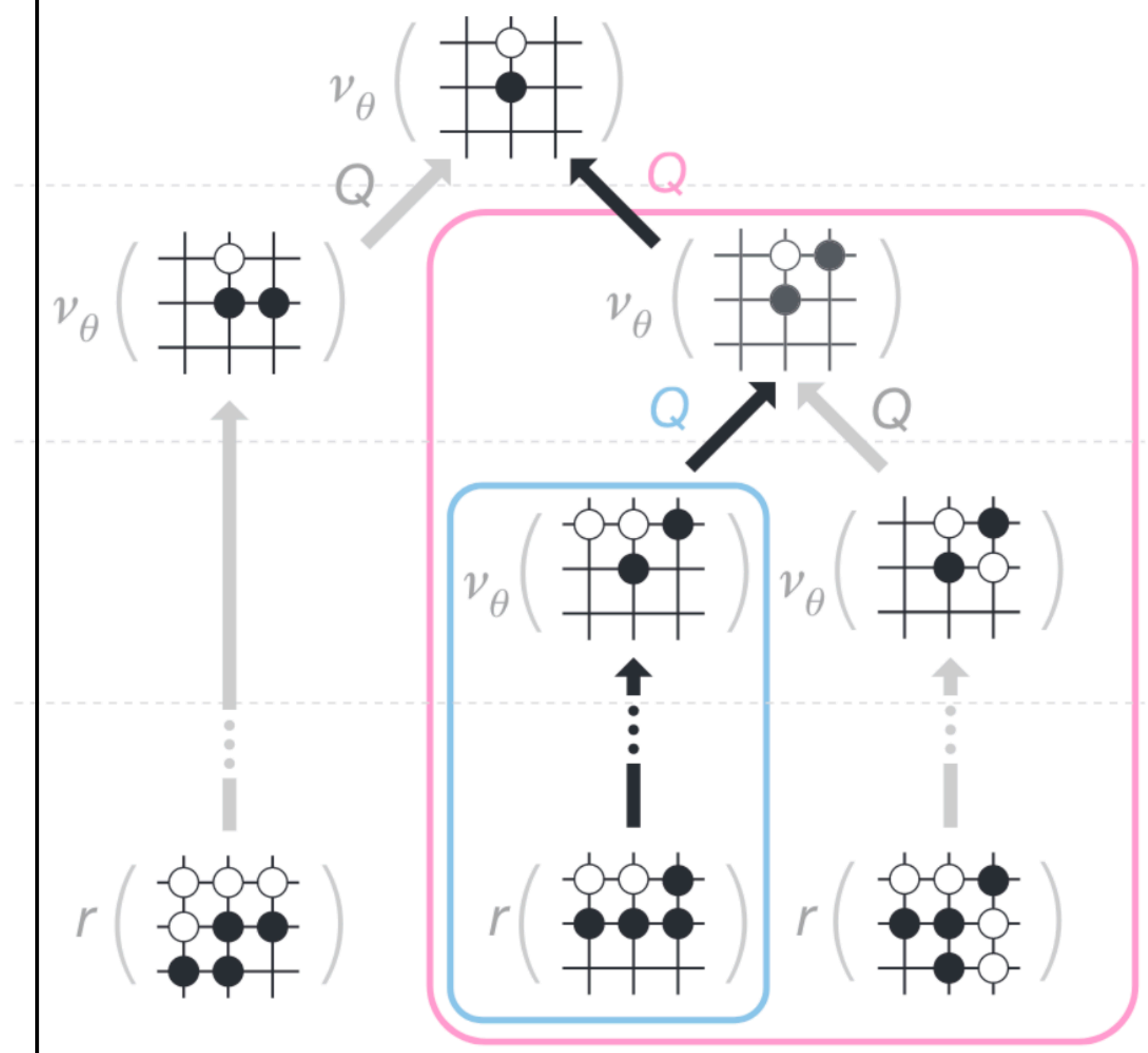
Training Pipeline

(1) Policy Network (2) Rollout Network (3) Value Network



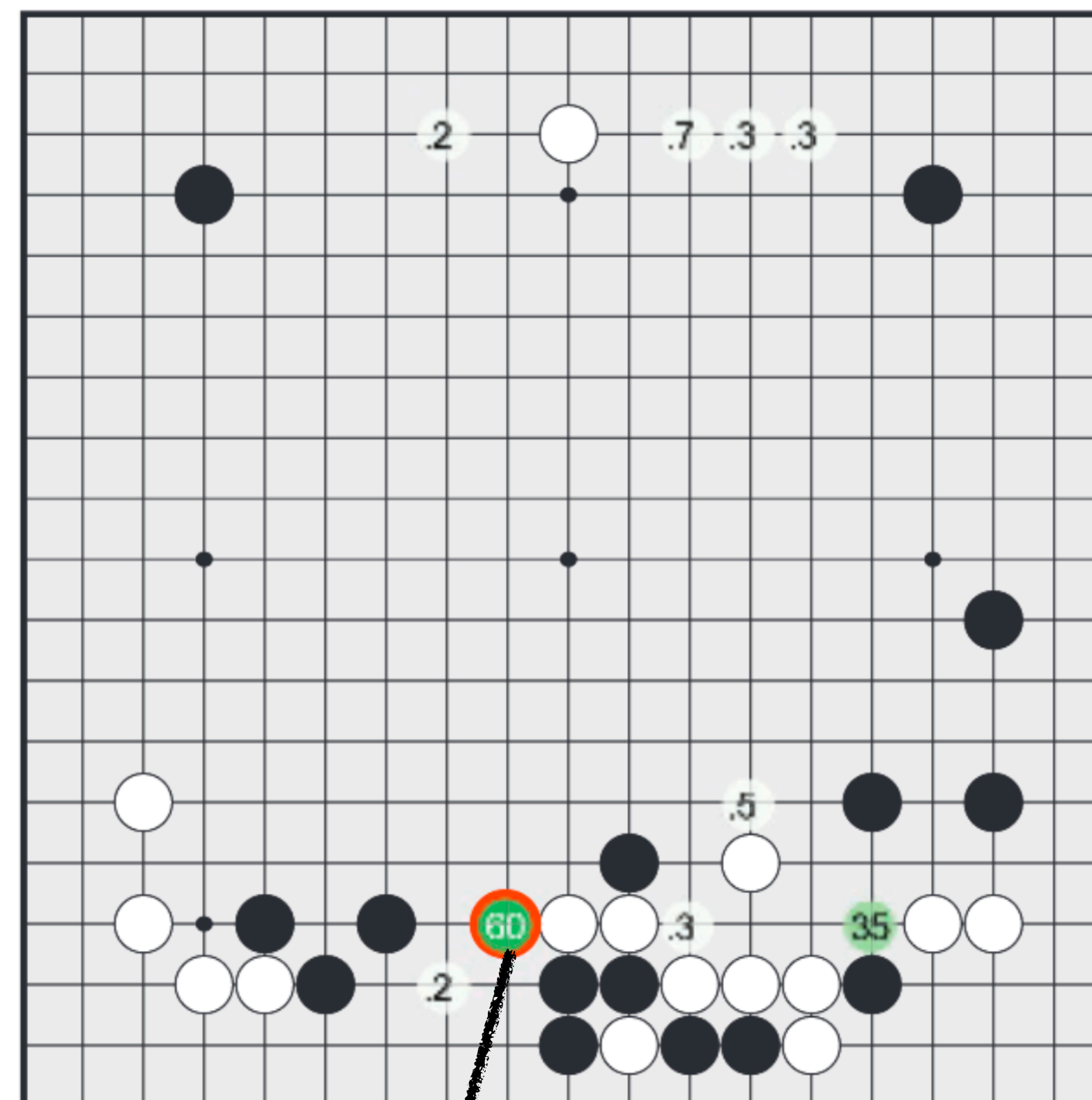
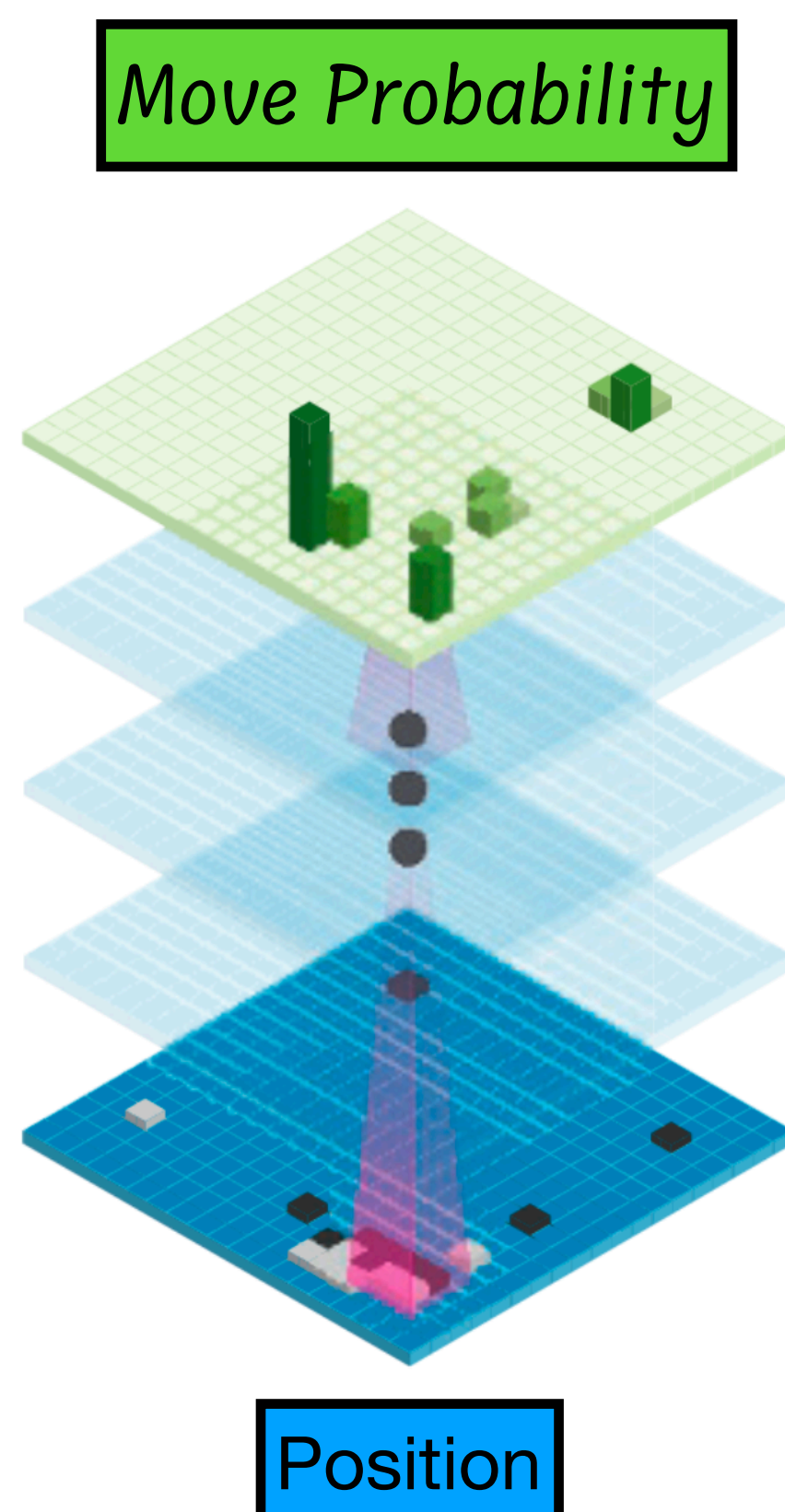
Inference Pipeline

(c) Monte Carlo Tree Search



Training Pipeline: Policy Network

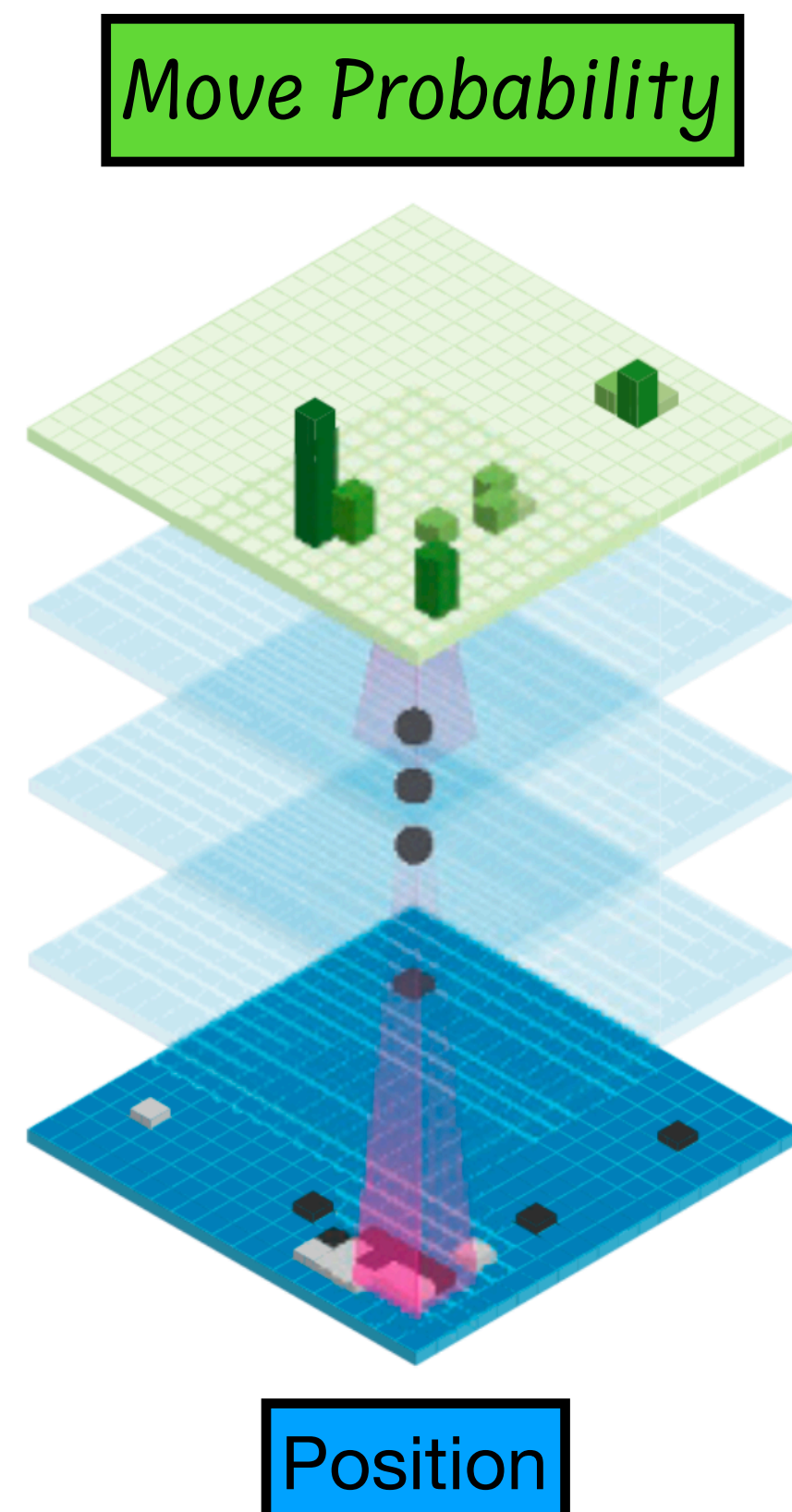
(1) Policy Network



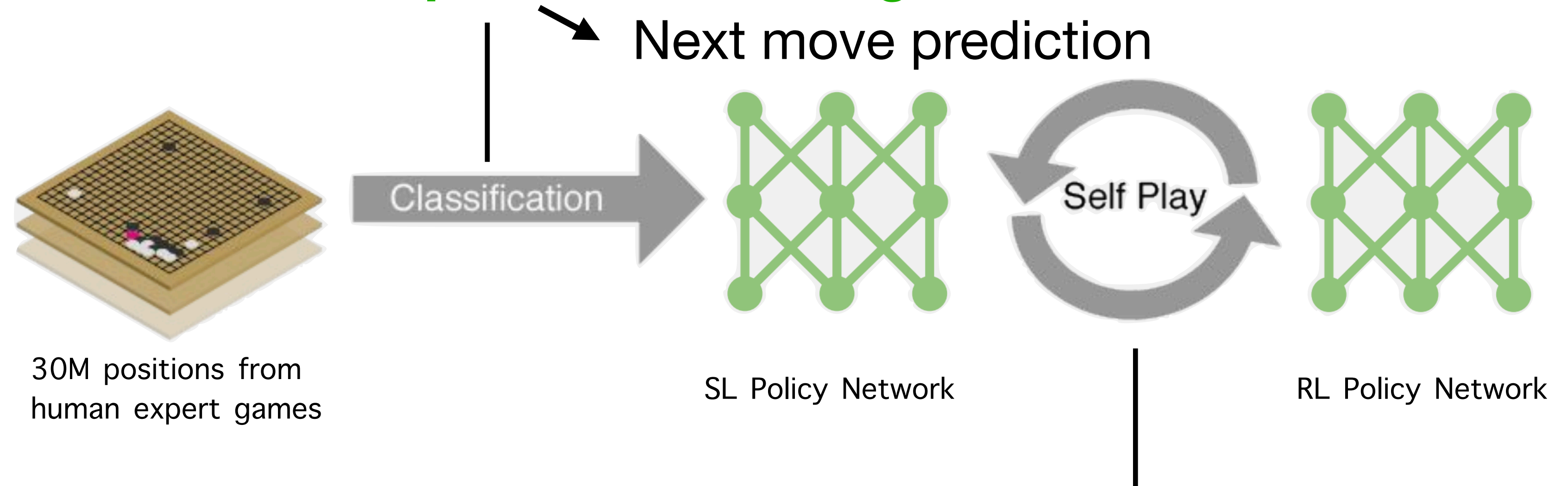
Predicted probability for next moves.

Training Pipeline: Policy Network

(1) Policy Network



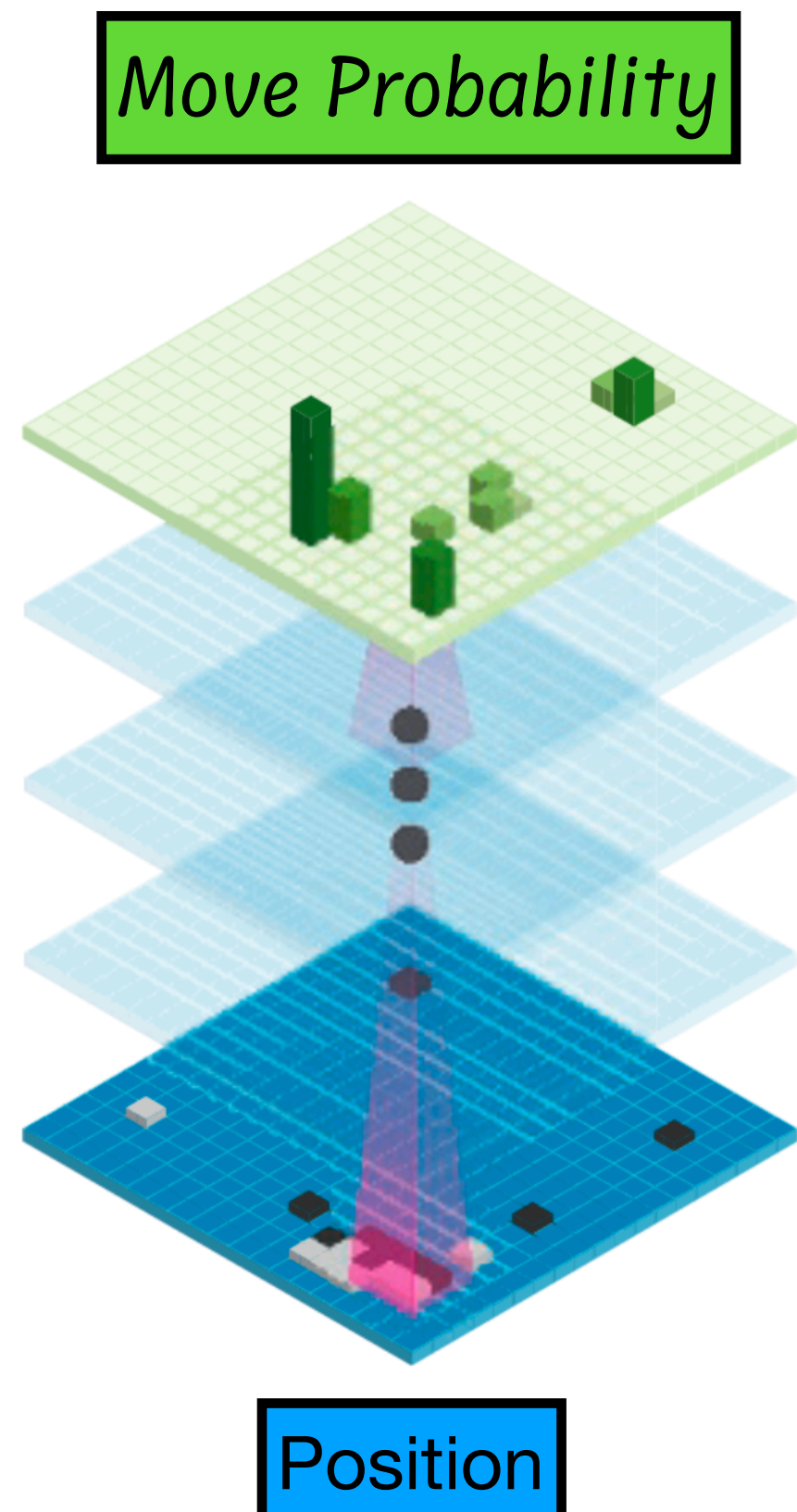
STAGE 1: Supervised Learning on Human Data



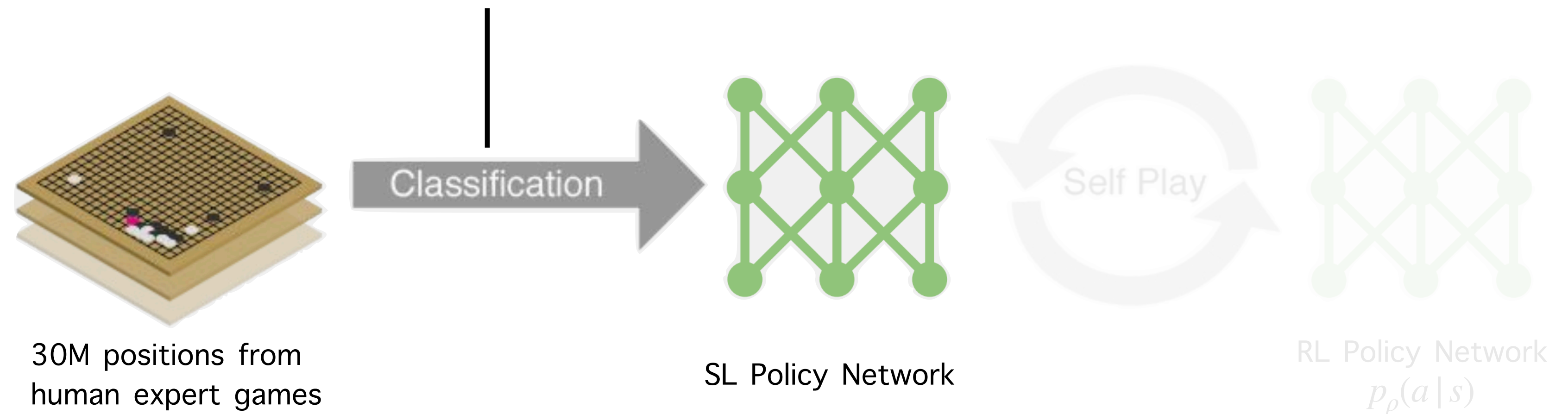
STAGE 2: Reinforcement Learning by Self-Play

Training Pipeline: Policy Network

(1) Policy Network

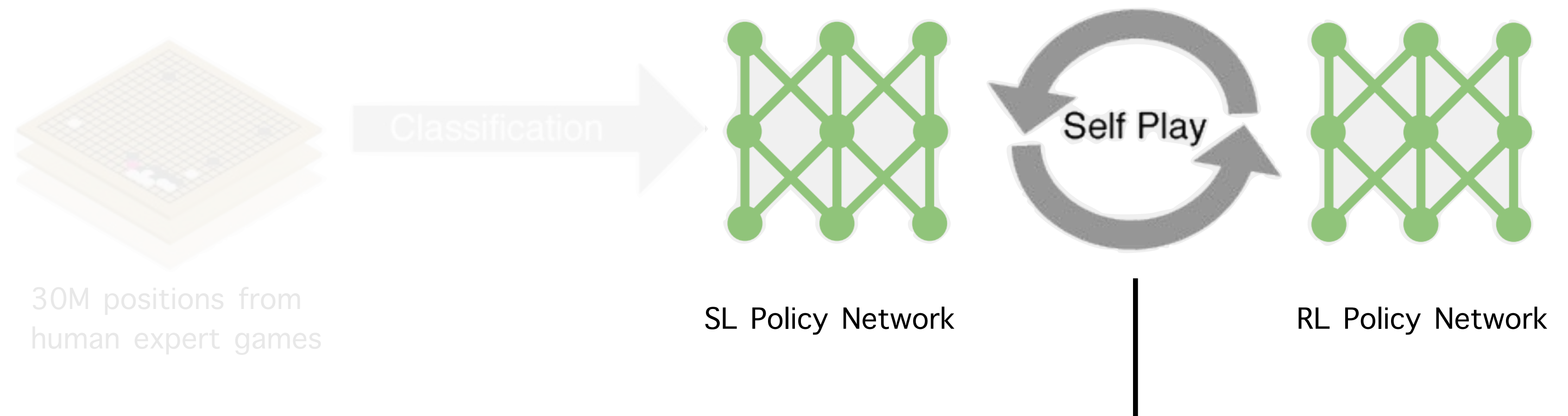
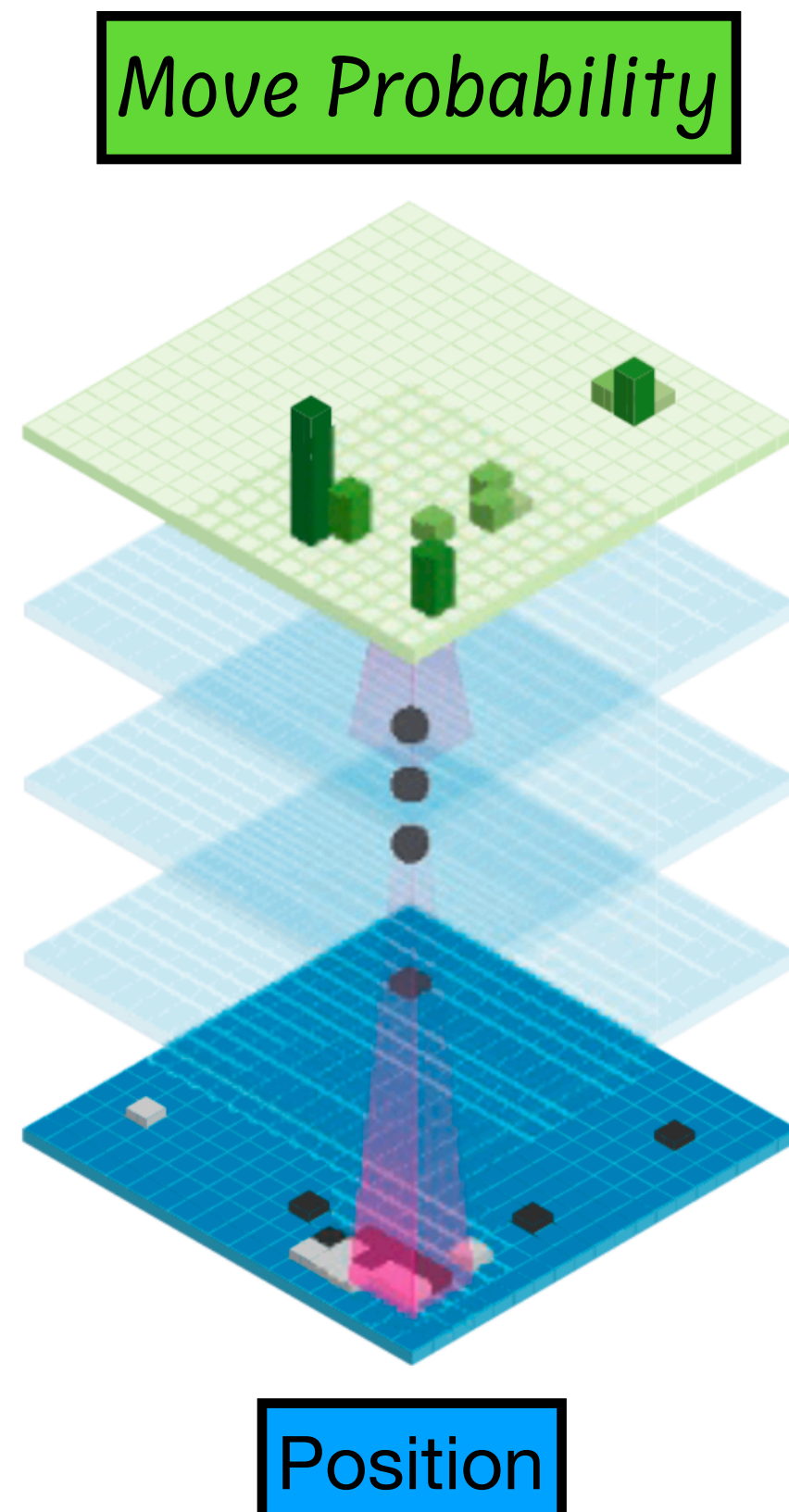


STAGE 1: Supervised Learning on Human Data



Training Pipeline: Policy Network

(1) Policy Network

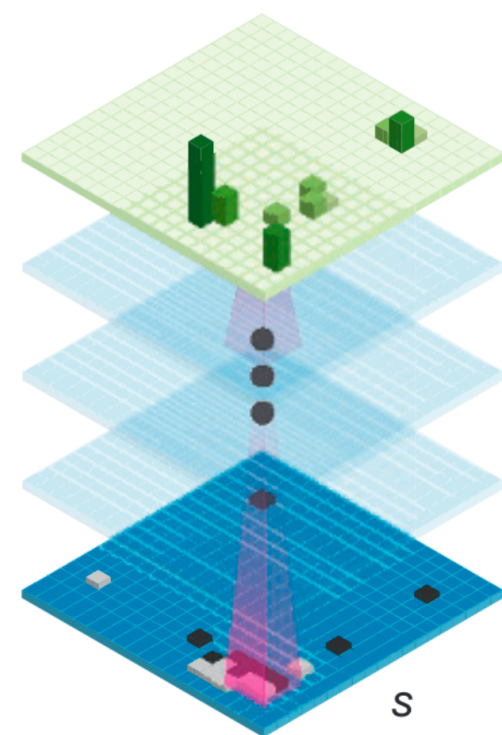


STAGE 2: Reinforcement Learning by Self-Play

Training Pipeline: Rollout Policy

(2) Rollout Network

Move Probability



Position

Rollout Policy Model: faster but less accurate rollout policy

1000x faster



30M positions from human expert games

Classification

Classification



SL Policy Network

Self Play



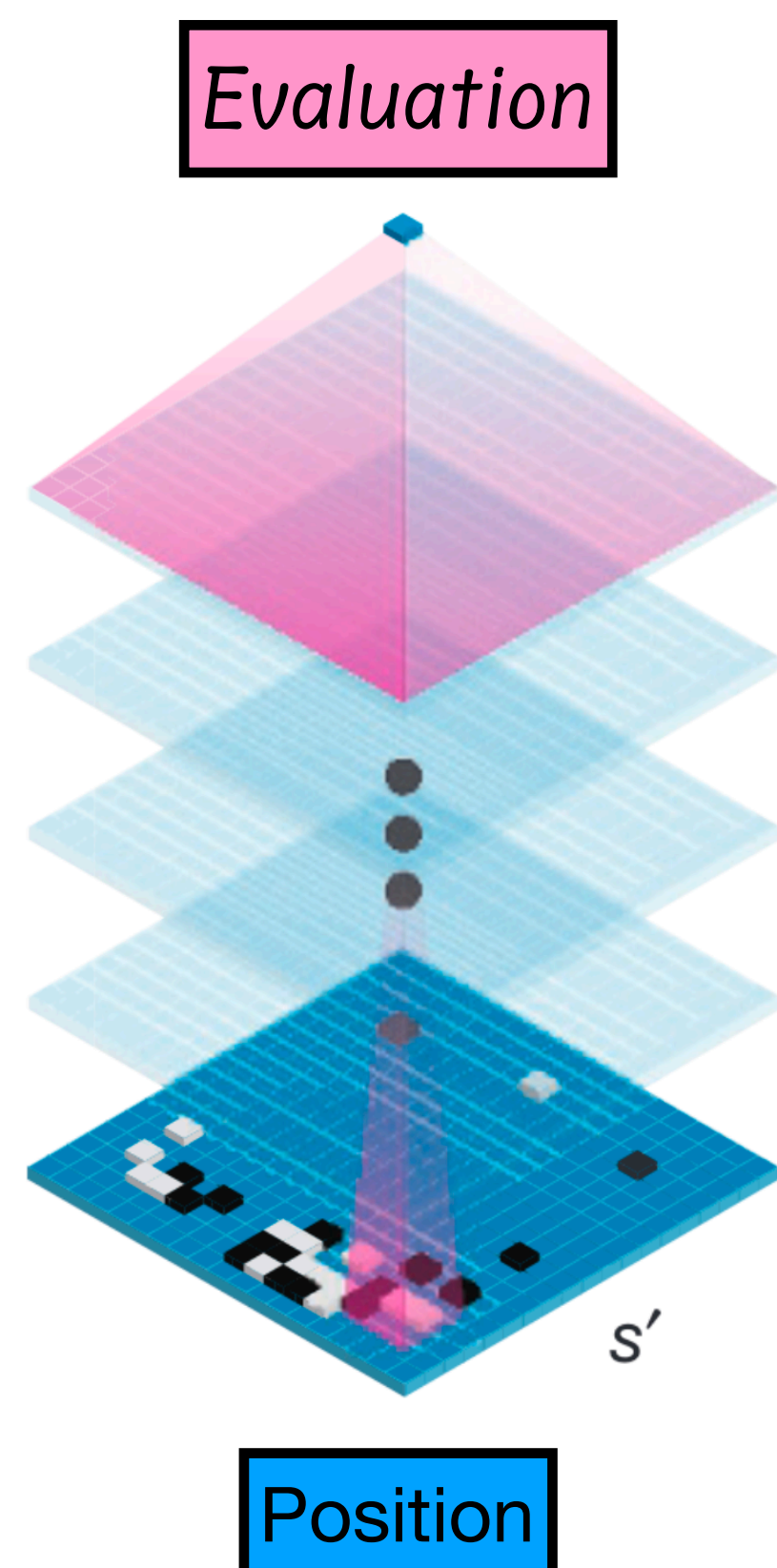
RL Policy Network



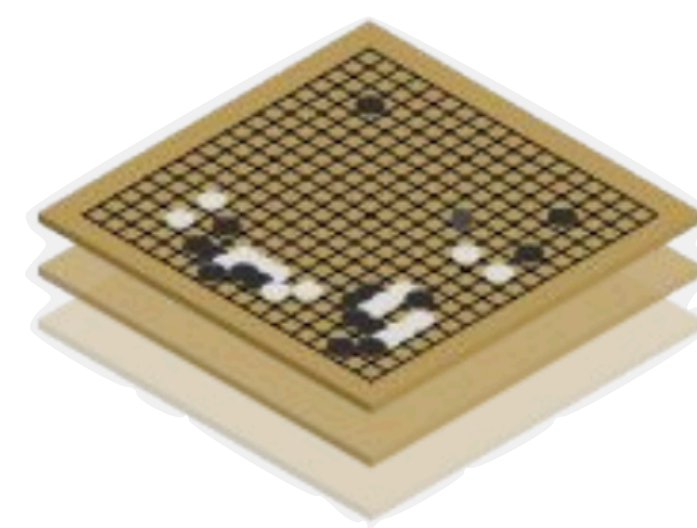
Rollout Policy

Training Pipeline: Value Network

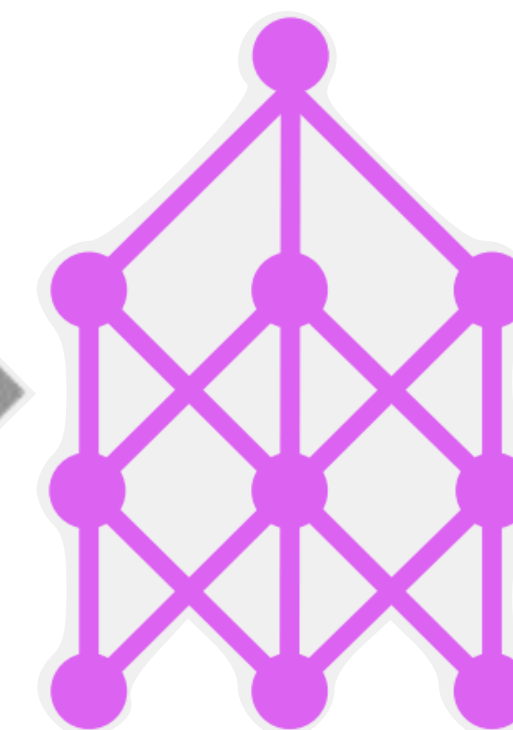
(3) Value Network



Reinforcement Learning by using data from self-play



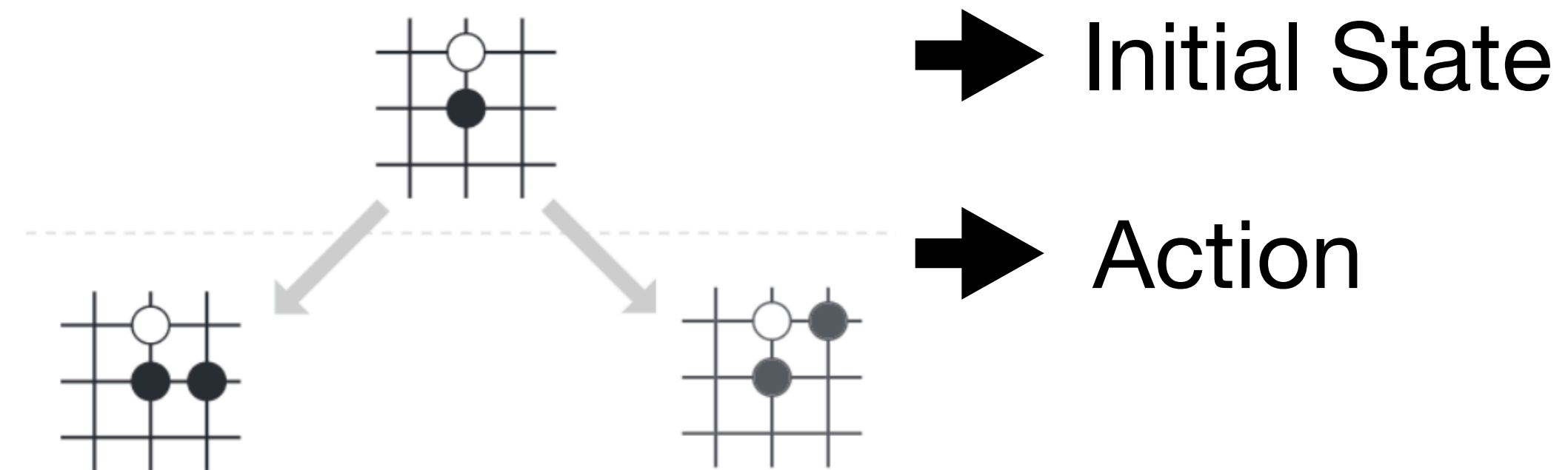
Regression



- 30M games of self-play

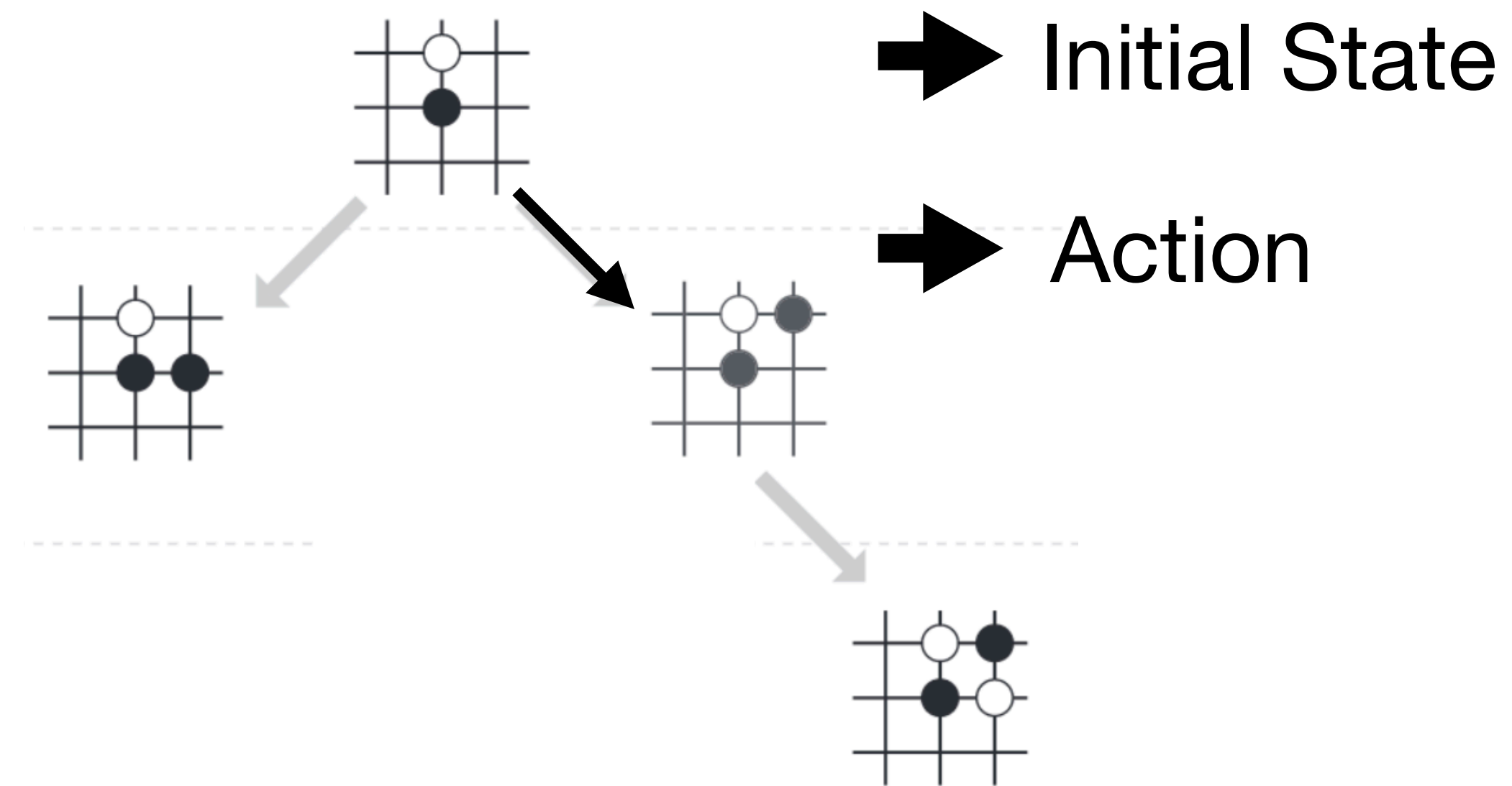
Inference Pipeline: MCTS

- Monte Carlo Tree Search (MCTS) for online planning and lookahead search.



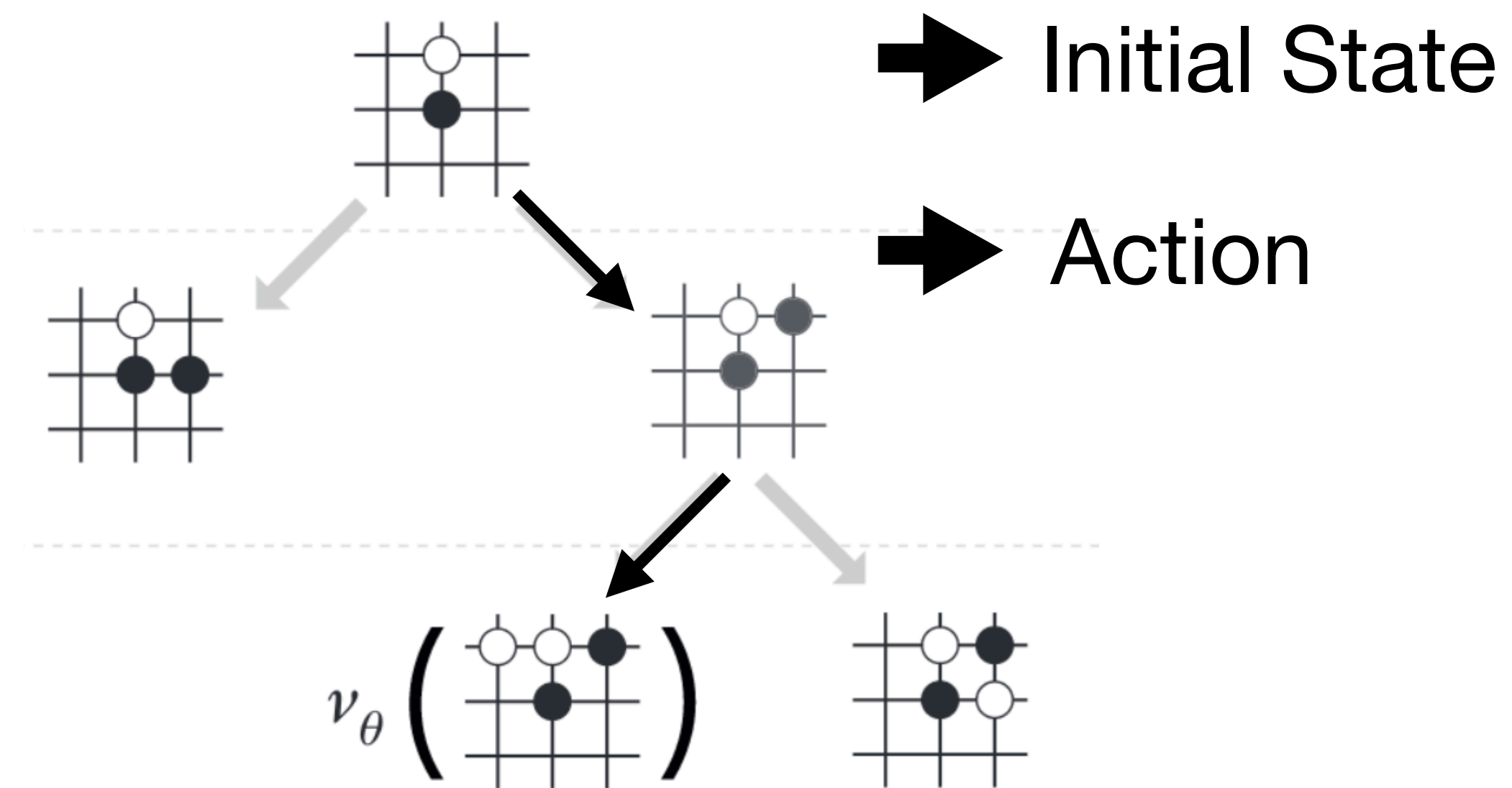
Inference Pipeline: MCTS

- Monte Carlo Tree Search (MCTS) for online planning and lookahead search.



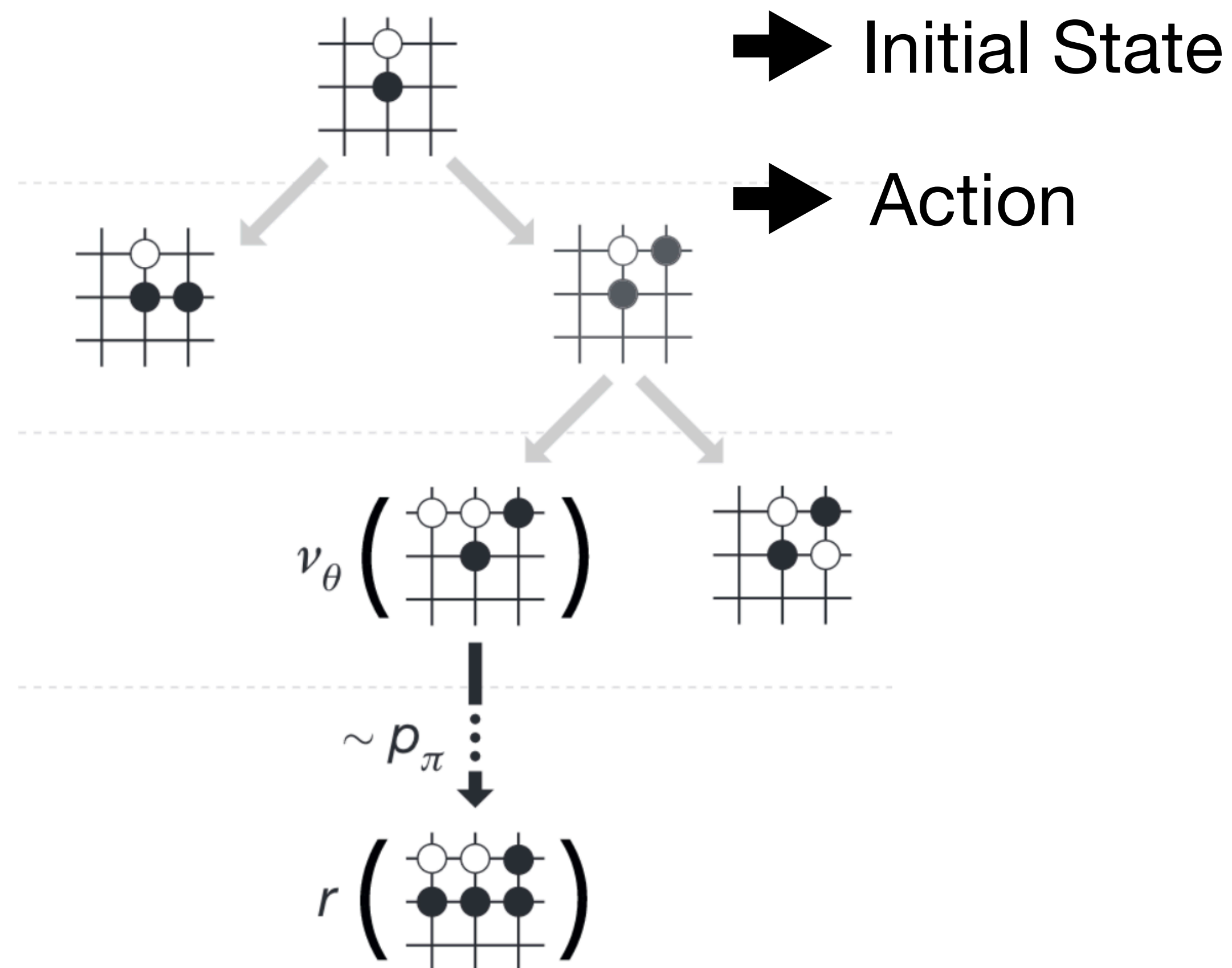
Inference Pipeline: MCTS

- Monte Carlo Tree Search (MCTS) for online planning and lookahead search.



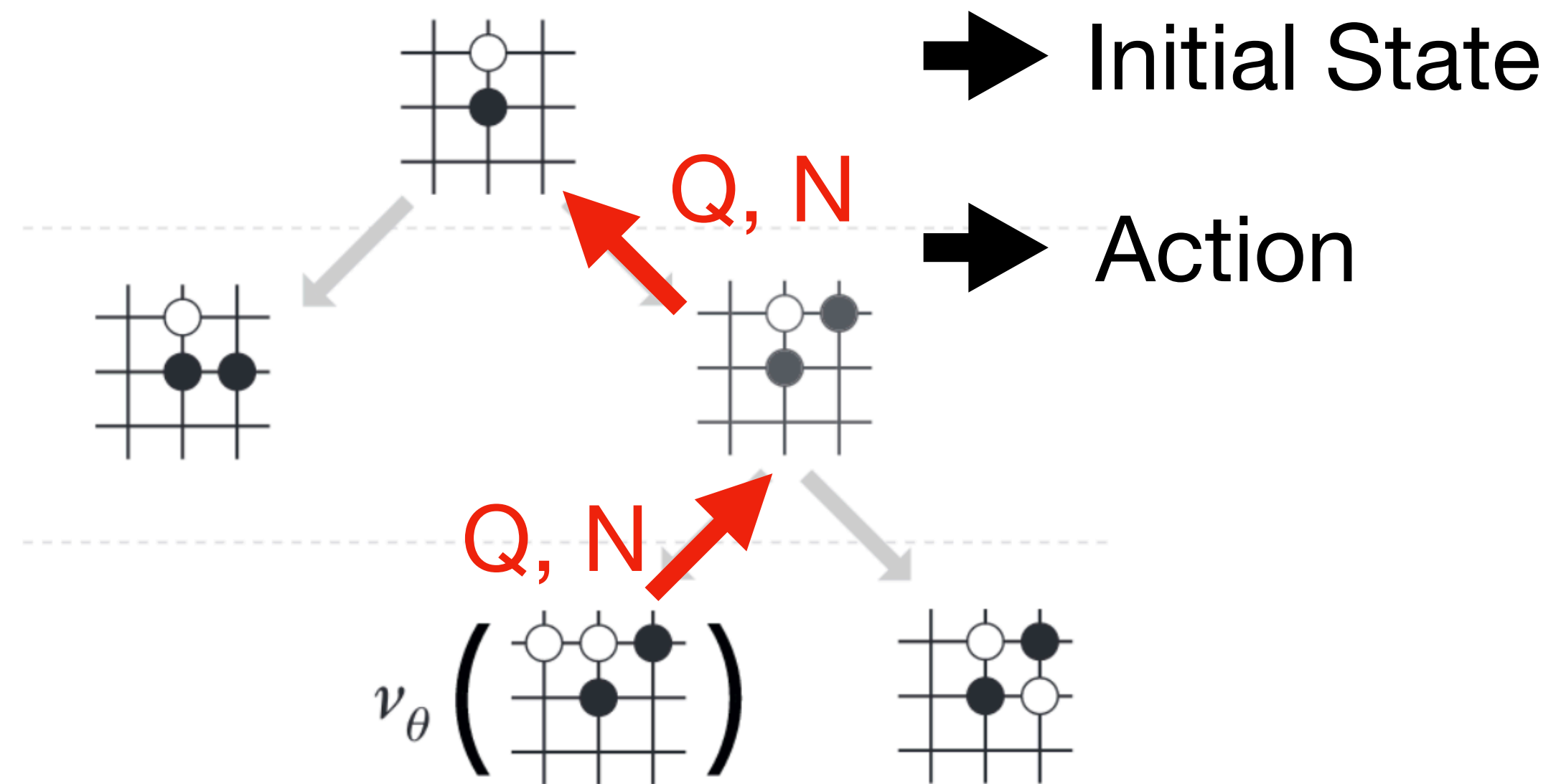
Inference Pipeline: MCTS

- Monte Carlo Tree Search (MCTS) for online planning and lookahead search.



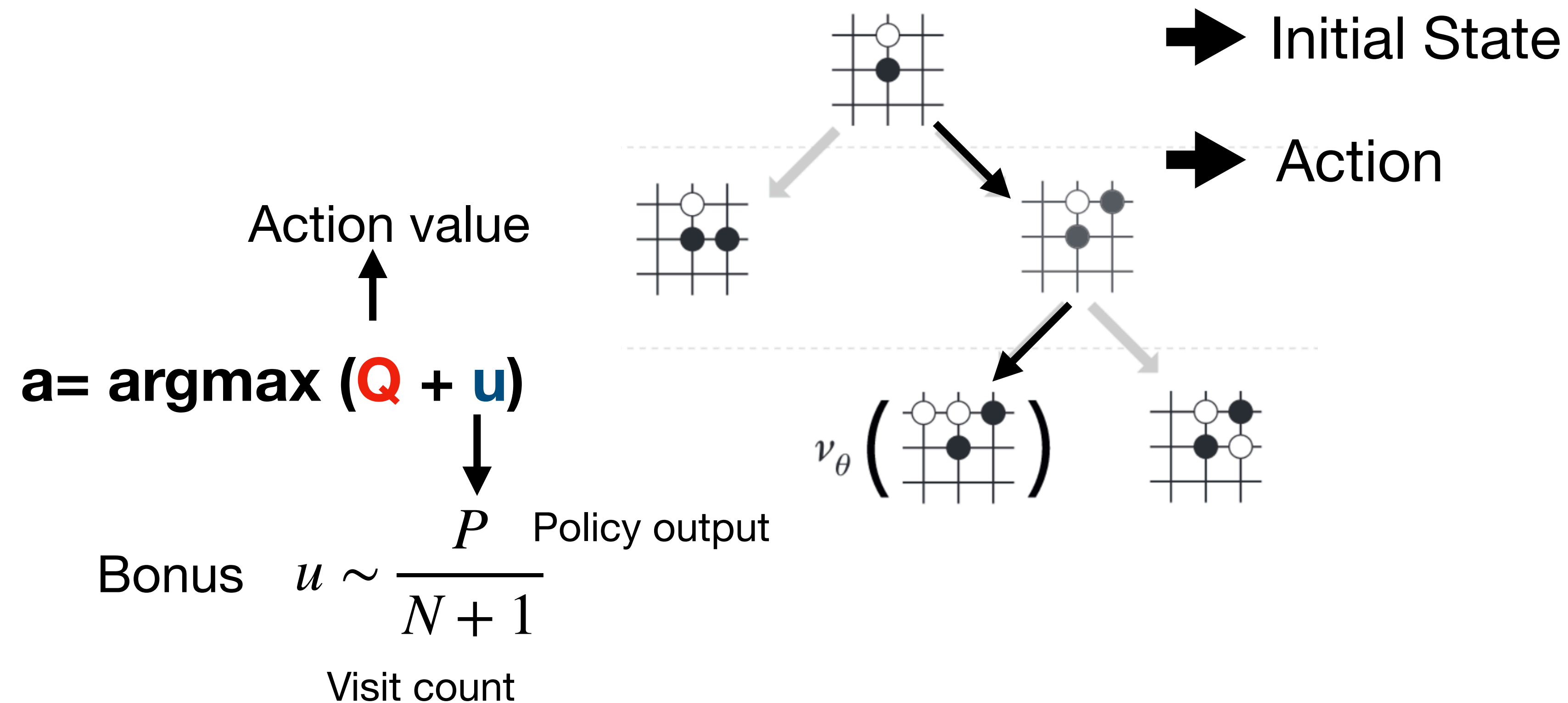
Inference Pipeline: MCTS

- Monte Carlo Tree Search (MCTS) for online planning and lookahead search.



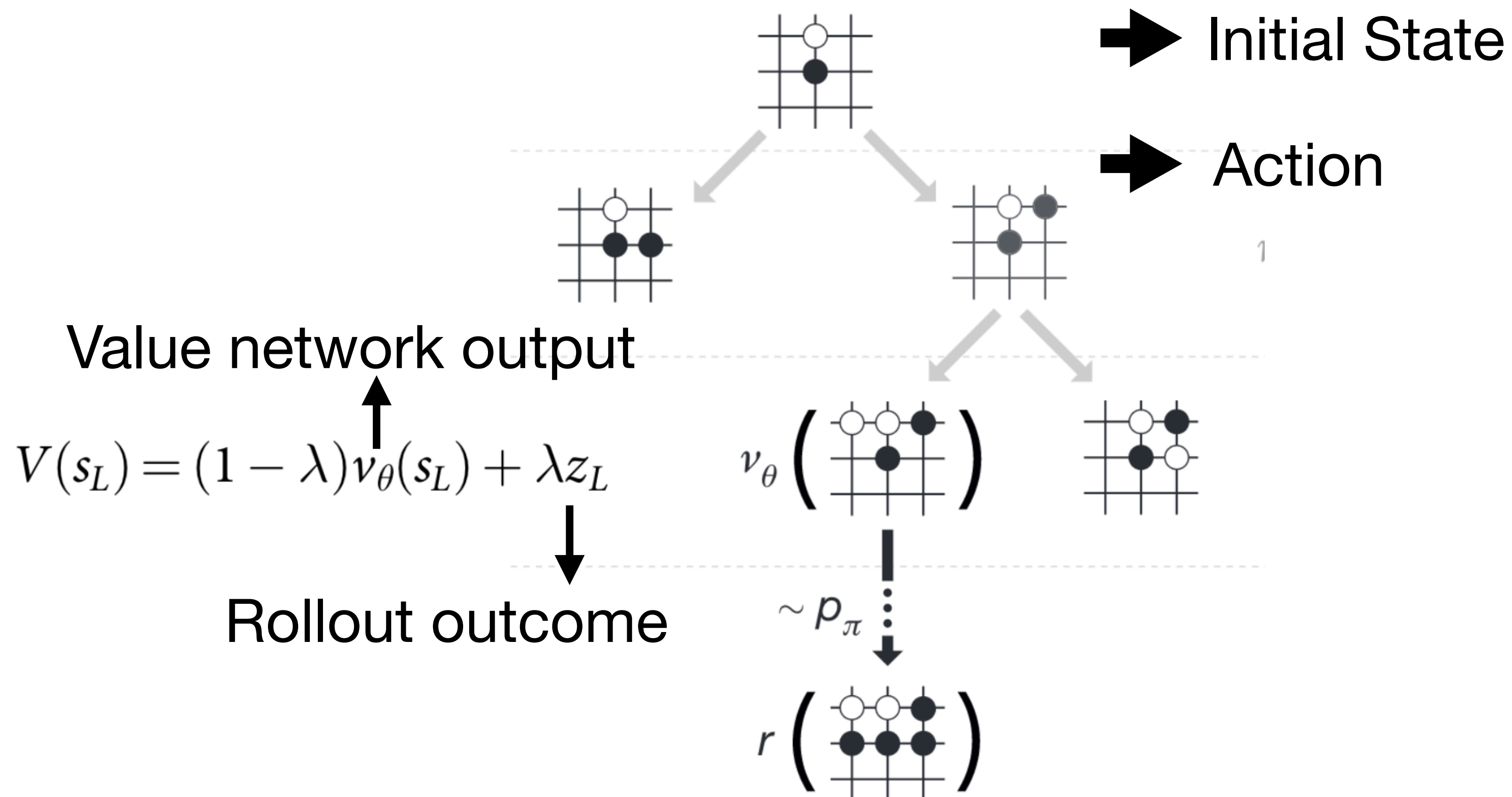
Inference Pipeline: MCTS

- Monte Carlo Tree Search (MCTS) for online planning and lookahead search.

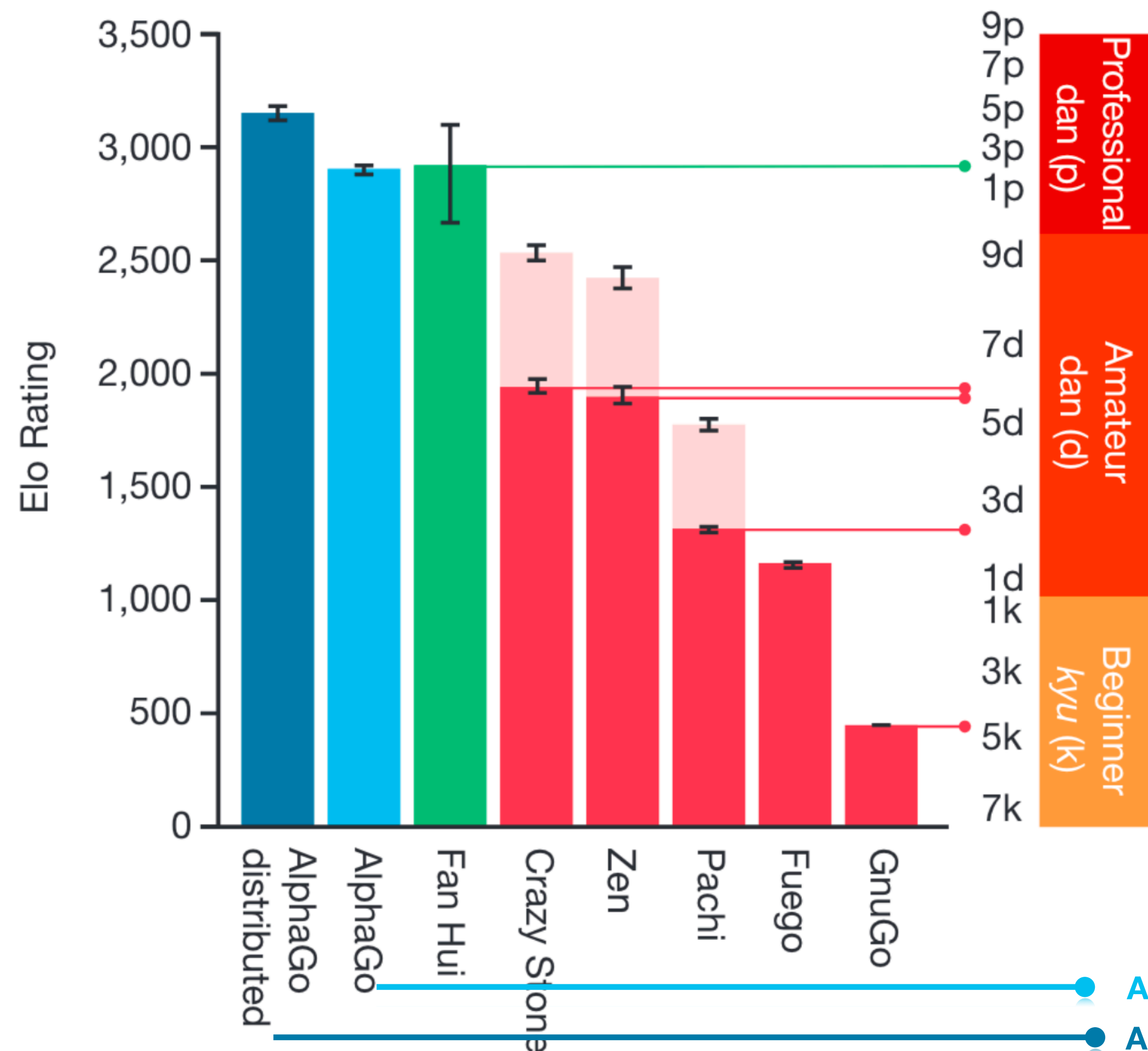


Inference Pipeline: MCTS

- Monte Carlo Tree Search (MCTS) for online planning and lookahead search.



Results



Fixed Inference Time: 5s of computation time per move

AlphaGo is many dan ranks **stronger than any **previous Go program**!**

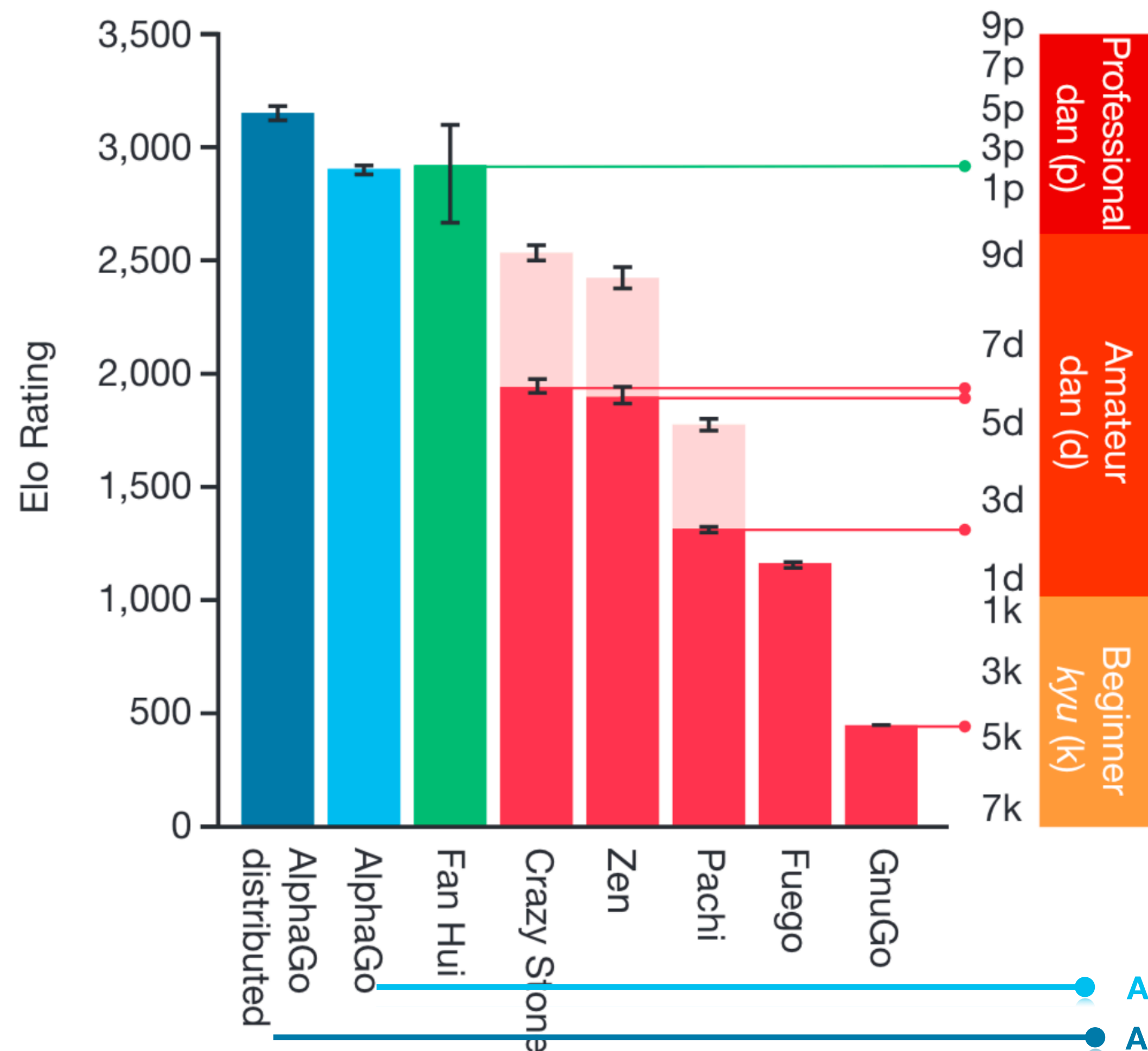
- Single-machine version win 494 out of 495 games (99.8%).
- Distributed version win 100%.

Alpha Go: 40 search threads, 48 CPUs and 8GPUs

Alpha Go Distributed: 40 search threads, 1,202 CPUs and 176 GPUs

Elo Gap vs. Winning Rate: $E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$ E.g., a 1000 Elo difference corresponds to a 99% winning rate.

Results



Fixed Inference Time: 5s of computation time per move

AlphaGo is many dan ranks **stronger than any **previous Go program**!**

- Single-machine version win 494 out of 495 games (99.8%).
- Distributed version win 100%.

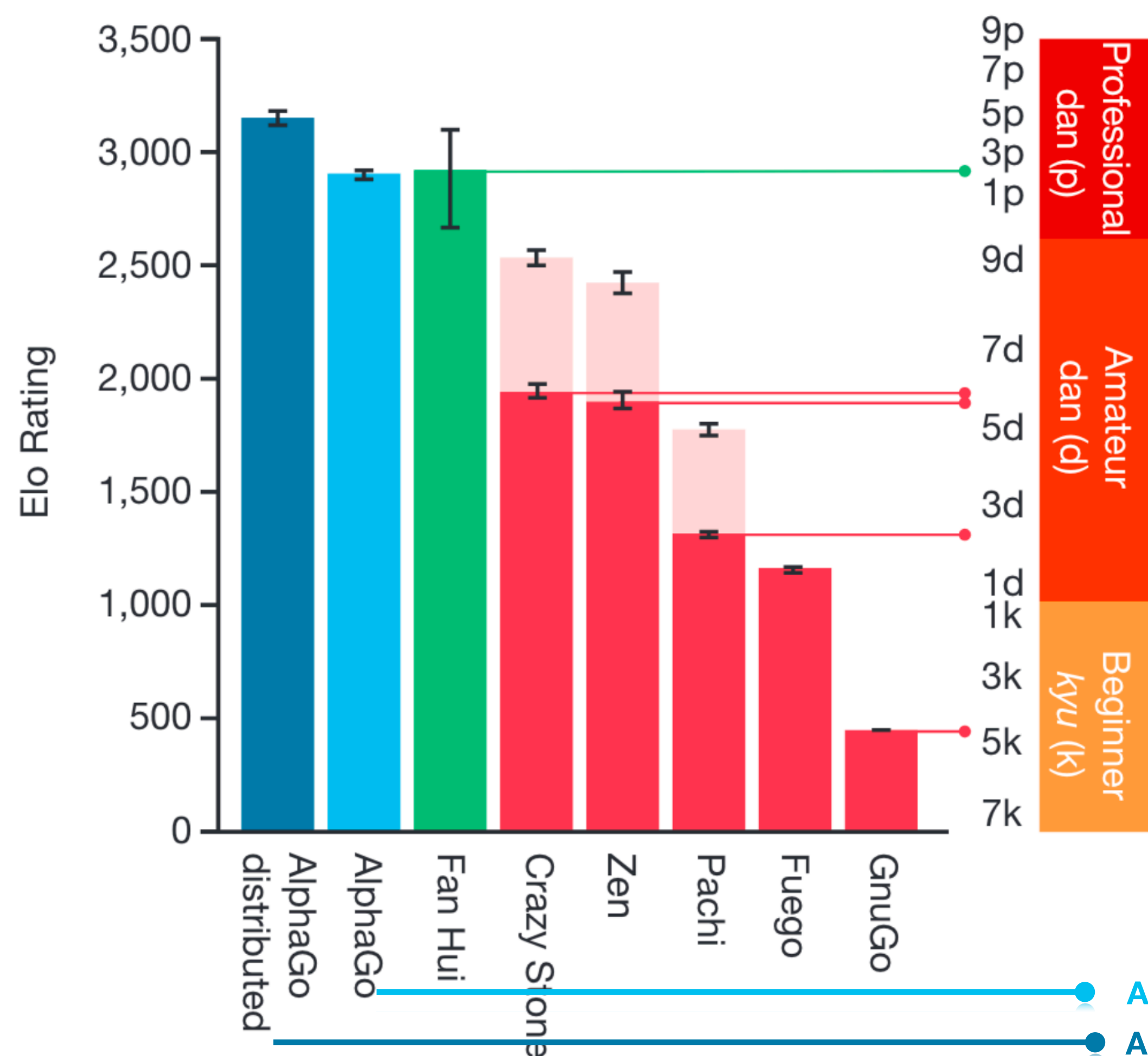
The distributed version of AlphaGo **beat professional *2-dan* player **Fan Hui** in a formal five-game match.**

● **Alpha Go:** 40 search threads, 48 CPUs and 8GPUs

● **Alpha Go Distributed:** 40 search threads, 1,202 CPUs and 176 GPUs

Elo Gap vs. Winning Rate: $E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$ E.g., a 1000 Elo difference corresponds to a 99% winning rate.

Results



Fixed Inference Time: 5s of computation time per move

AlphaGo is many dan ranks **stronger than any **previous Go program**!**

- Single-machine version win 494 out of 495 games (99.8%).
- Distributed version win 100%.

The distributed version of AlphaGo **beat professional *2-dan* player **Fan Hui** in a formal five-game match.**

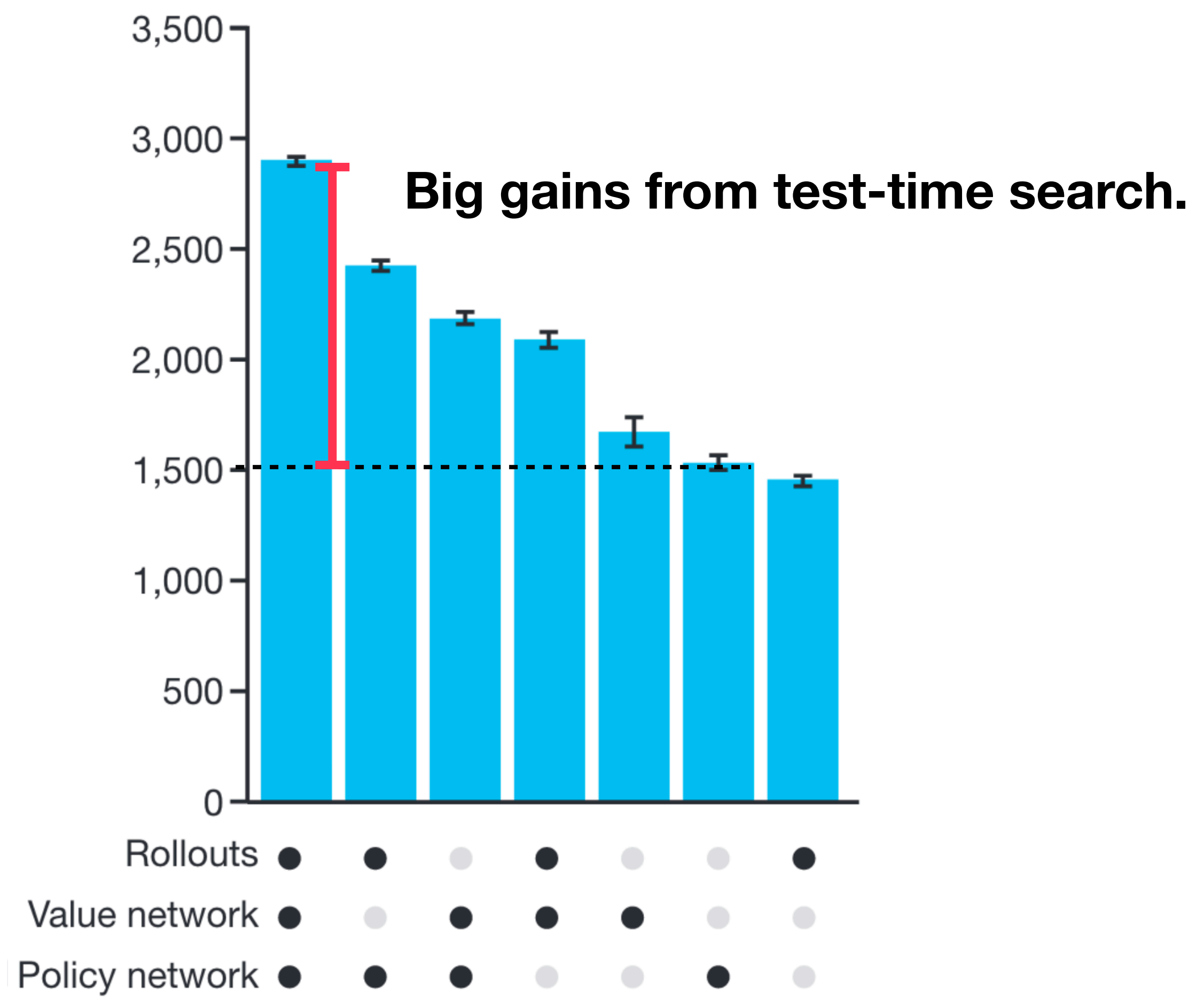
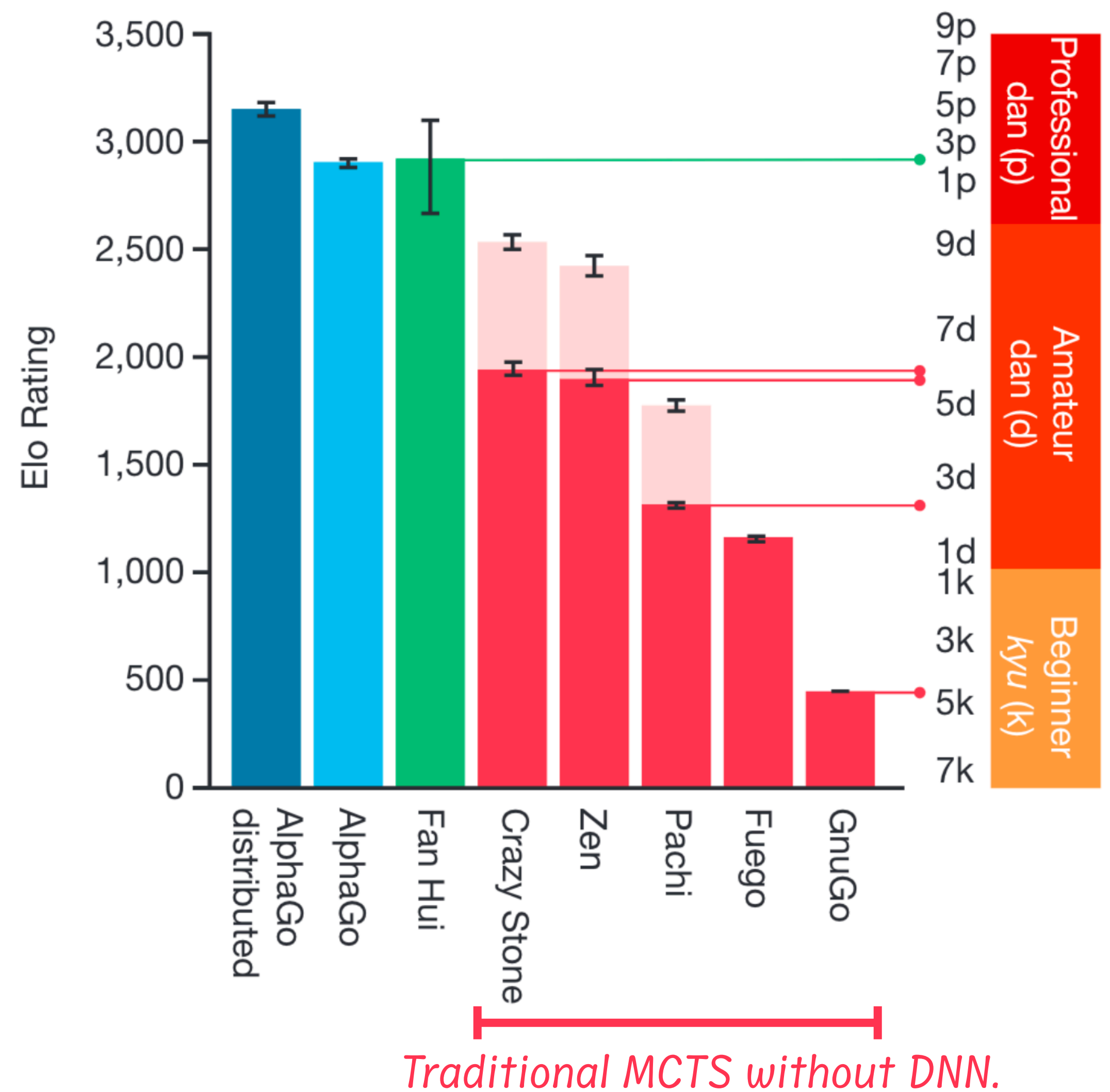
One year later, a further scaled version of AlphaGo beat **Lee Sedol, a *9-dan* professional (the highest professional rank).**

Alpha Go: 40 search threads, 48 CPUs and 8GPUs

Alpha Go Distributed: 40 search threads, 1,202 CPUs and 176 GPUs

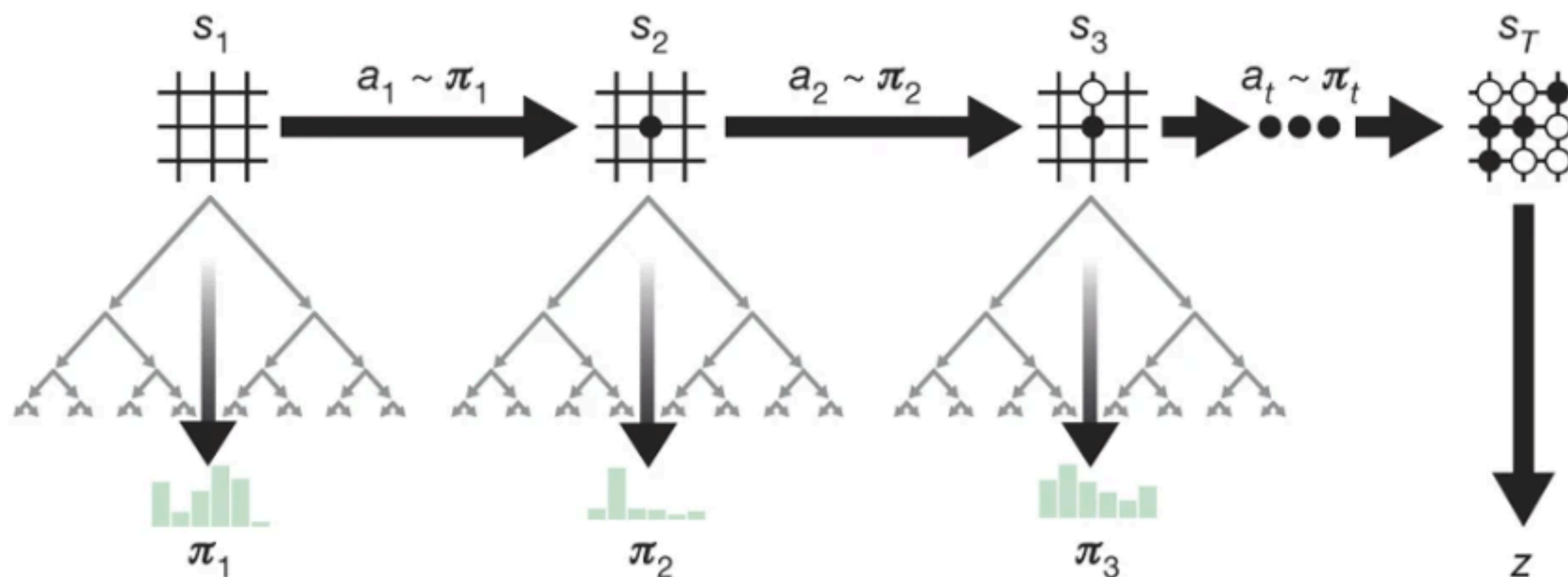
Elo Gap vs. Winning Rate: $E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$ E.g., a 1000 Elo difference corresponds to a 99% winning rate.

Results



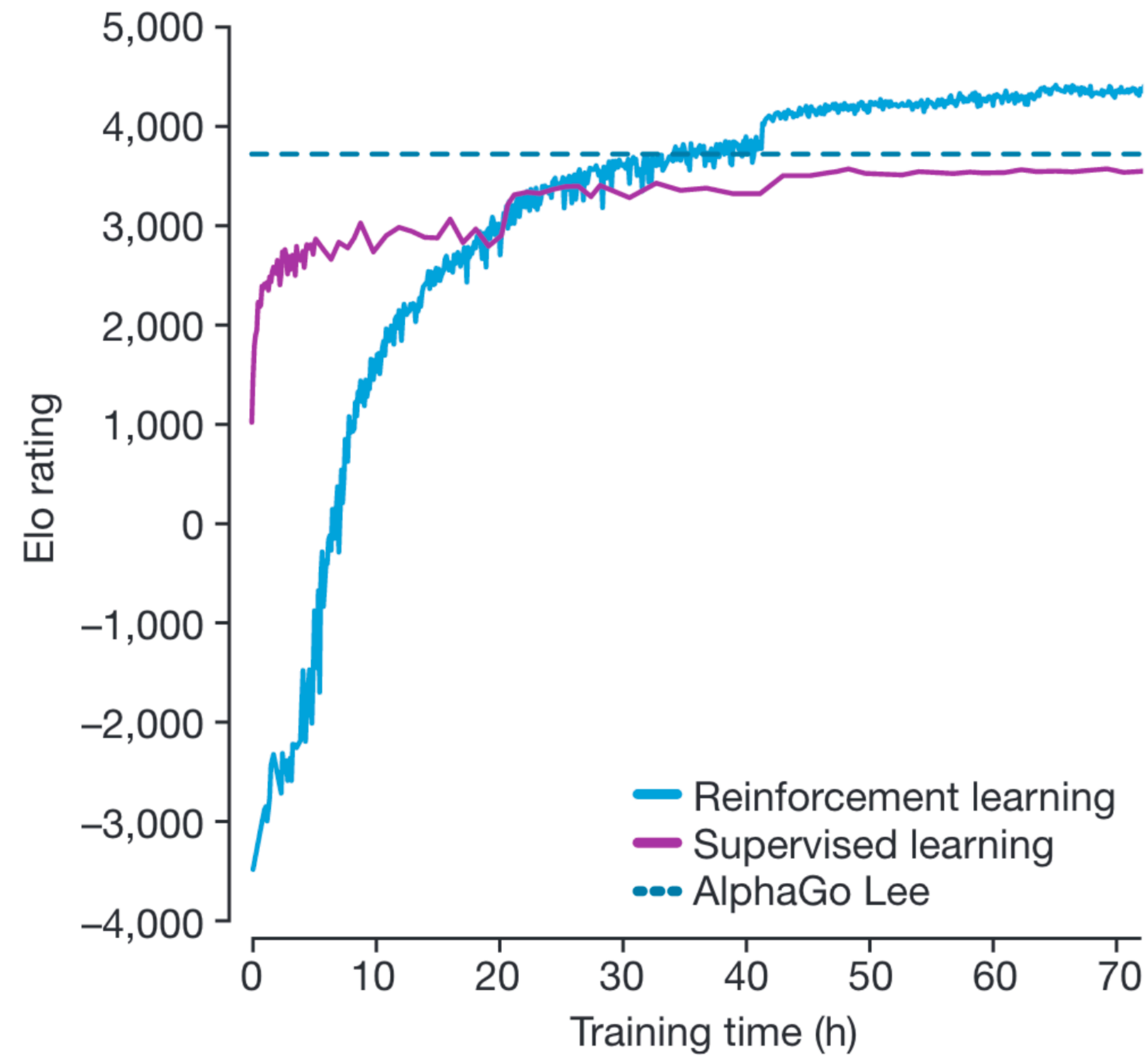
From AlphaGo to AlphaGo Zero (2017)

1. Single network with from-scratch RL self-play.
2. Search during both **training & testing**.



[1] Silver, D., Schrittwieser, J., Simonyan, K. *et al.* Mastering the game of Go without human knowledge. *Nature* 550, 354–359 (2017)

From AlphaGo to AlphaGo Zero (2017)



- **AlphaGo Zero defeated AlphaGo** after **72 hours** of training, whereas AlphaGo was trained over several months.

Takeaways



Search is a powerful method for planning at test time.



MCTS may not be suitable for more complex open problems.

2nd paper

DeepSeek-R1: Incentivizing *Reasoning* Capability in LLMs via Reinforcement Learning

DeepSeek-AI. 2025



DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI
research@deepseek.com

Abstract

We introduce our first-generation reasoning models, DeepSeek-R1-Zero and DeepSeek-R1. DeepSeek-R1-Zero, a model trained via large-scale reinforcement learning (RL) without supervised fine-tuning (SFT) as a preliminary step, demonstrates remarkable reasoning capabilities. Through RL, DeepSeek-R1-Zero naturally emerges with numerous powerful and intriguing reasoning behaviors. However, it encounters challenges such as poor readability, and language mixing. To address these issues and further enhance reasoning performance, we introduce DeepSeek-R1, which incorporates multi-stage training and cold-start data before RL. DeepSeek-R1 achieves performance comparable to OpenAI-o1-1217 on reasoning tasks. To support the research community, we open-source DeepSeek-R1-Zero, DeepSeek-R1, and six dense models (1.5B, 7B, 8B, 14B, 32B, 70B) distilled from DeepSeek-R1 based on Qwen and Llama.

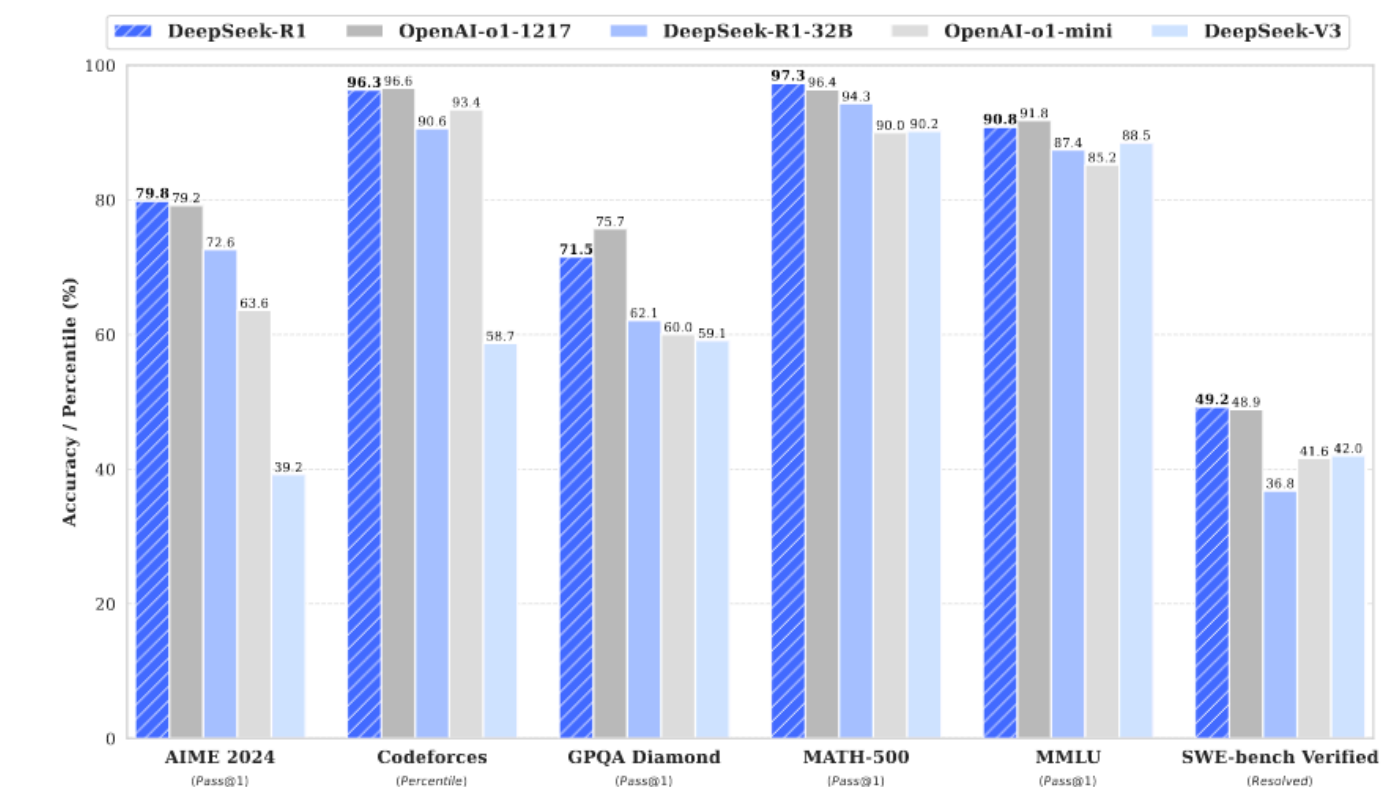


Figure 1 | Benchmark performance of DeepSeek-R1.

arXiv:2501.12948v1 [cs.CL] 22 Jan 2025

Problem: Reasoning in LLMs

Math Benchmark (eg. 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 $\{1, 2, 3, \dots, n\}$ 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.

[Hendrycks et al. Measuring mathematical problem solving with the MATH dataset, 2021]

Coding Benchmark (eg. Codeforce)

You are given a permutation $p_1, p_2, \dots, p_n$ of length n .

You have to build an array $a_1, a_2, \dots, a_n$ in the following way:

- for each $1 \leq i \leq n$, set either $a_i = p_i$ or $a_i = 2n - p_i$.

Find the minimum possible number of inversions in the array $a_1, a_2, \dots, a_n$.

A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

An inversion in the array $a_1, a_2, \dots, a_n$ is a pair of indices (i, j) such that $1 \leq i < j \leq n$ and $a_i > a_j$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^3$). The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \leq n \leq 5 \cdot 10^3$).

The second line of each test case contains n integers $p_1, p_2, \dots, p_n$ ($1 \leq p_i \leq n$). It is guaranteed that $p_1, p_2, \dots, p_n$ is a permutation.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^3$.

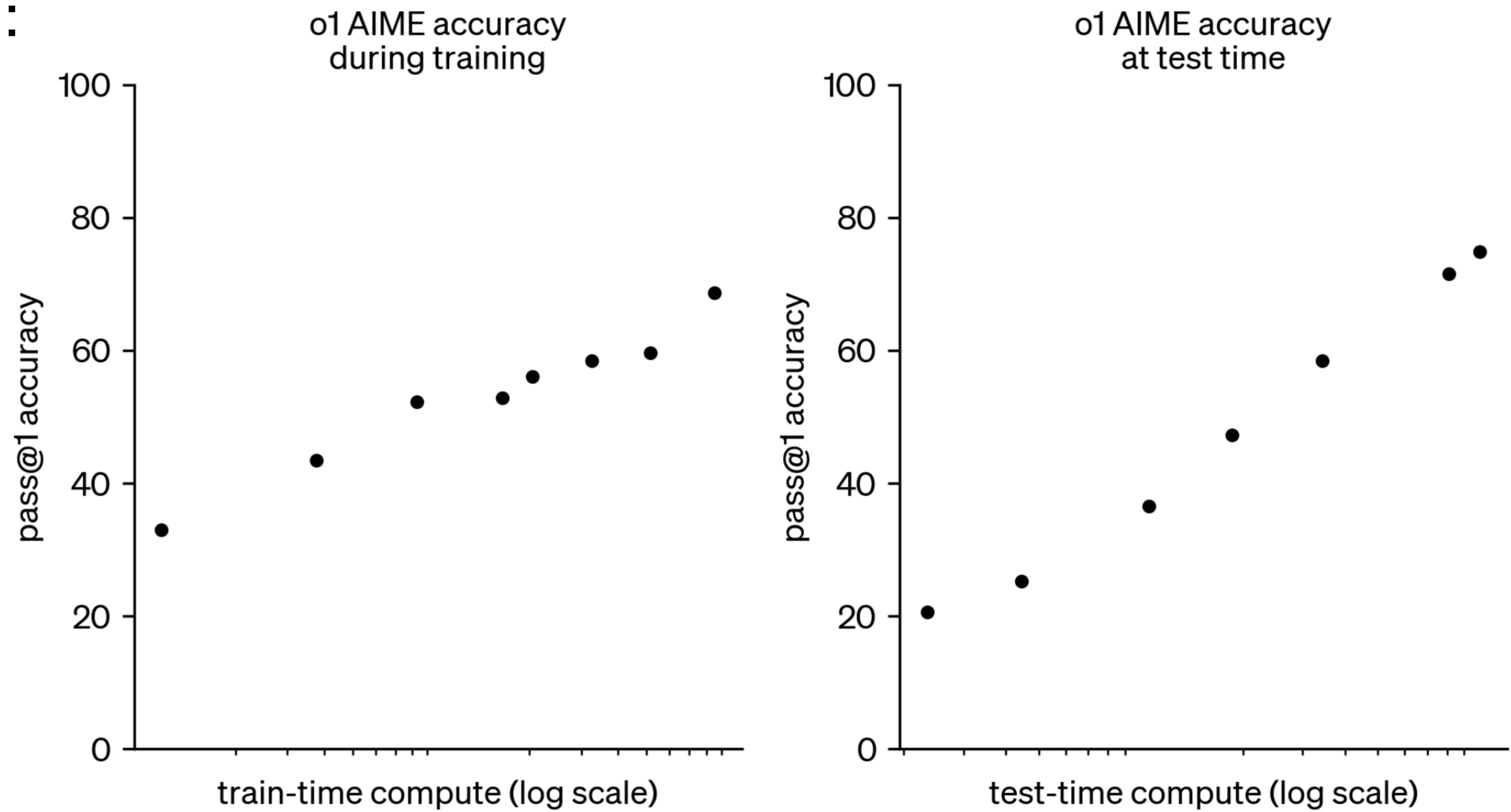
Output

For each test case, print a single integer — the minimum number of inversions in the array a .

Example from <https://codeforces.com>.

Background: Effective Test-time Scaling techniques in LLMs

OpenAI o1:



[Learning to reason with LLMs, OpenAI, 2024]

Background: Effective Test-time Scaling techniques in LLMs

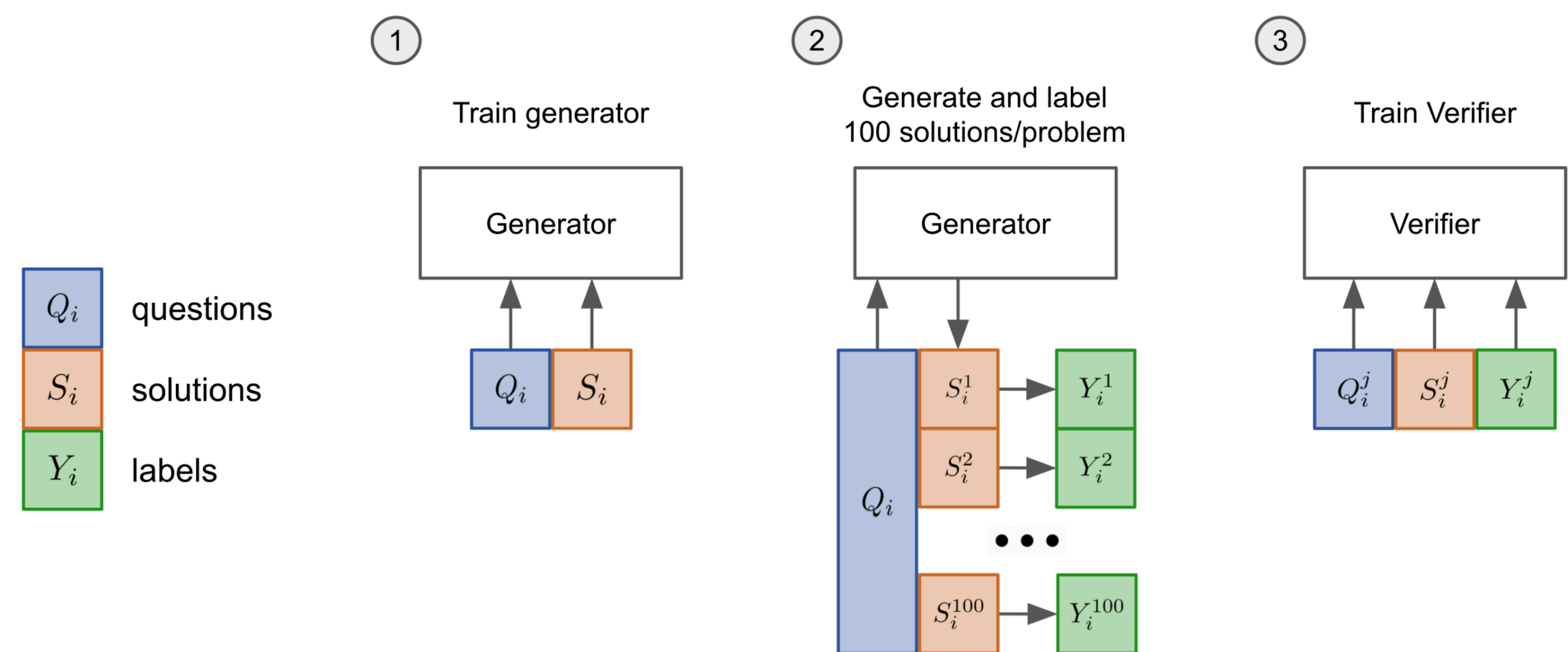
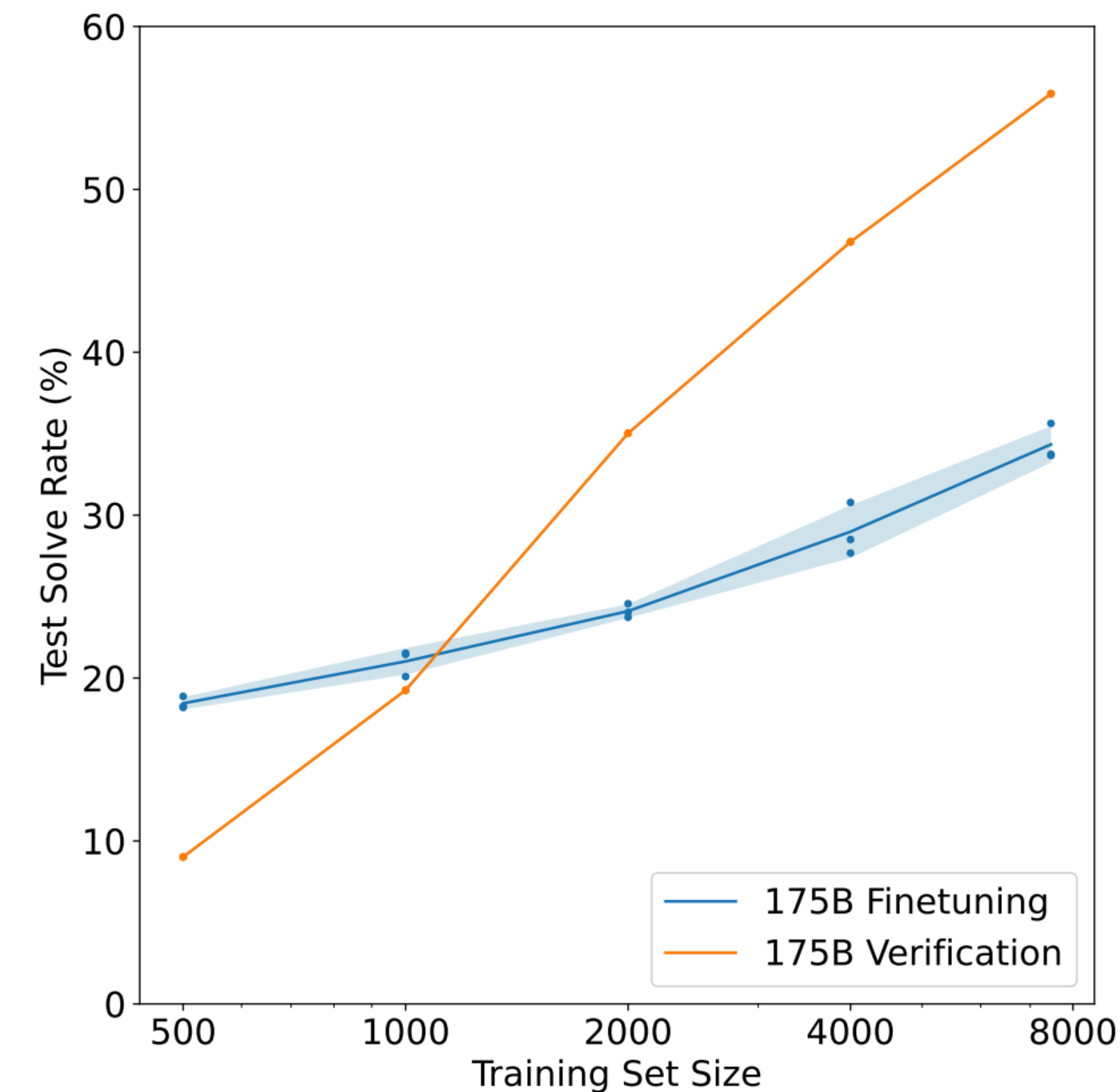


Figure 4: A diagram of the verification training pipeline.

[OpenAI. Training Verifiers to Solve Math Word Problems, 2021]



“We provide evidence that **verifiers scale far more favorably** with additional data than baseline methods.”

Background: Effective Test-time Scaling techniques in LLMs

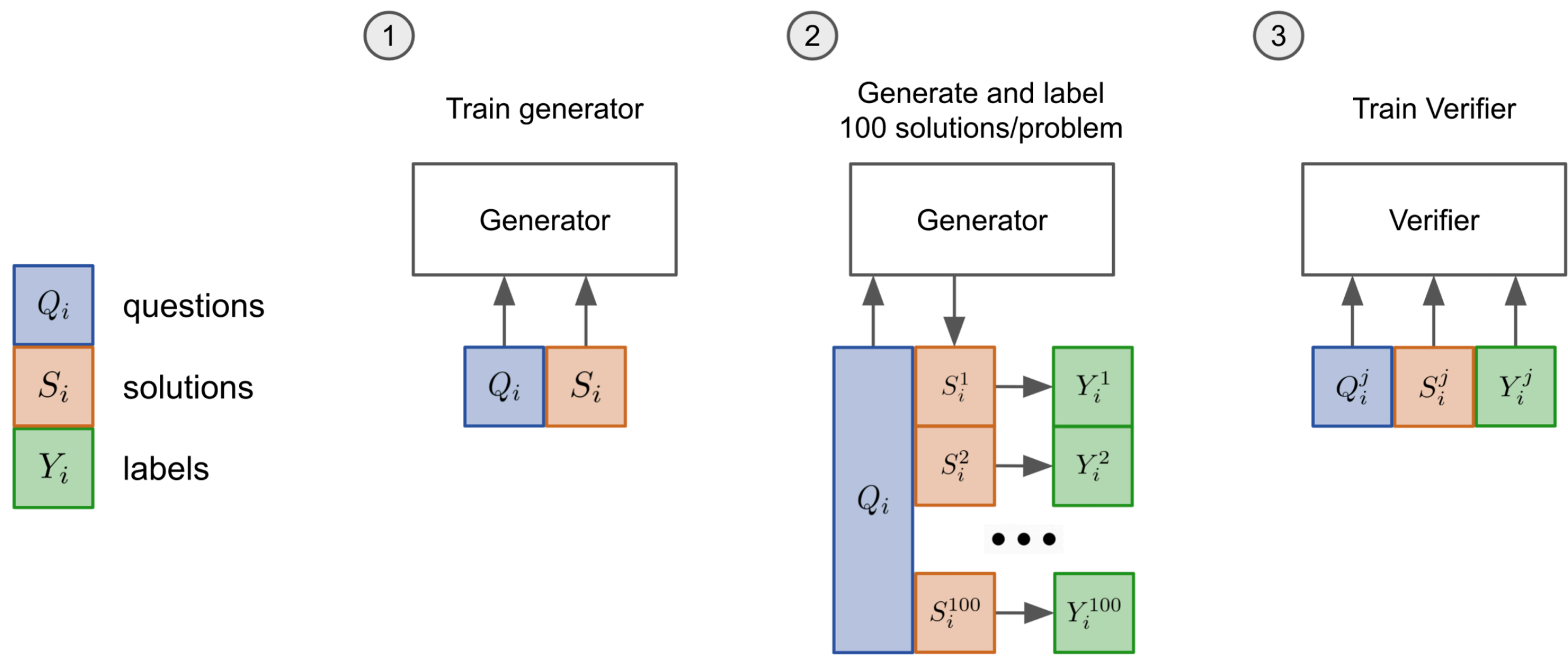


Figure 4: A diagram of the verification training pipeline.

[OpenAI. Training Verifiers to Solve Math Word Problems, 2021]

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $\frac{2}{5}$, what is the numerator of the fraction? (Answer: 14)

Let's call the numerator x.

So the denominator is $3x - 7$.

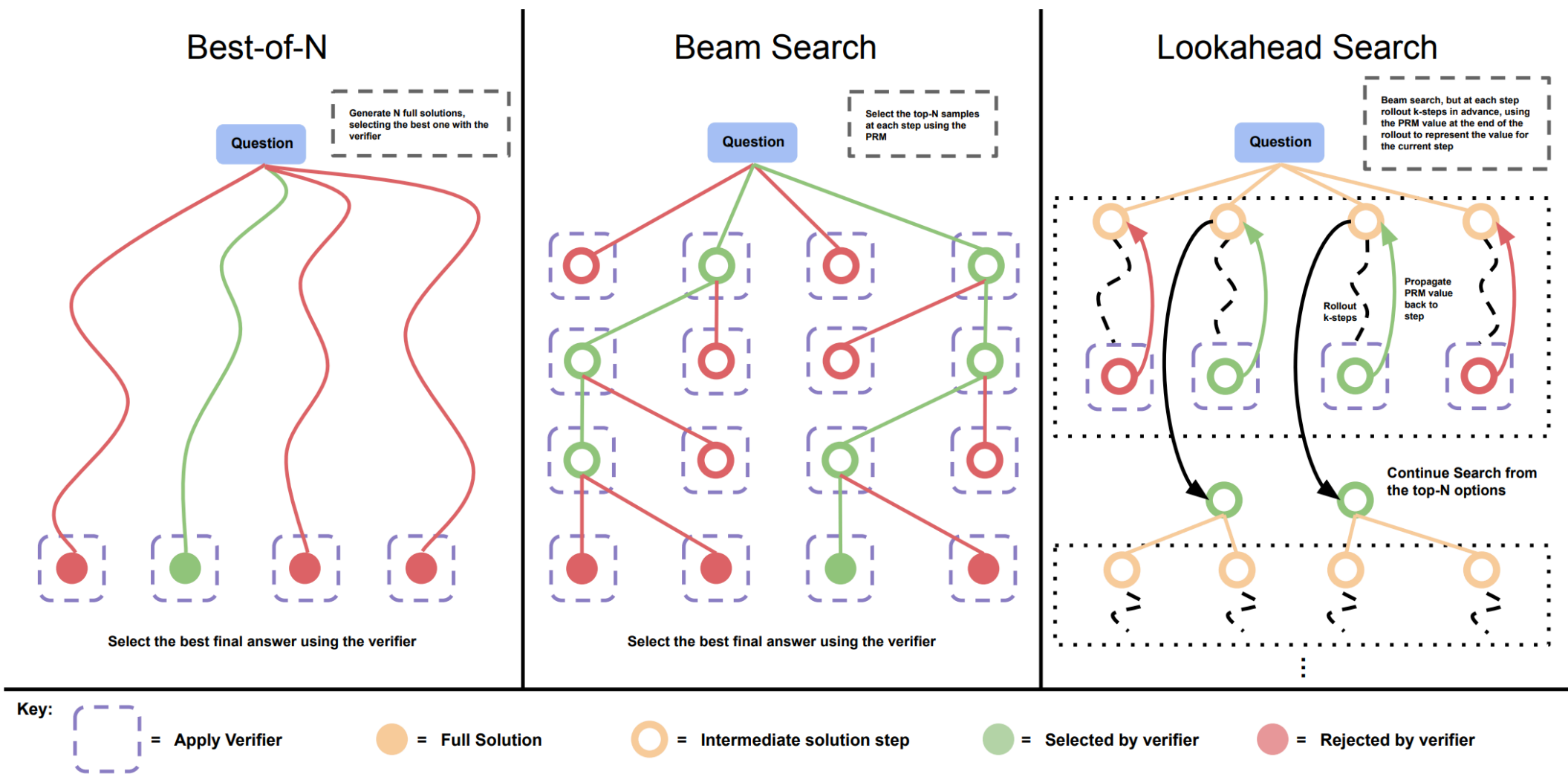
We know that $\frac{x}{3x - 7} = \frac{2}{5}$.

So $5x = 2(3x - 7)$.

$5x = 6x - 14$.

So $x = 7$.

[OpenAI. Let's verify step by step. 2023]

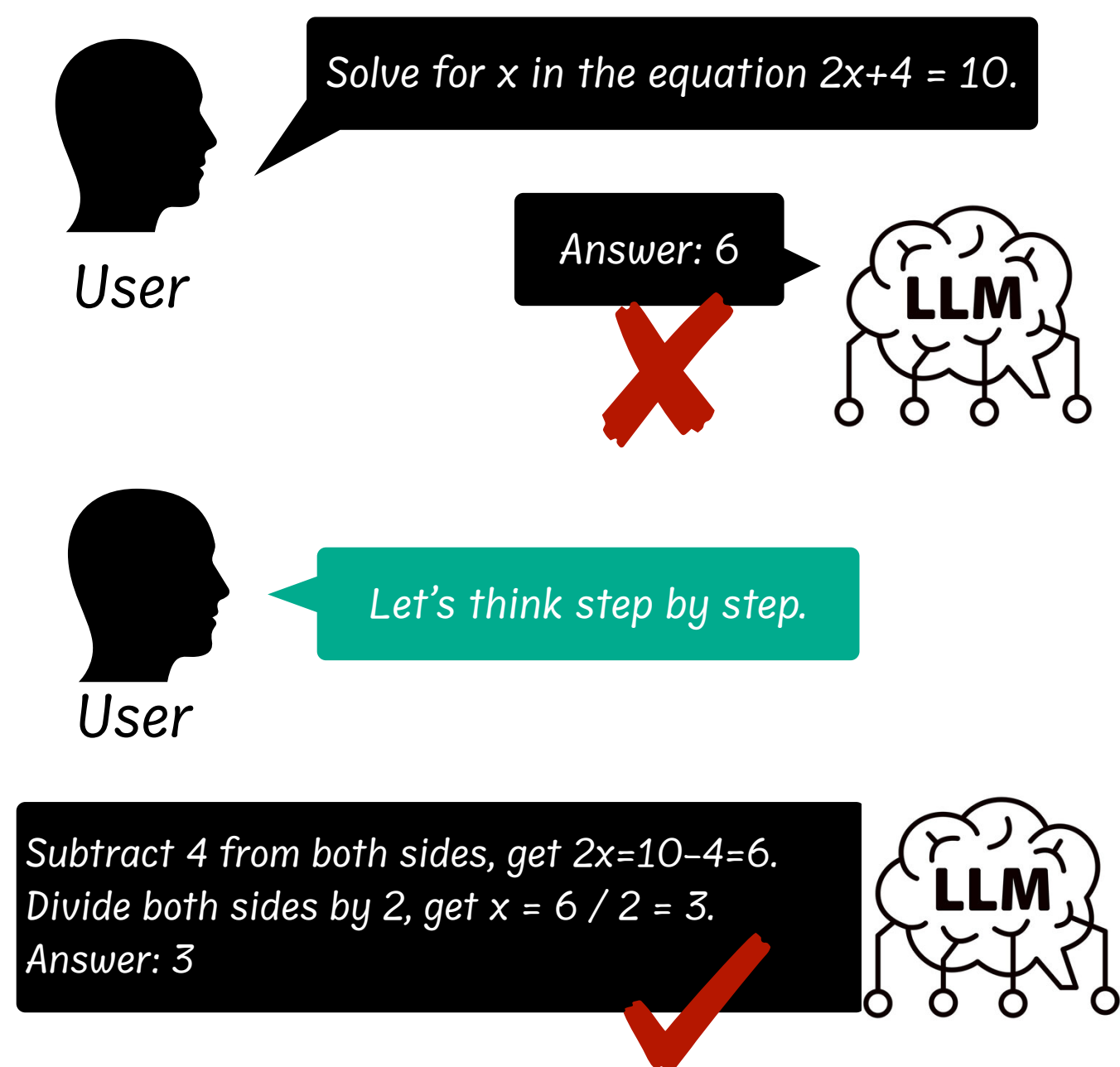


[Google Deepmind. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. 2024]

Main Idea: Chain-of-Thought (CoT) with RL

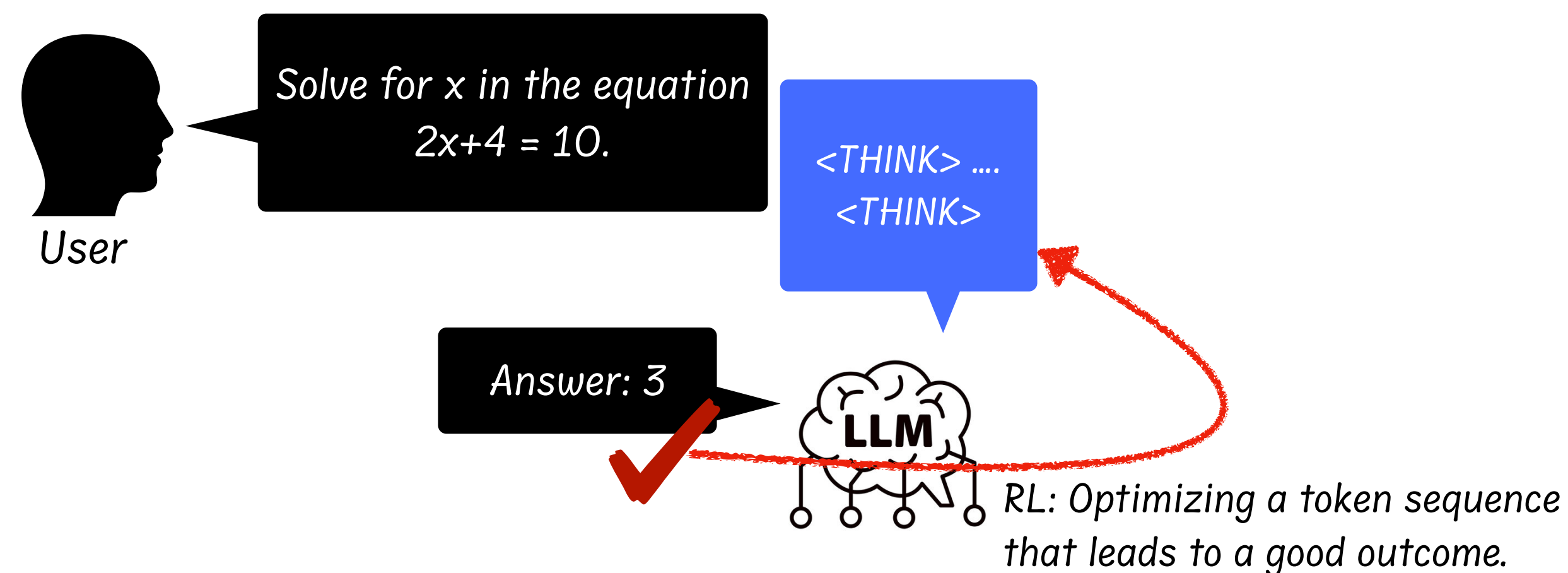
CoT Prompting:

Ask LLMs to “**think**” before answer.



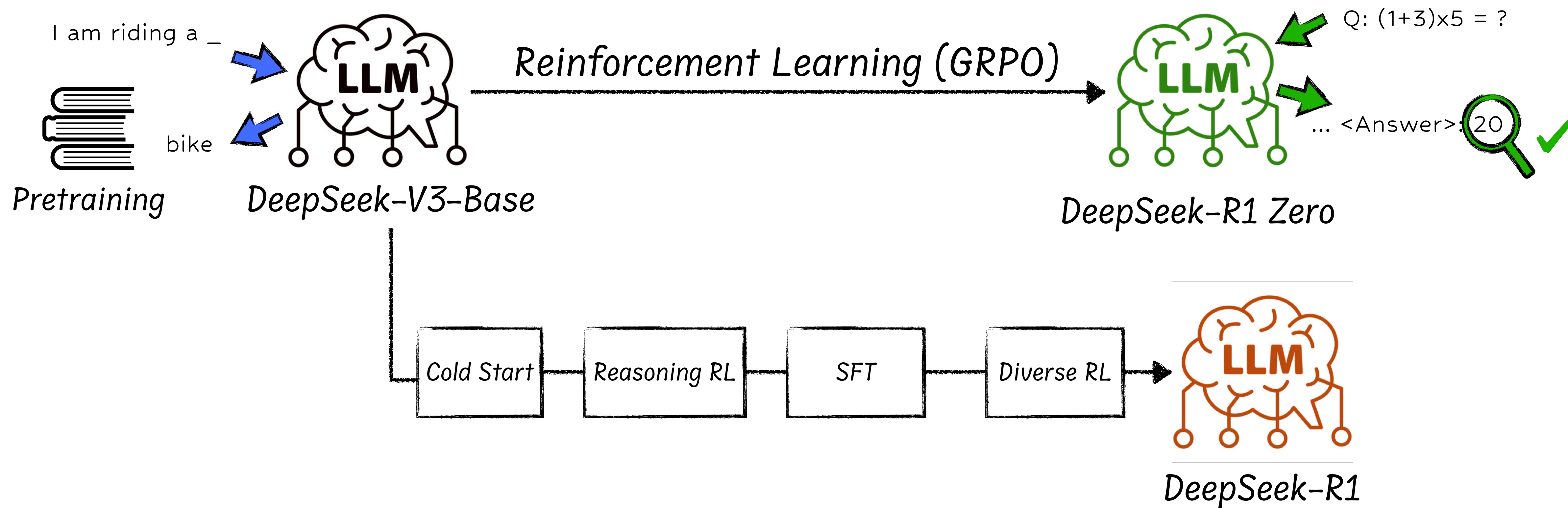
CoT trained with RL:

Teach LLMs how to “**think**” to get correct answers.

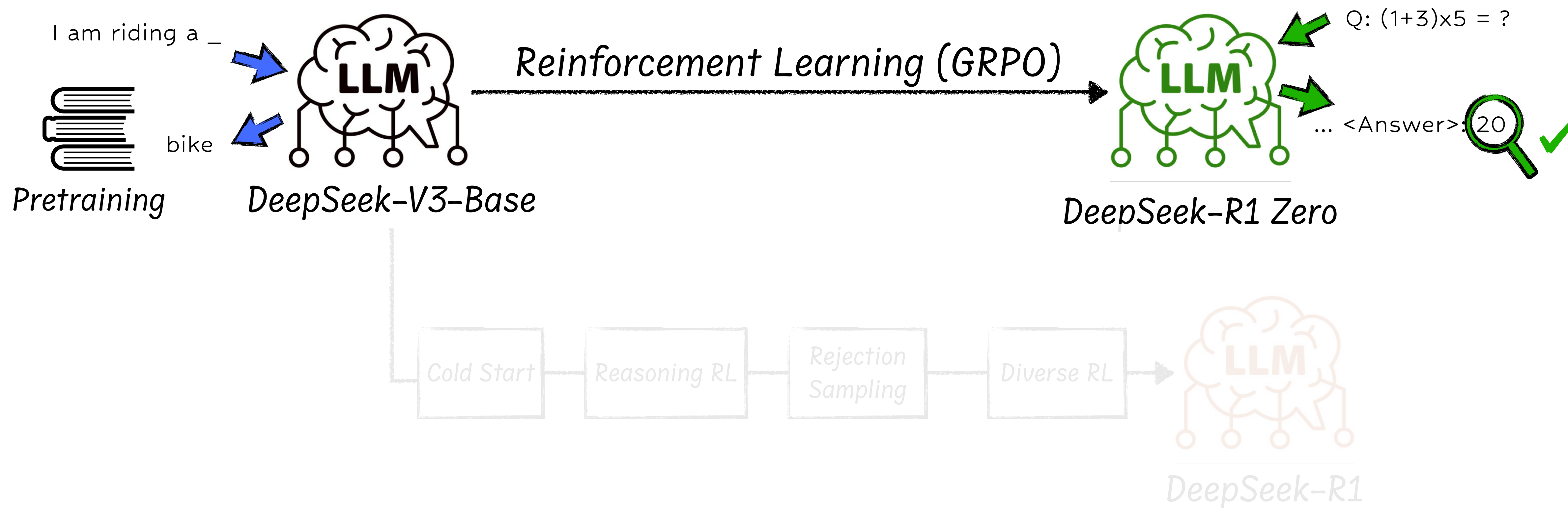


During this process, **chain-of-thought ability emerges**.

Method Overview

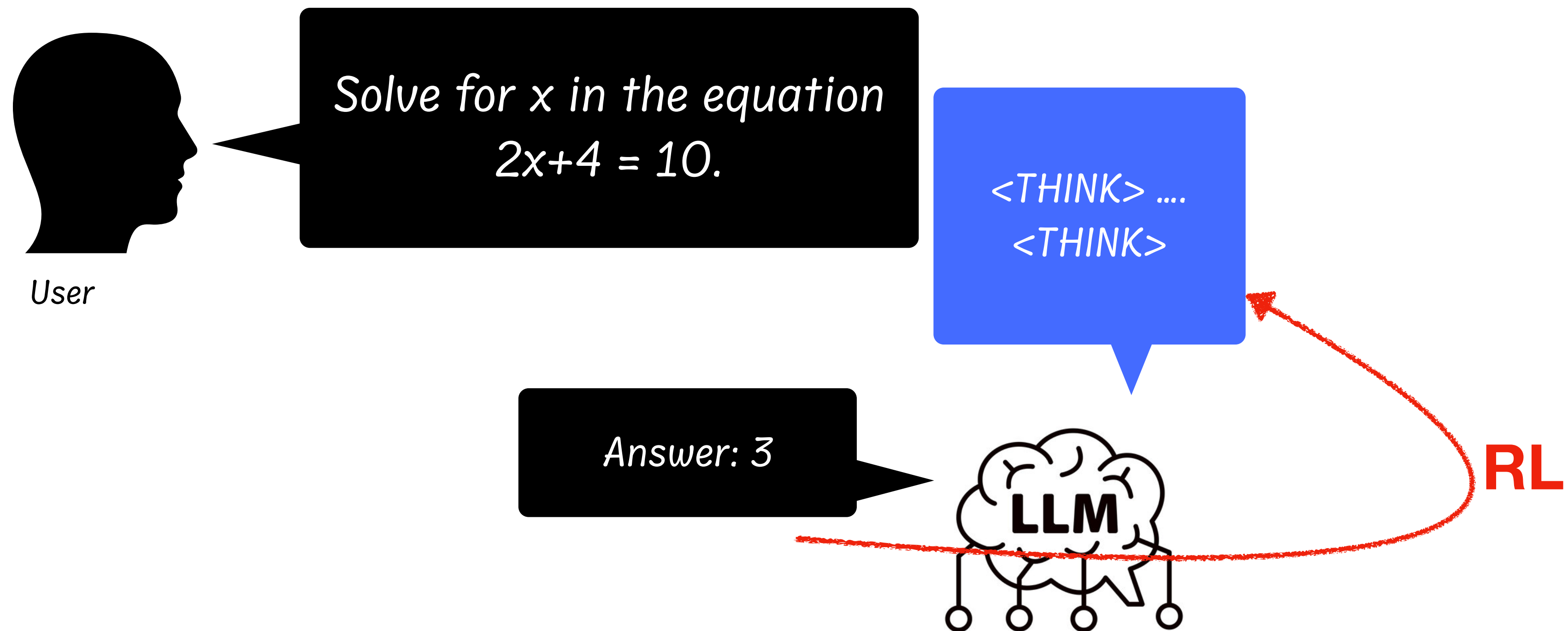


Method Overview



Method: Reinforcement Learning

- Goal: Optimizing a token sequence that leads to a good outcome.



Method: Reinforcement Learning

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

Zhihong Shao^{1,2*†}, Peiyi Wang^{1,3*†}, Qihao Zhu^{1,3*†}, Runxin Xu¹, Junxiao Song¹
Xiao Bi¹, Haowei Zhang¹, Mingchuan Zhang¹, Y.K. Li¹, Y. Wu¹, Daya Guo^{1*}

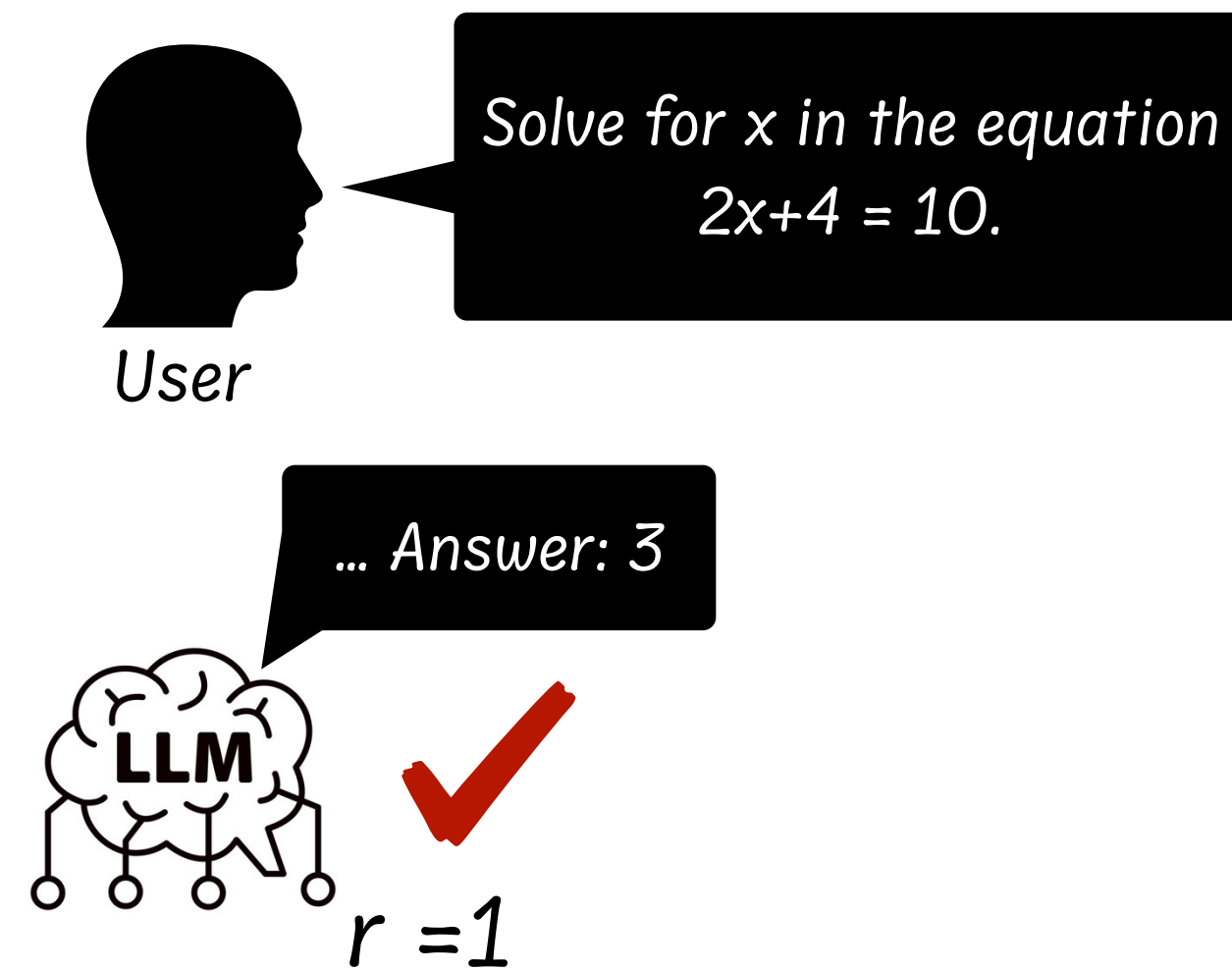
¹DeepSeek-AI, ²Tsinghua University, ³Peking University

{zhihongshao, wangpeiyi, zhuqh, guoday}@deepseek.com
<https://github.com/deepseek-ai/DeepSeek-Math>

(2024)

Method: Reinforcement Learning

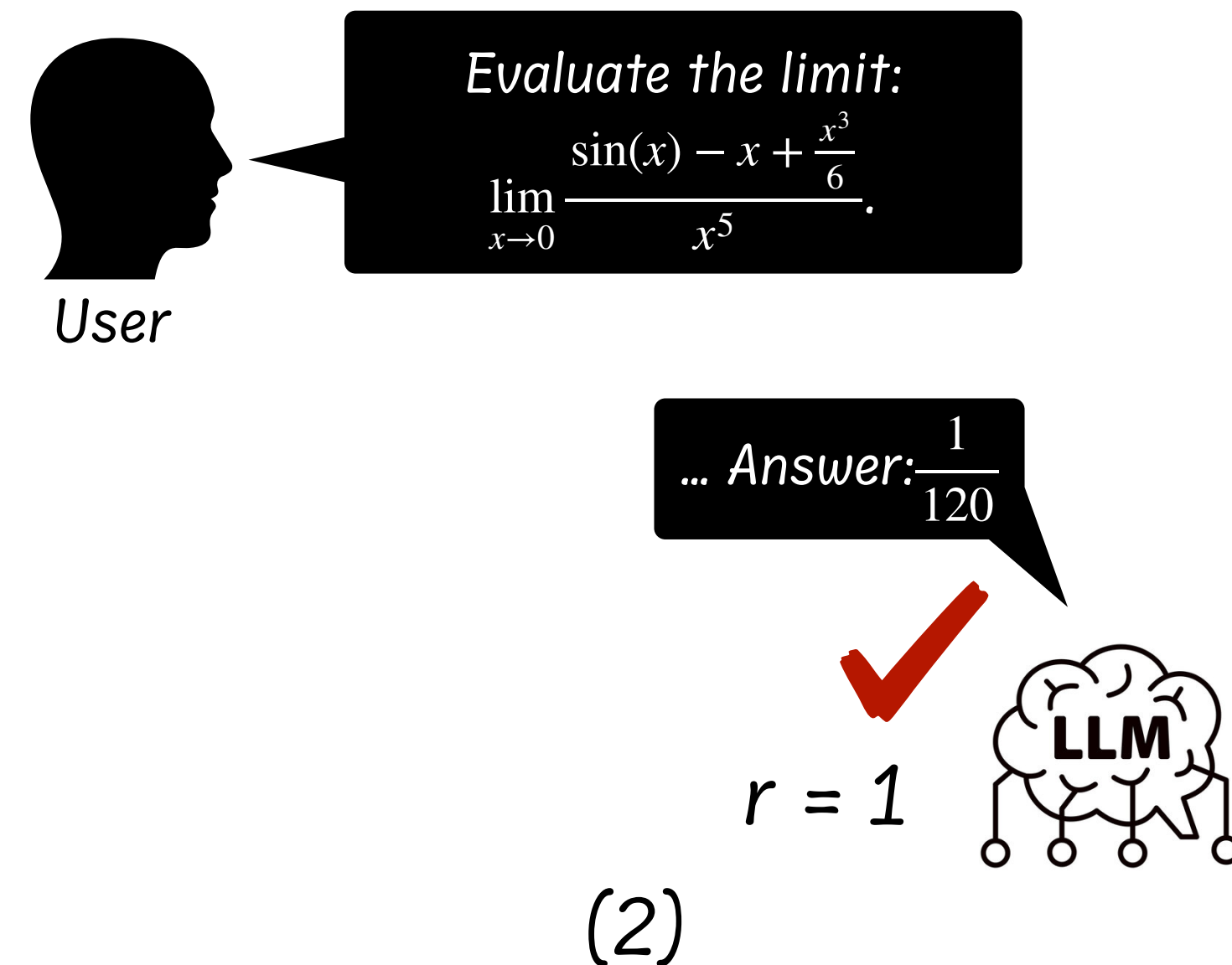
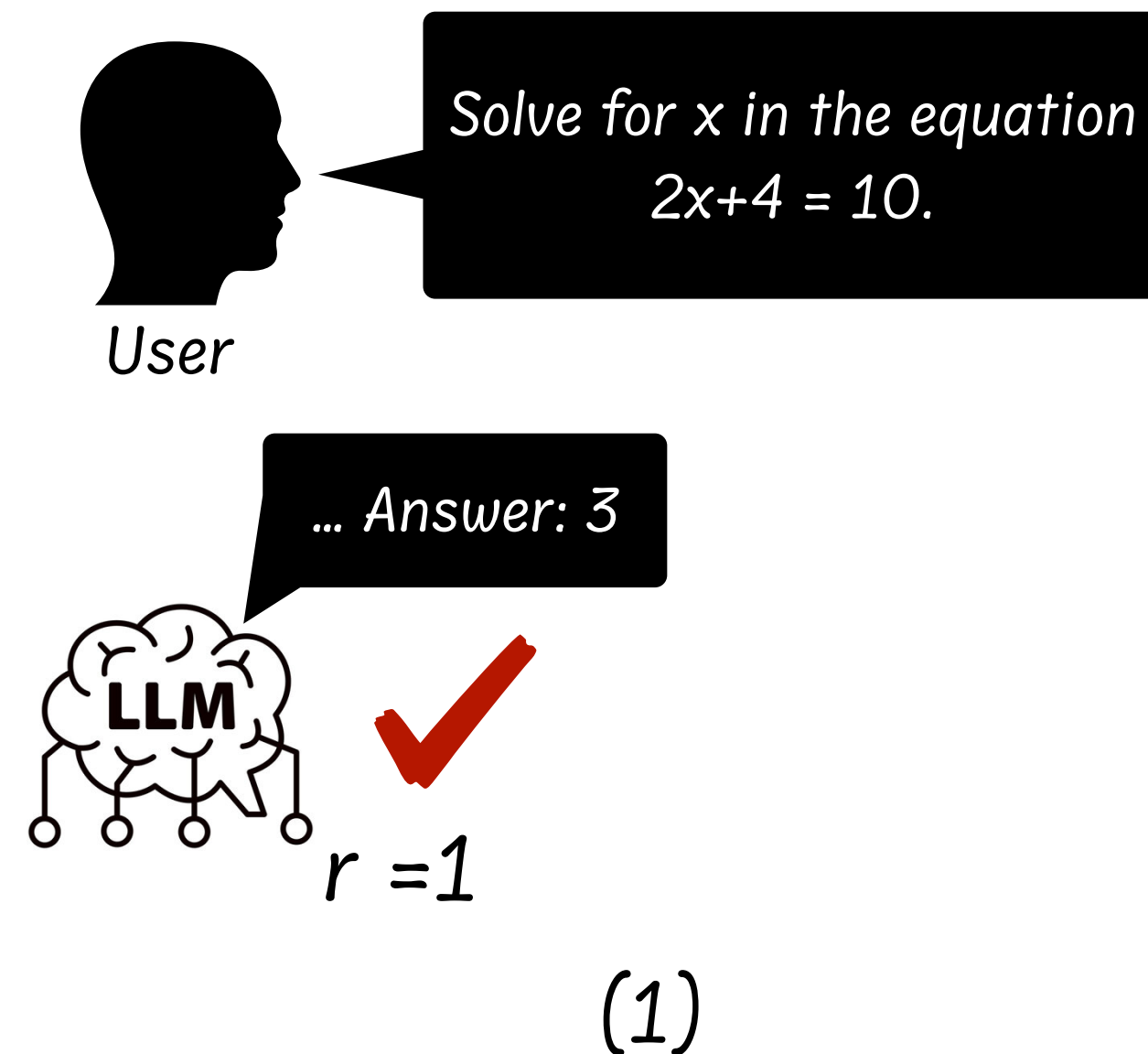
- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



(1)

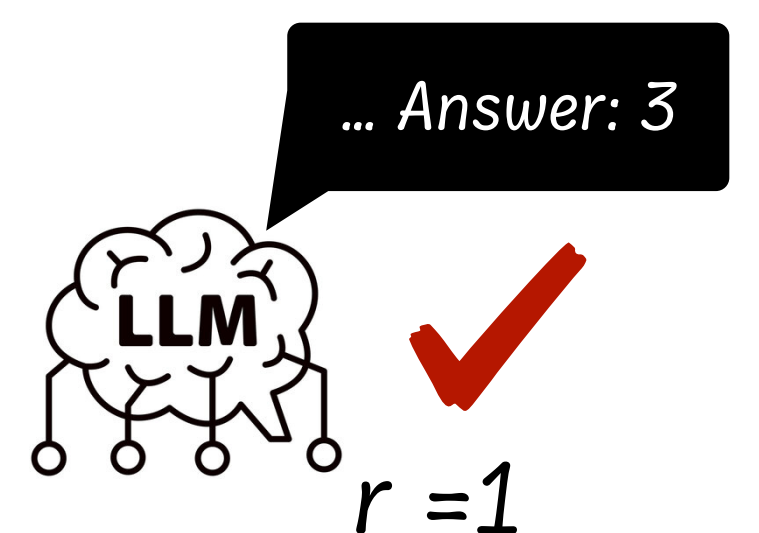
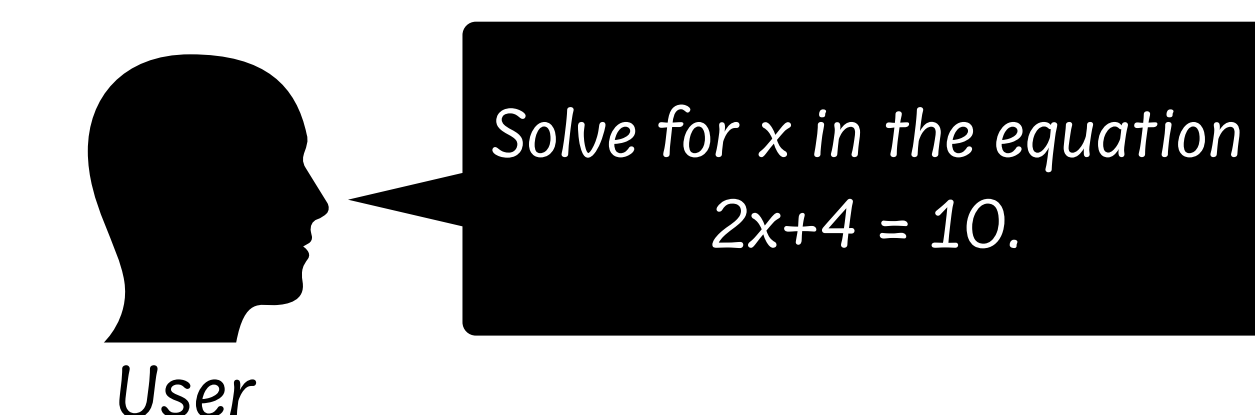
Method: Reinforcement Learning

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.

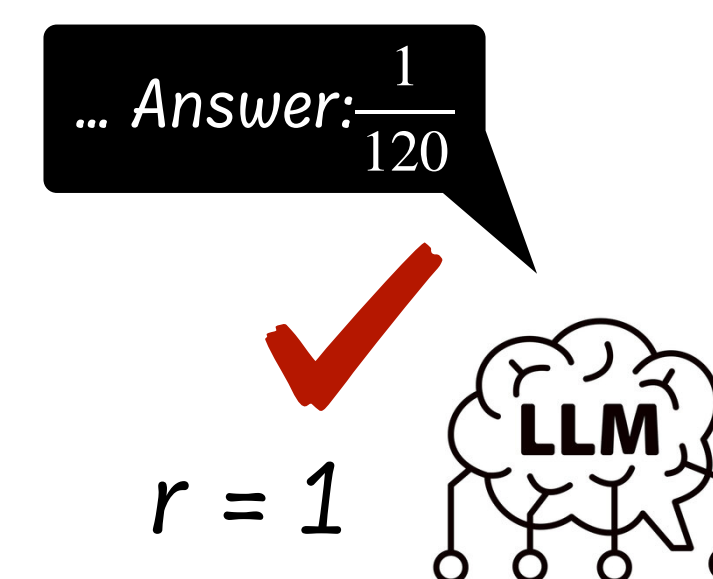
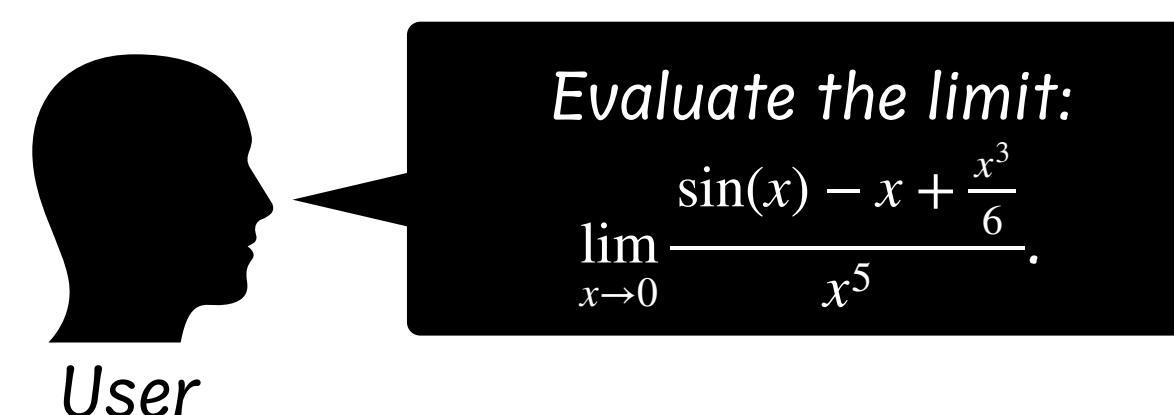


Method: Reinforcement Learning

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



(1)

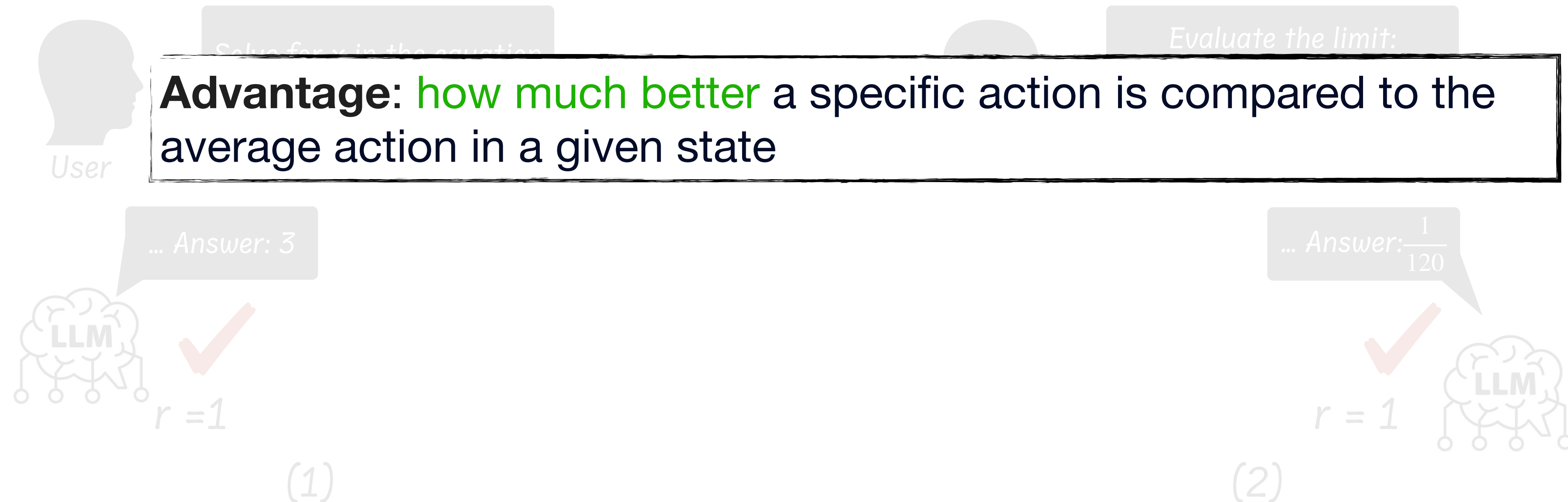


(2)

Problem: raw reward is too sparse!

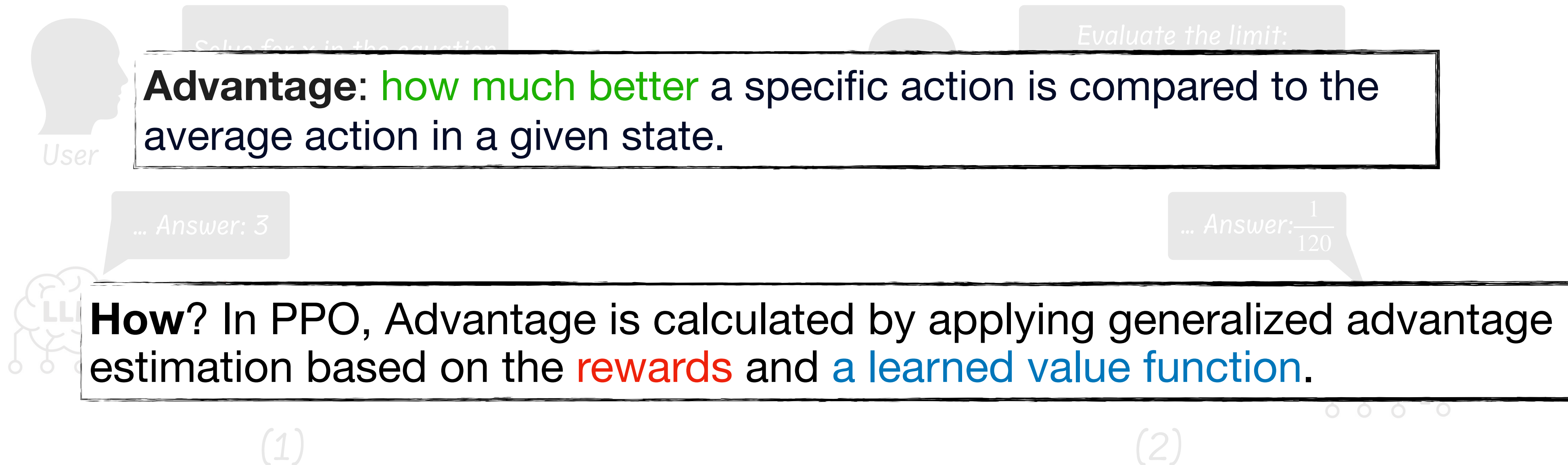
Method: Reinforcement Learning

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



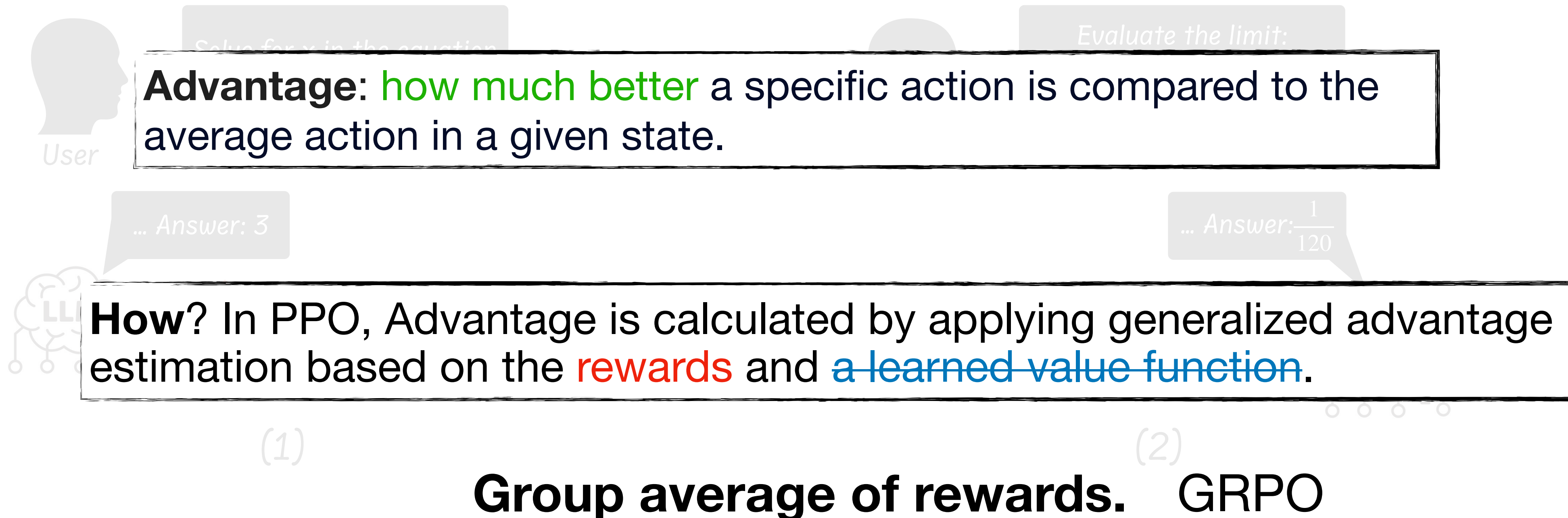
Method: Reinforcement Learning

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



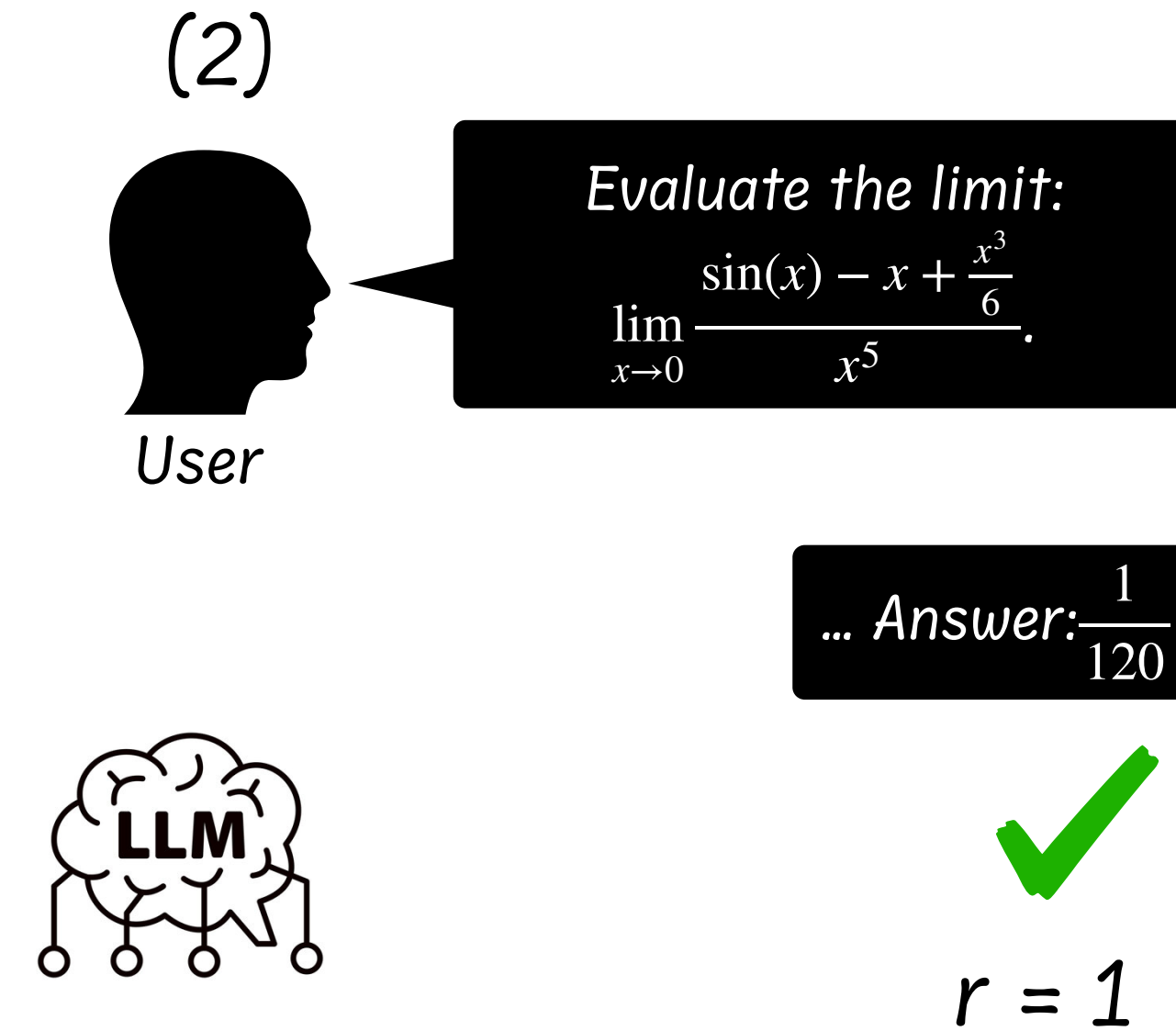
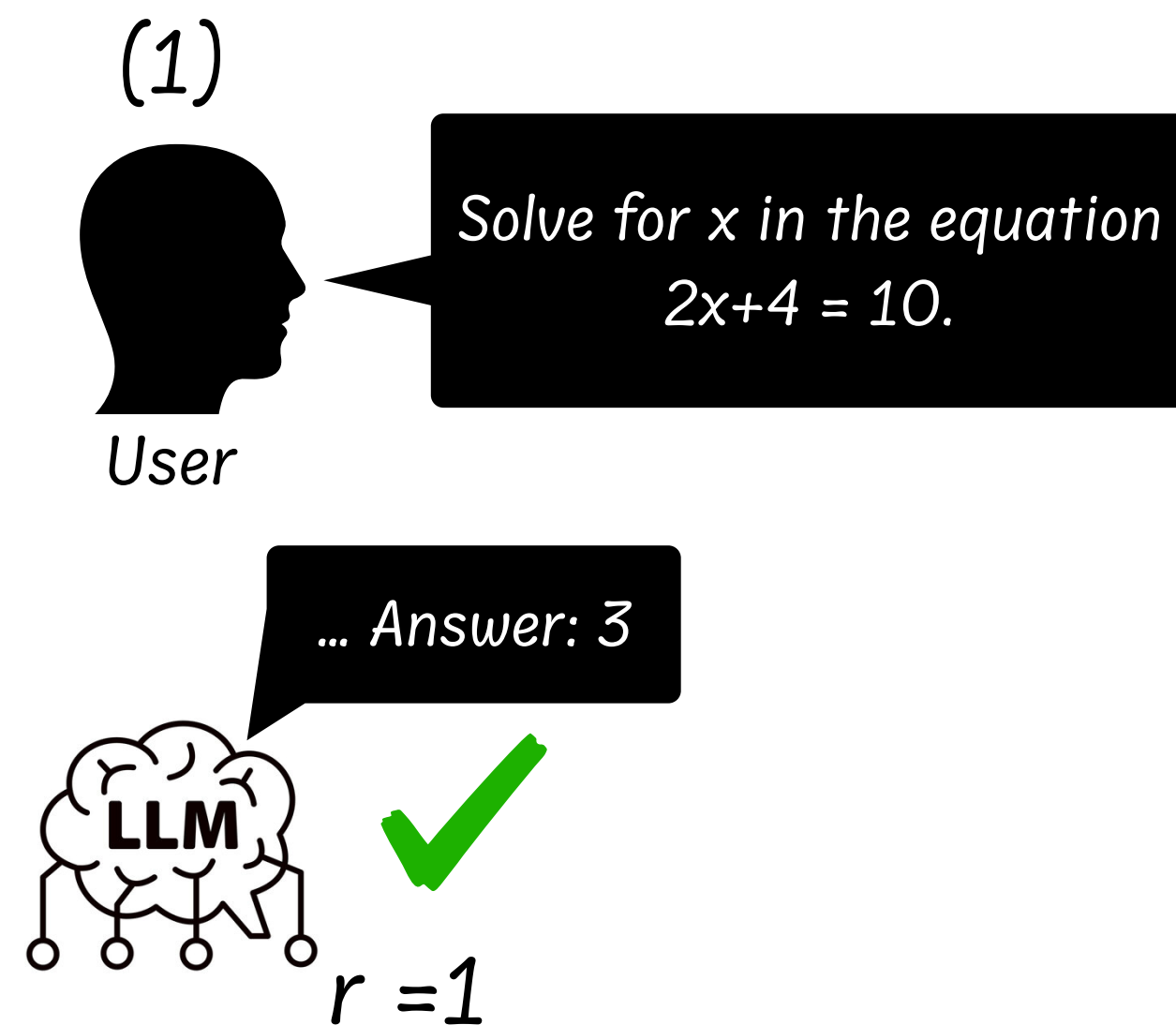
Method: Reinforcement Learning

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



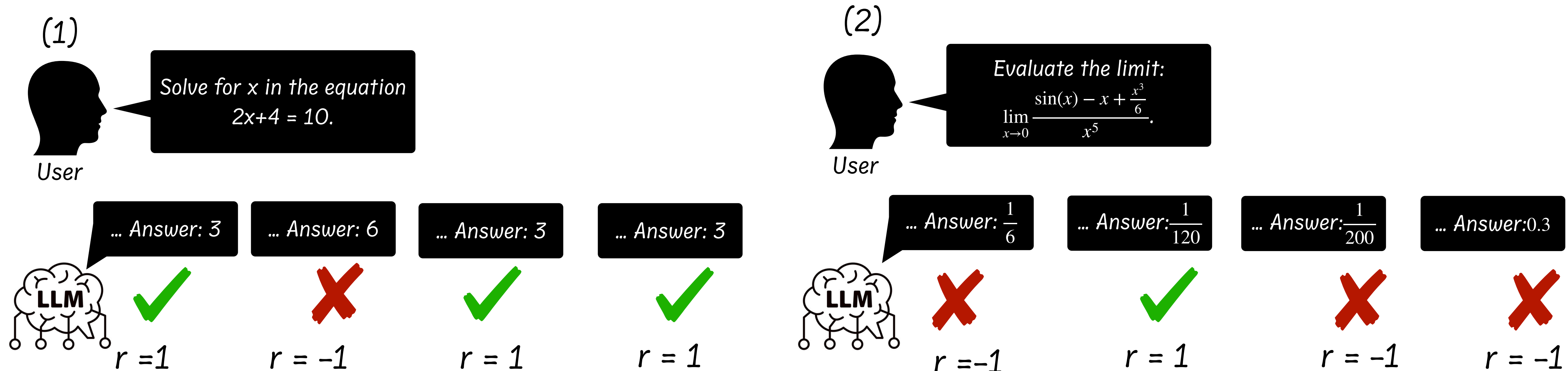
Method: The GRPO Algorithm

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



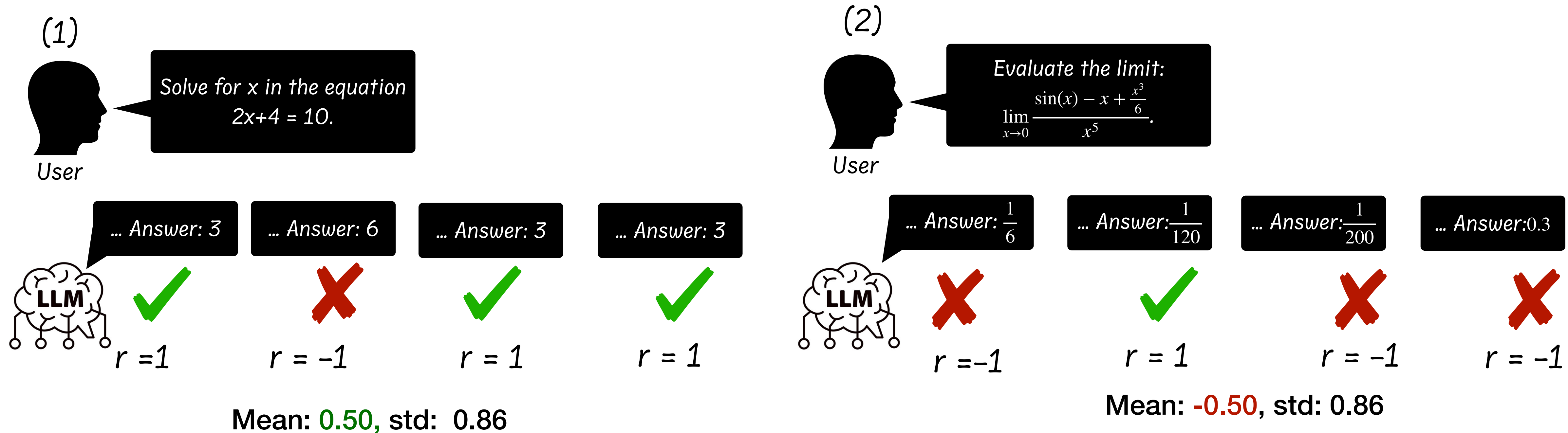
Method: The GRPO Algorithm

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



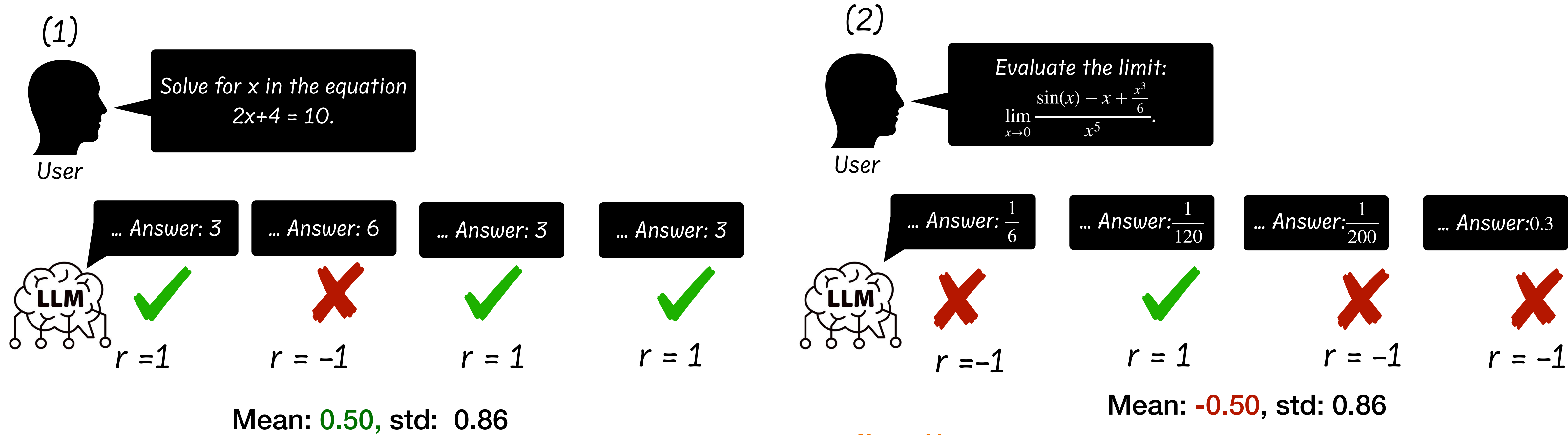
Method: The GRPO Algorithm

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



Method: The GRPO Algorithm

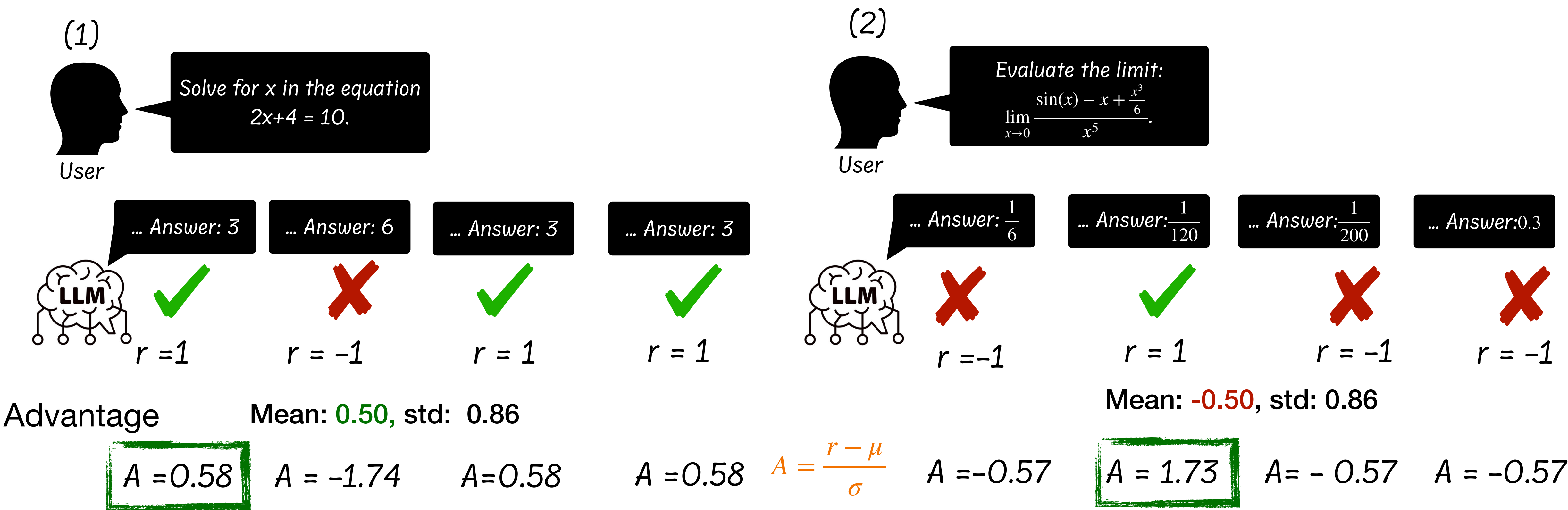
- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.



Advantage $A = \frac{r - \mu}{\sigma}$

Method: The GRPO Algorithm

- Goal: Optimizing a token sequence that leads to a good outcome.
- GRPO: Group Relative Policy Optimization.

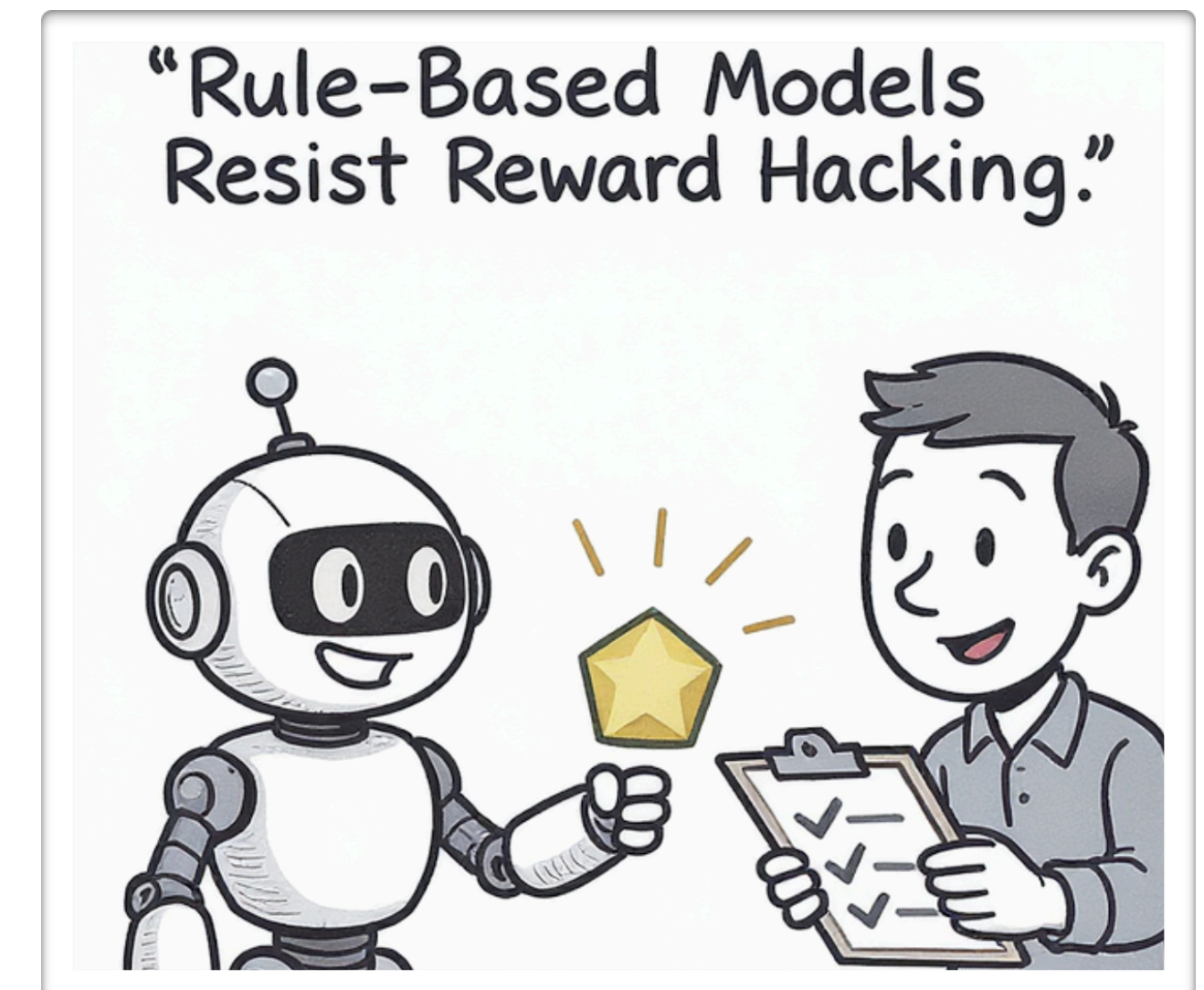


Method: Rule-based Reward Modeling

- **Accuracy:**
 - Math Problem -> Validate the correctness of final answer
 - Code Problem -> Run a compiler to generate feedback
- **Format:** Validate Output Format.
 - Thinking process should be between '`<think>`' and '`</think>`'.

Training template:

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags, respectively, i.e., `<think> reasoning process here </think>` `<answer> answer here </answer>`. User: **prompt**. Assistant:



Generated by FLUX

Rule-based RL training is simpler and cheaper, and so easier to run at large scale.

Results for DeepSeek R1 Zero

- Comparison of DeepSeek-R1-Zero and OpenAI o1 models on reasoning-related benchmarks.

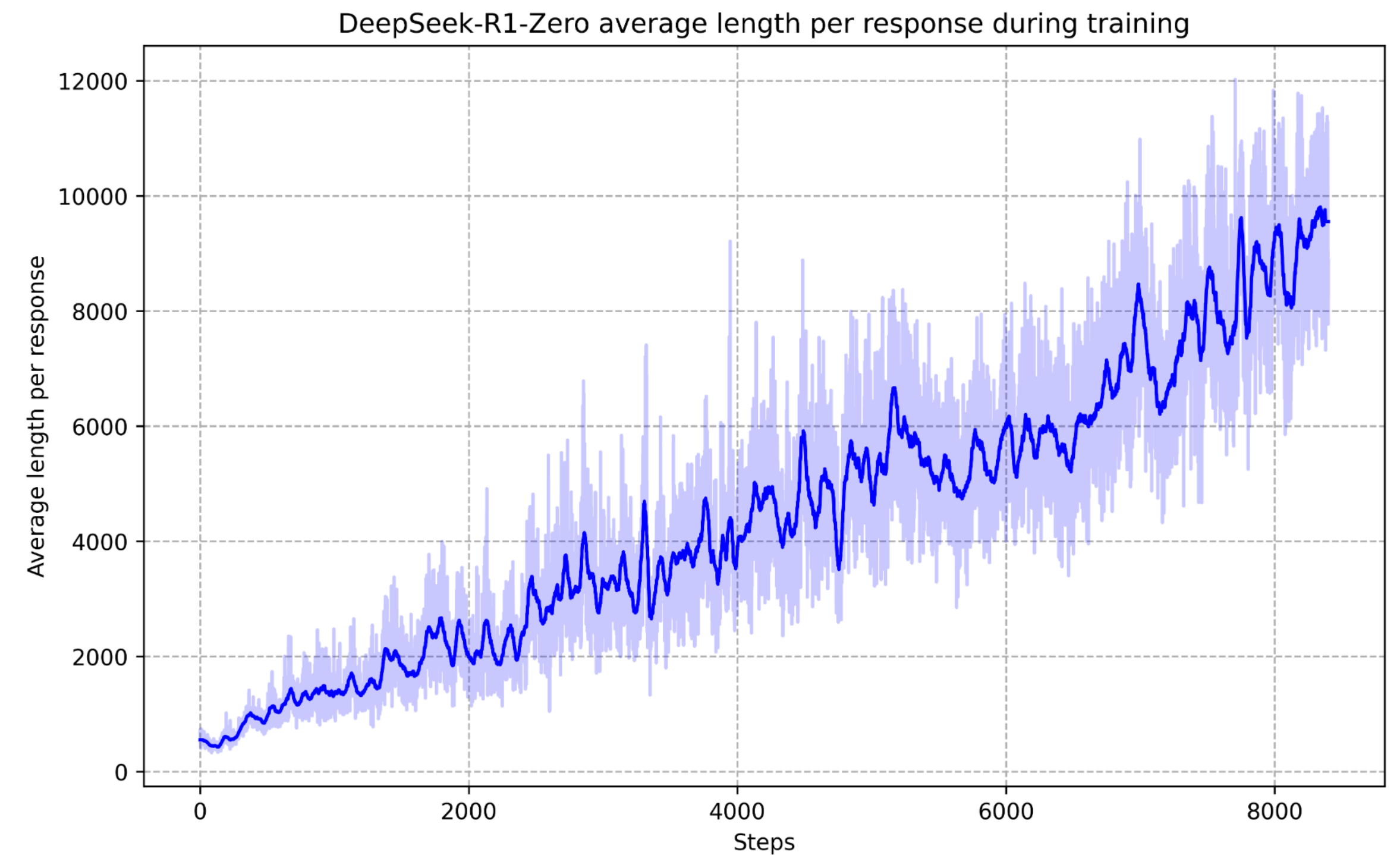
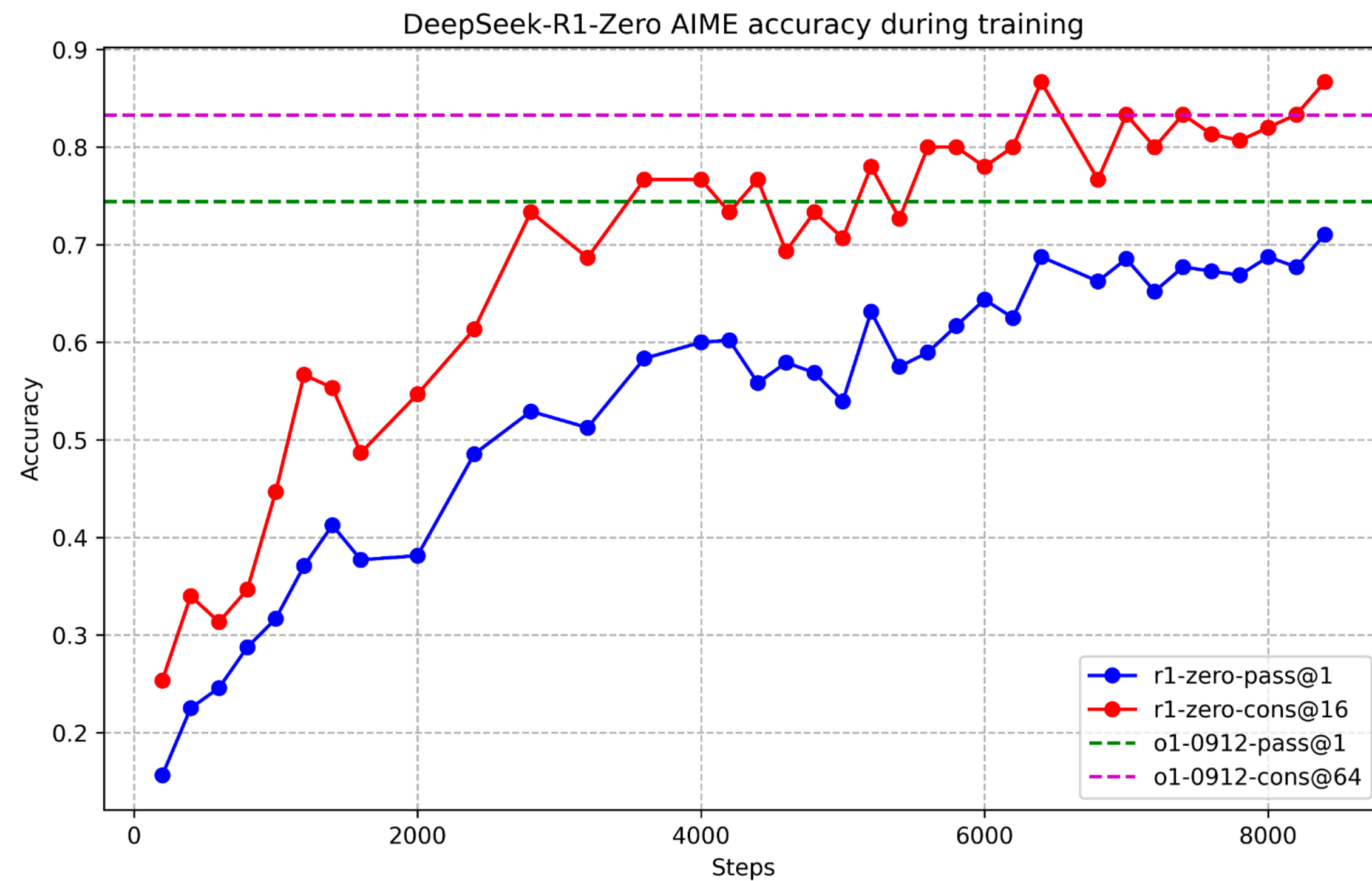
Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820
OpenAI-o1-0912	74.4	83.3	94.8	77.3	63.4	1843
DeepSeek-R1-Zero	71.0	86.7	95.9	73.3	50.0	1444

DeepSeek-R1-Zero reach comparable performance to OpenAI's o1 model.

Pass @ 1: Average correctness per single attempt.
Cons @ 64: Accuracy of the majority vote across 64 attempts.

Results for DeepSeek R1 Zero

- Self-evolution process on AIME dataset



DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time!

Results for DeepSeek R1 Zero

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both ...

$$\left(\sqrt{a - \sqrt{a + x}}\right)^2 = x^2 \implies a - \sqrt{a + x} = x^2.$$

Rearrange to isolate the inner square root term:

$$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$$

...

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be ...

We started with the equation:

$$\sqrt{a - \sqrt{a + x}} = x$$

First, let's square both sides:

$$a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$$

Next, I could square both sides again, treating the equation: ...

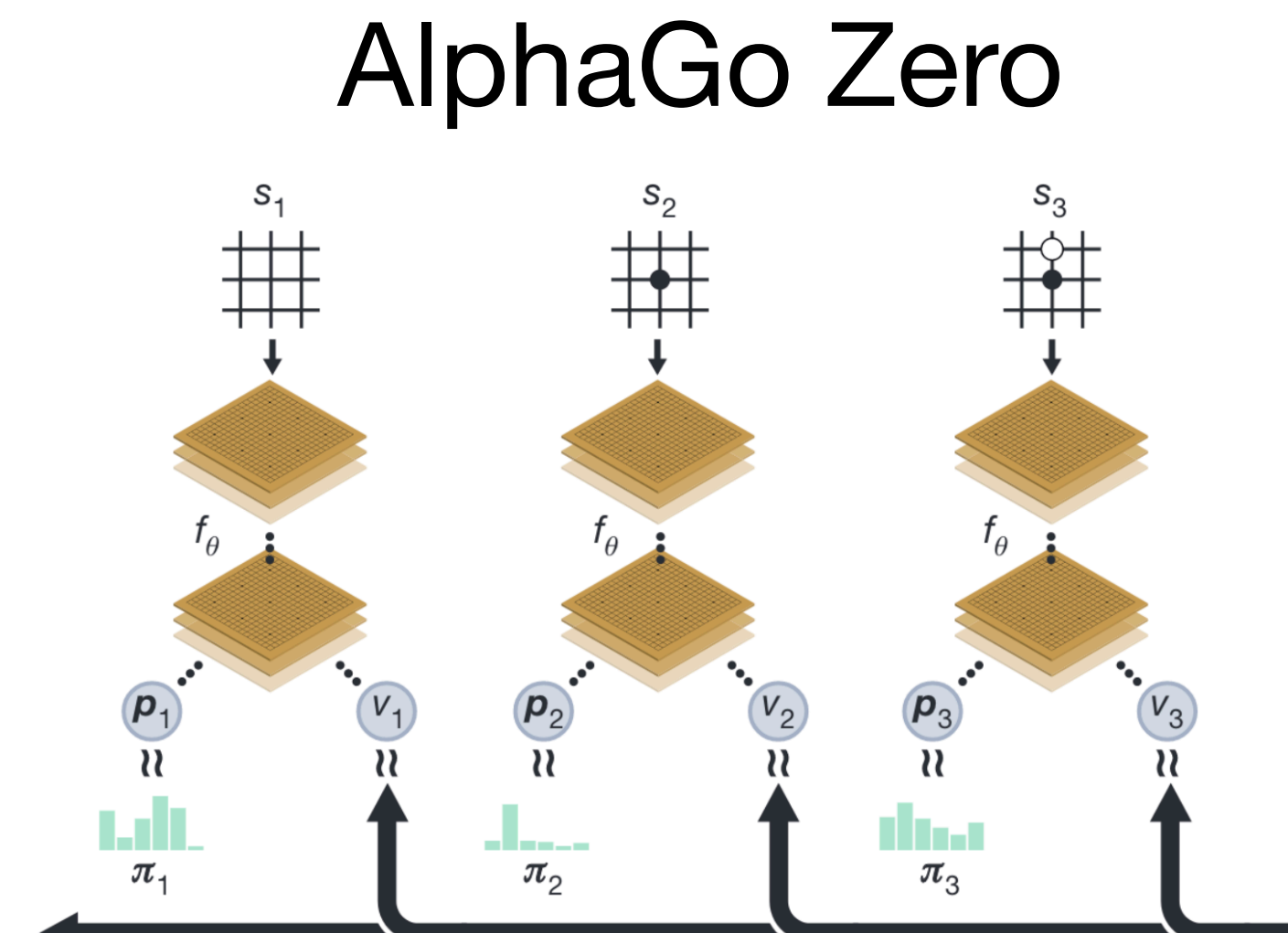
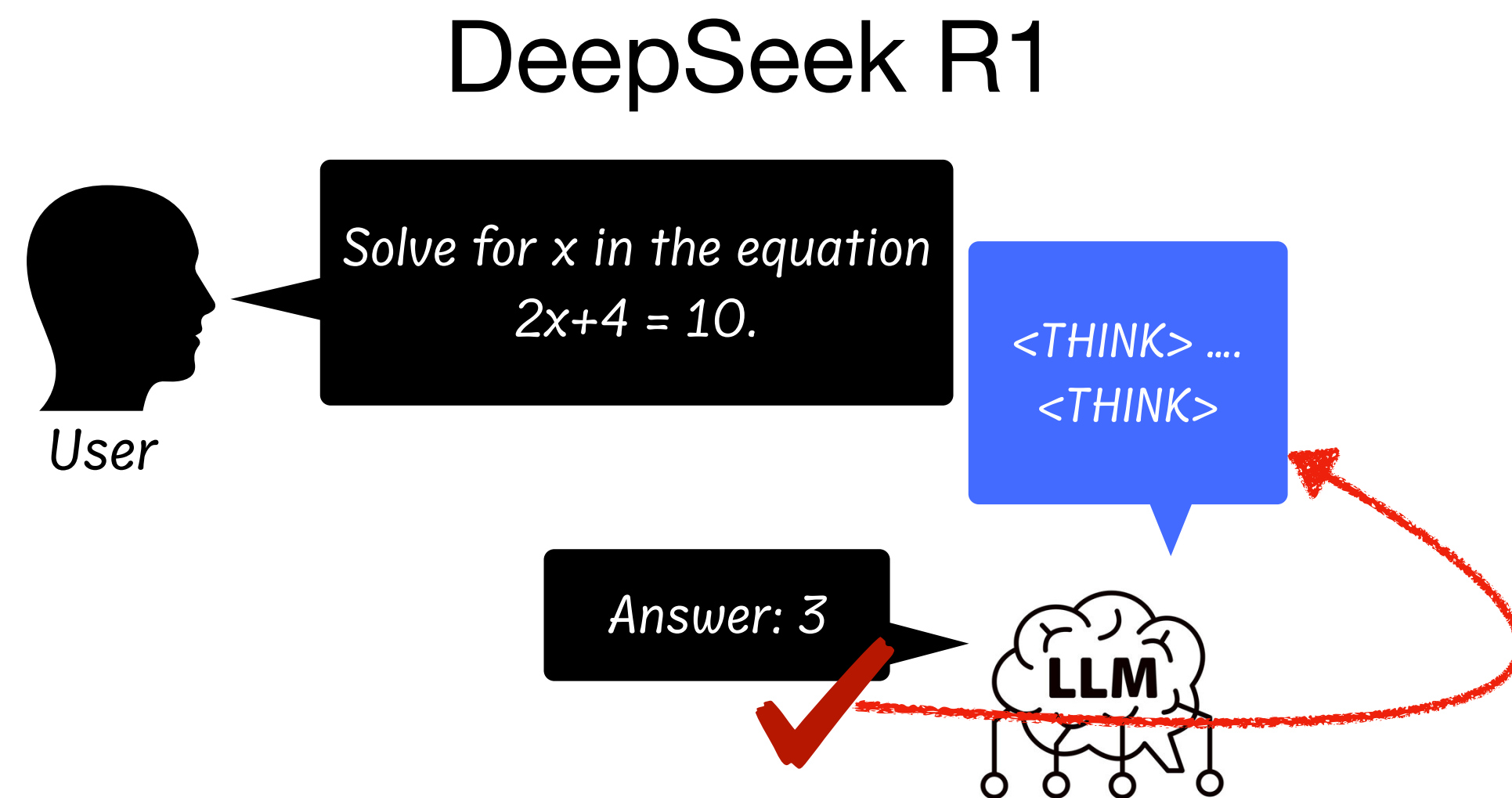
...

“Aha Moment”: DeepSeek-R1-Zero learns to allocate more thinking time to a problem by reevaluating its initial approach.

Discussion

Connection between **AlphaGo Zero** and **DeepSeek R1 Zero**

1. Both names contain “Zero”.
2. Both involve RL and search.
- 3. Learning to search.**



Takeaways



Chain-of-Thought (CoT) is search and planning in LLM



- **CoT may not be directly**
- **Overthinking!**

solve 1+1 = ?



Thought for 19 seconds ^

First, the user asked: "solve 1+1 = ?". This is a very basic arithmetic question. I know that $1 + 1$ equals 2. It's one of the simplest math facts.

As an AI, I should respond accurately and helpfully. Since this is straightforward, I can answer directly without much explanation. But to be engaging, I might add a bit of context or a friendly tone.

Possible responses:

- Just say "2".
- Say "The answer is 2."
- Or, to make it more fun: " $1 + 1$ equals 2!"

I should consider if there's any trick here. Is this a test for something deeper? Like in binary or other bases? But the user didn't specify any base, so it's safe to assume decimal (base 10).

In binary, $1 + 1$ is 10, but that's not typically what people mean in this context. The question is written in standard numerical form, so decimal is implied.

Also, in some philosophical contexts, like in set theory, $1 + 1 = 2$ is a fundamental concept, but that might be overcomplicating it.

The user said "solve", which might imply they want a step-by-step solution, but for something this simple, it's unnecessary. I can just state the answer.

To be thorough, I could say: "1 plus 1 is equal to 2."

Finally, since I'm an AI, I should ensure my response is clear and concise. No need for markdown or anything fancy; plain text is fine.

Response: " $1 + 1 = 2$ "

Or to match the query: "The solution to $1 + 1$ is 2."

The solution to $1 + 1$ is 2.

This is a basic arithmetic operation where adding the number 1 to another 1 results in 2. If you have any more questions or need clarification, feel free to ask! 😊



3rd paper

Inference-Time Scaling for Diffusion Models beyond Scaling Denoising Steps

Google DeepMind. 2025

Inference-Time Scaling for Diffusion Models beyond Scaling Denoising Steps

Nanye Ma^{*, 1}, Shangyuan Tong^{*, 2}, Haolin Jia³, Hexiang Hu³, Yu-Chuan Su³, Mingda Zhang³, Xuan Yang³, Yandong Li³, Tommi Jaakkola², Xuhui Jia³ and Saining Xie^{1,3}

^{*}Equal contribution, ¹NYU, ²MIT, ³Google, ^{*}Work done during an internship at Google

Generative models have made significant impacts across various domains, largely due to their ability to scale during training by increasing data, computational resources, and model size, a phenomenon characterized by the scaling laws. Recent research has begun to explore inference-time scaling behavior in Large Language Models (LLMs), revealing how performance can further improve with additional computation during inference. Unlike LLMs, diffusion models inherently possess the flexibility to adjust inference-time computation via the number of denoising steps, although the performance gains typically flatten after a few dozen. In this work, we explore the inference-time scaling behavior of diffusion models *beyond* increasing denoising steps and investigate how the generation performance can further improve with increased computation. Specifically, we consider a search problem aimed at identifying better noises for the diffusion sampling process. We structure the design space along two axes: the verifiers used to provide feedback, and the algorithms used to find better noise candidates. Through extensive experiments on class-conditioned and text-conditioned image generation benchmarks, our findings reveal that increasing inference-time compute leads to substantial improvements in the quality of samples generated by diffusion models, and with the complicated nature of images, combinations of the components in the framework can be specifically chosen to conform with different application scenario.

1. Introduction

Generative models have transformed various fields, including language [1, 78, 80], vision [60, 61], and biology [86], by learning to sample from the underlying data distributions. A key factor in their success is their ability to scale up during training by increasing data volumes, computational resources, and model sizes. This training-time scaling behavior, often described as *Scaling Laws* [27, 31], predicts how performance improves as the models grow larger, consume more data, and are trained for longer time, guiding the development of increasingly capable generative models.

Recently, in Large Language Models (LLMs), the study on scaling has expanded to inference-time [7, 69, 90]. By allocating more compute during inference, often through sophisticated search processes, these works show that LLMs can produce higher-quality and more contextually appropriate responses [20, 74, 85, 87, 95]. Inference-time scaling opens new avenues for improving model performance when additional resources become available after training.

Diffusion models [26, 70, 72], trained to remove noises from data, are a class of generative models that dominates the continuous data domains such as images [15], audio [67], and videos [56]. To generate a single sample, their generation process usually starts from pure noise and requires multiple forward passes of trained models to denoise and obtain clean data. These forward passes are thus dubbed *denoising steps*. Since the number of denoising steps can be adjusted to trade sample quality for computational cost, the generation process of diffusion models naturally provides flexibility in allocating inference-time computation budget. Under the context of generative models, such computation budget is also commonly

Background: Diffusion Models

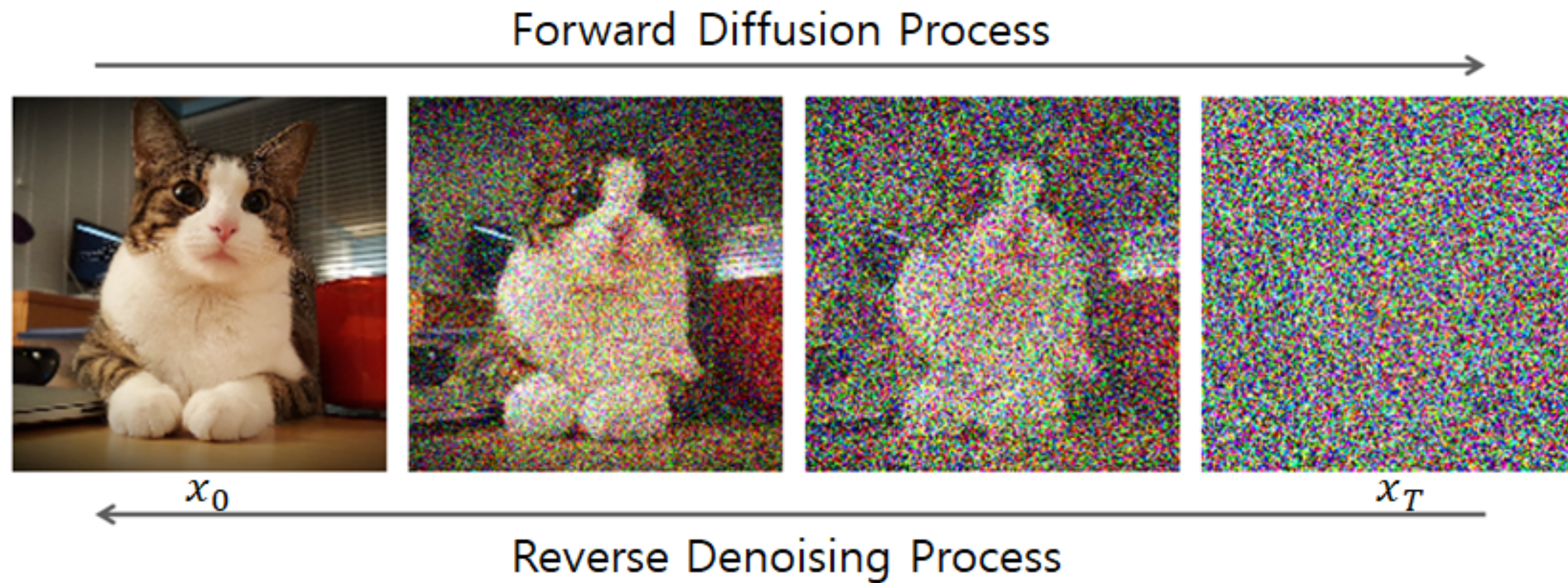


Figure source: <https://sushant-kumar.com/blog/ddpm-denoising-diffusion-probabilistic-models>

Background: Test-time Scaling for Diffusion Models

Inference-time Scaling by increasing denoising steps?

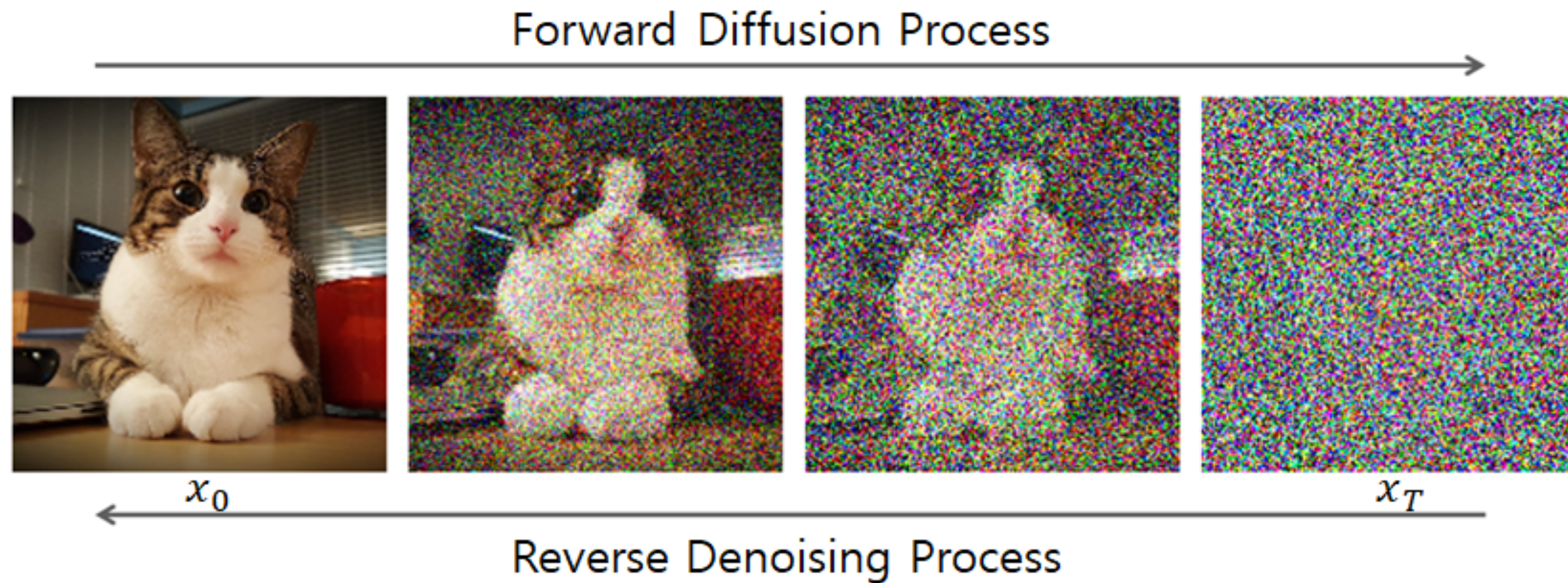
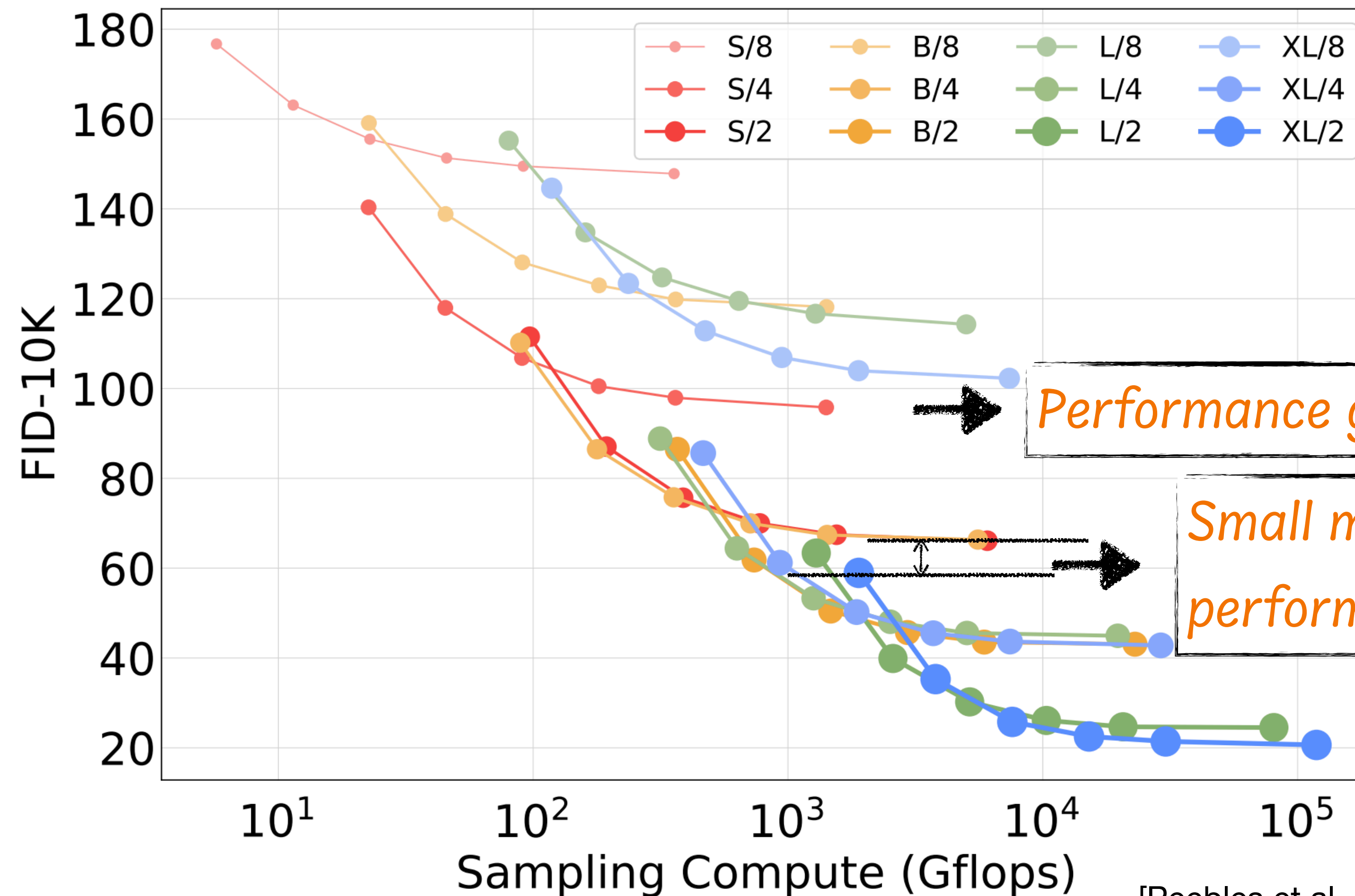


Figure source: <https://sushant-kumar.com/blog/ddpm-denoising-diffusion-probabilistic-models>

Background: Test-time Scaling for Diffusion Models

Inference-time Scaling by increasing denoising steps?



Key Question: How to effectively scale diffusion model during test-time?

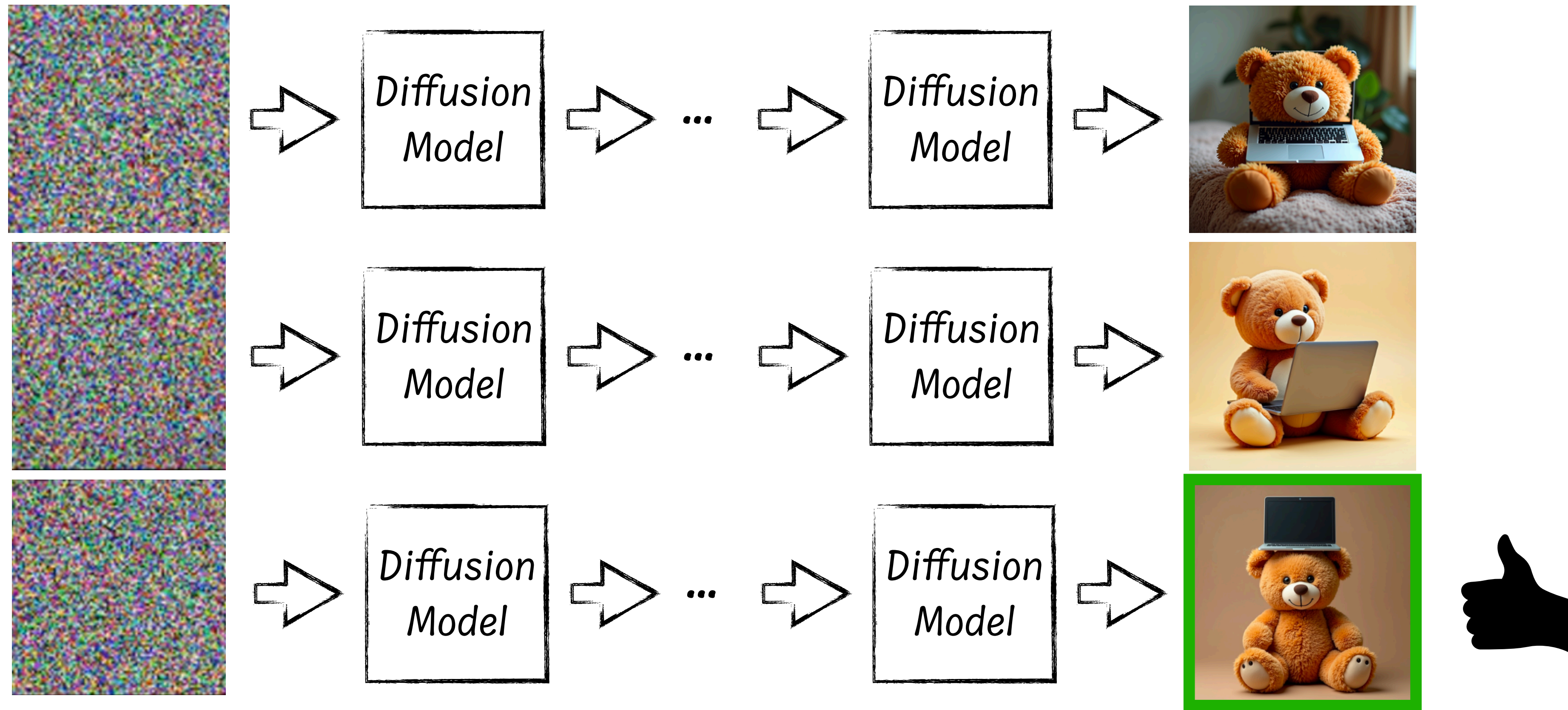
Performance gains tend to plateau.

Small models cannot close the performance gap with our large models.

[Peebles et al. Scalable Diffusion Models with Transformers, 2022.]

Method Overview: Scaling by Search

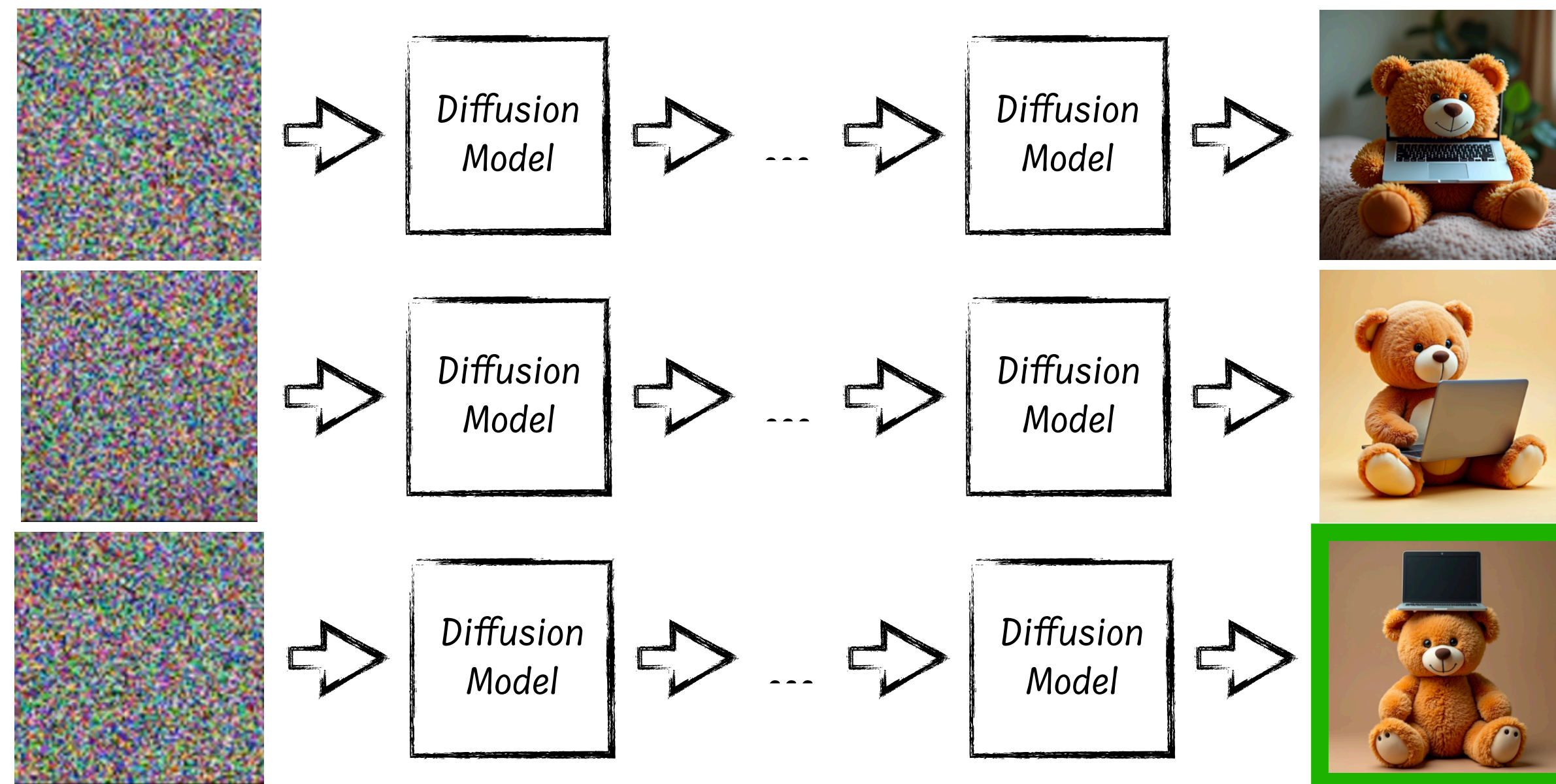
Prompt: A laptop **on top of** a teddy bear



[Images generated by FLUX-1.dev]

Method: Design Axes for Search

Prompt: A laptop **on top of** a teddy bear



1. Verifiers:

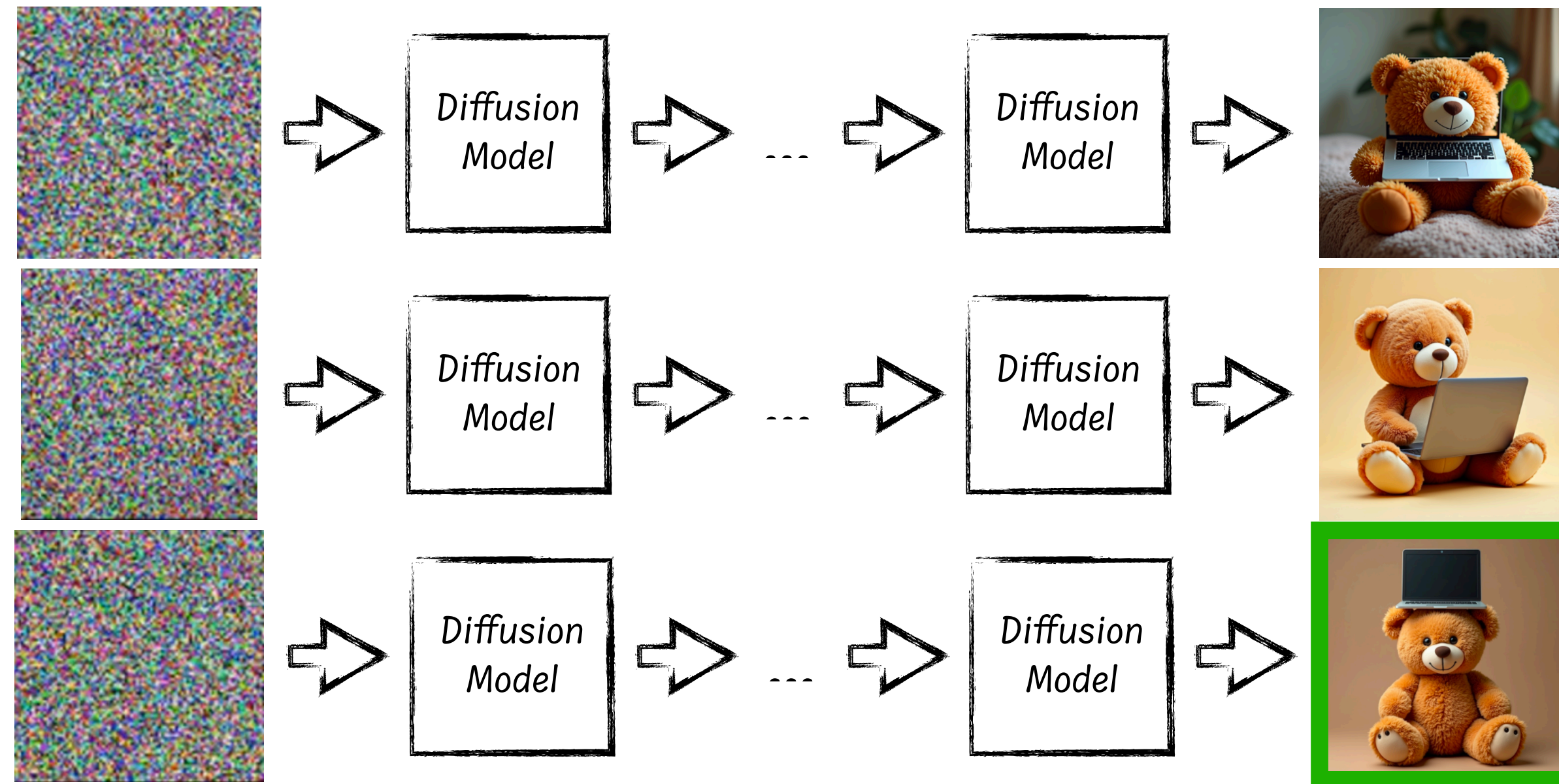
Evaluate the goodness of candidates

2. Search Algorithm:

Find better candidates based on the verifiers scores

Verifiers

Prompt: A laptop **on top of** a teddy bear



1. Verifiers:

Evaluate the goodness of candidates

2. Search Algorithm:

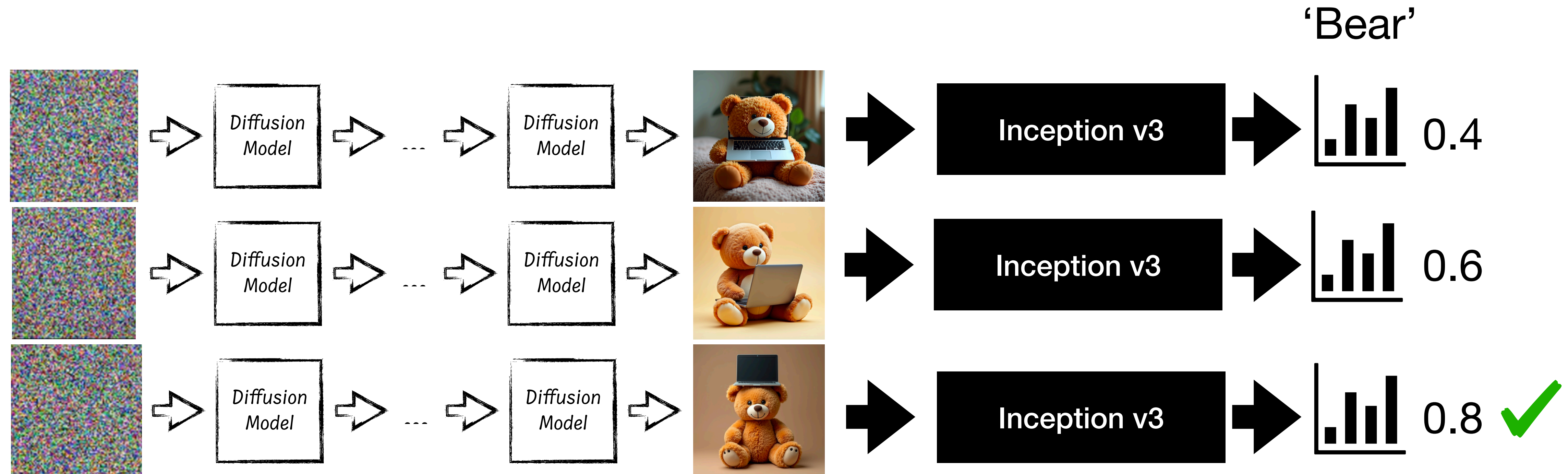
Find better candidates based on the verifiers scores

Verifiers

- Oracle Verifiers: Inception Score, FID Score

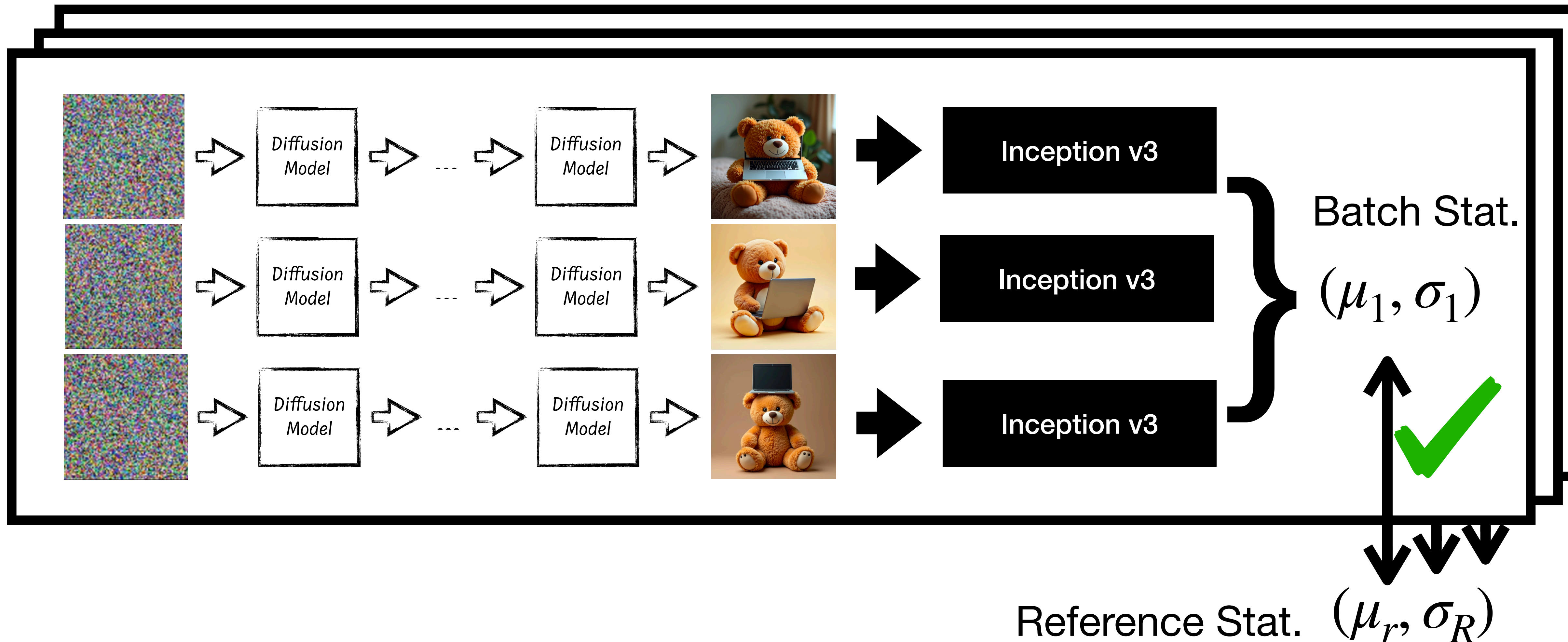
Verifiers

- Oracle Verifiers: Inception Score, FID Score



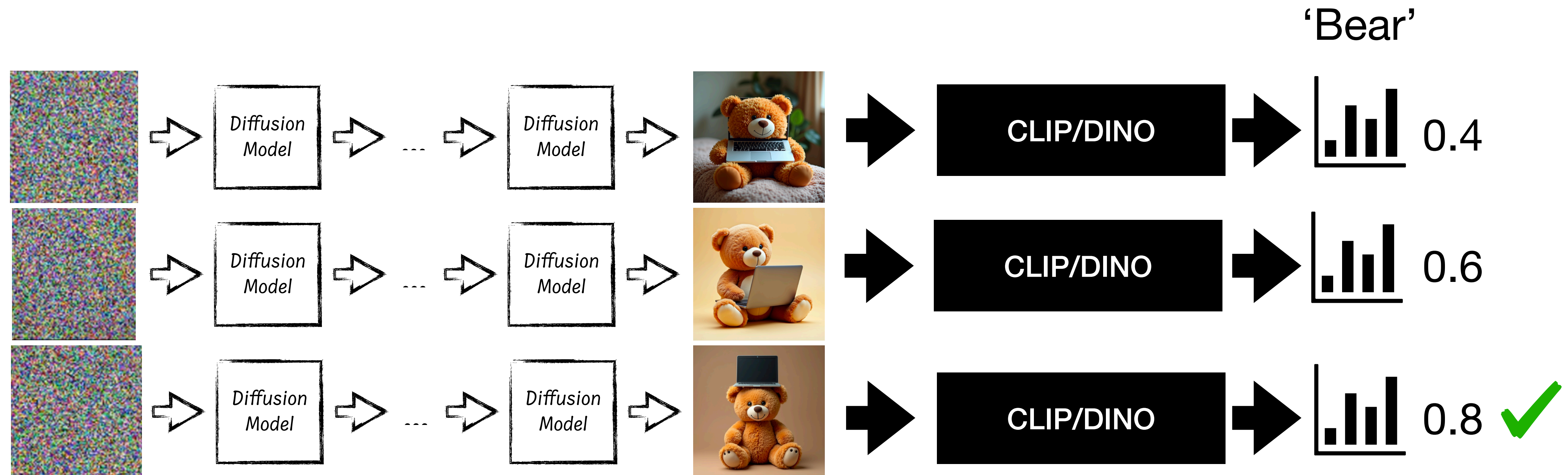
Verifiers

- Oracle Verifiers: Inception Score, FID Score



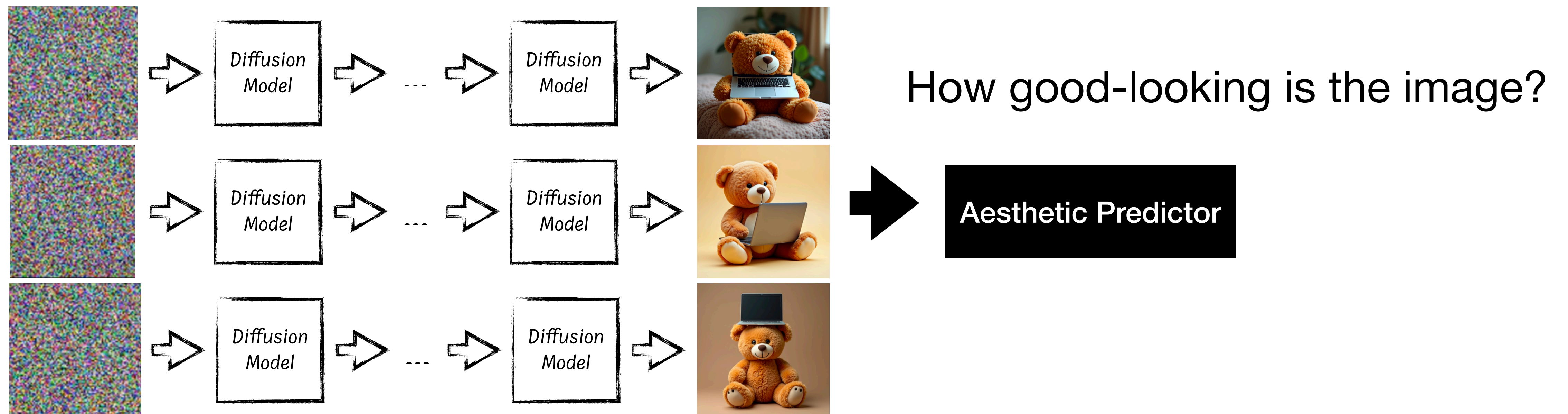
Verifiers

- Supervised Verifier: CLIP, DINO (with pretrained linear head)



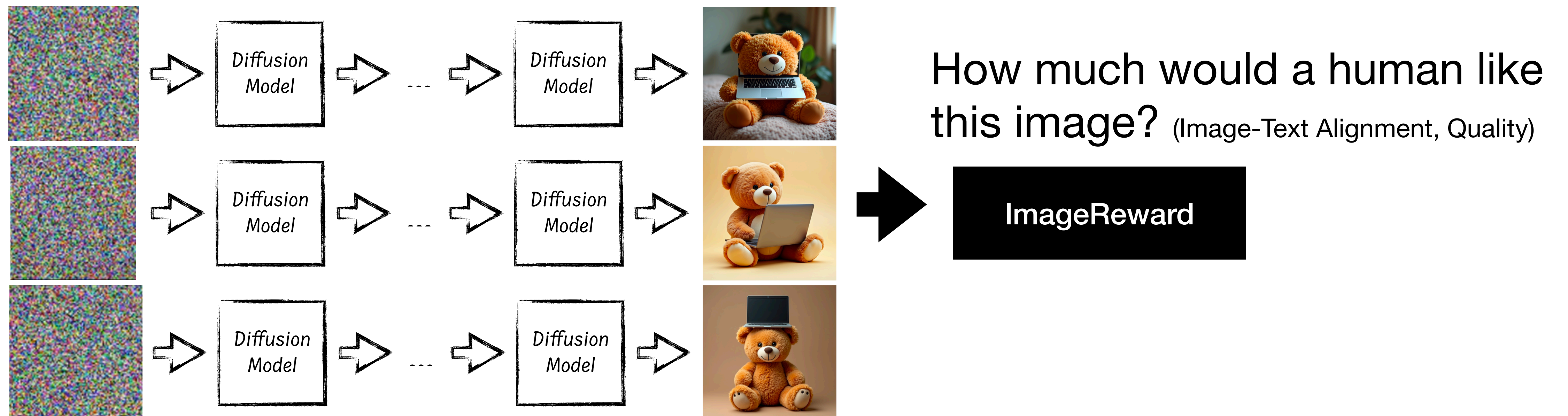
Verifiers

- Other Verifiers: Aesthetic Score, Imagereward, Verifier Ensemble



Verifiers

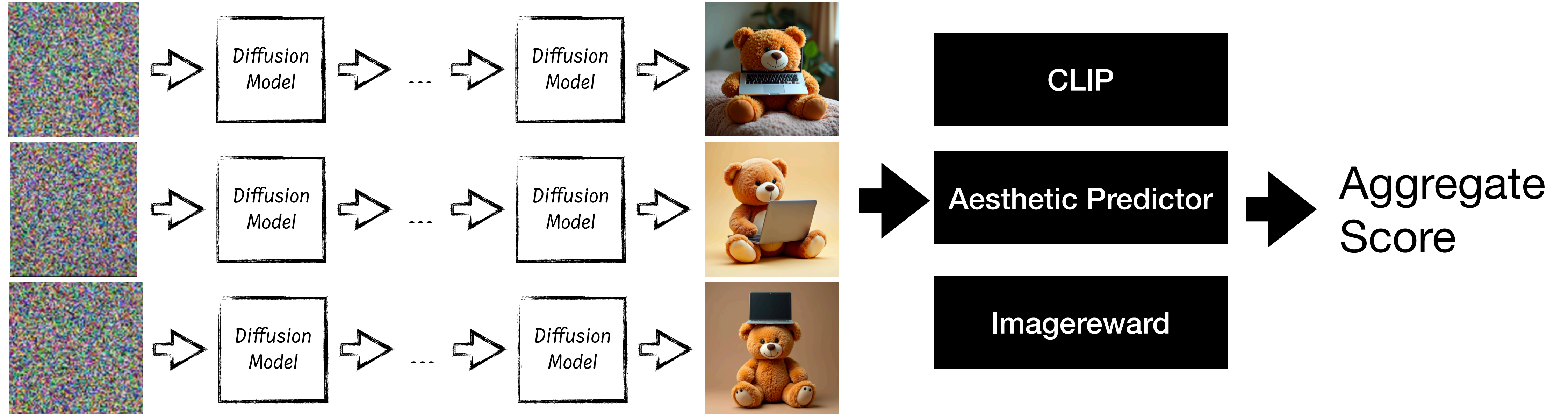
- Other Verifiers: Aesthetic Score, Imagereward, Verifier Ensemble



[Xu et al. ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation, 2024]

Verifiers

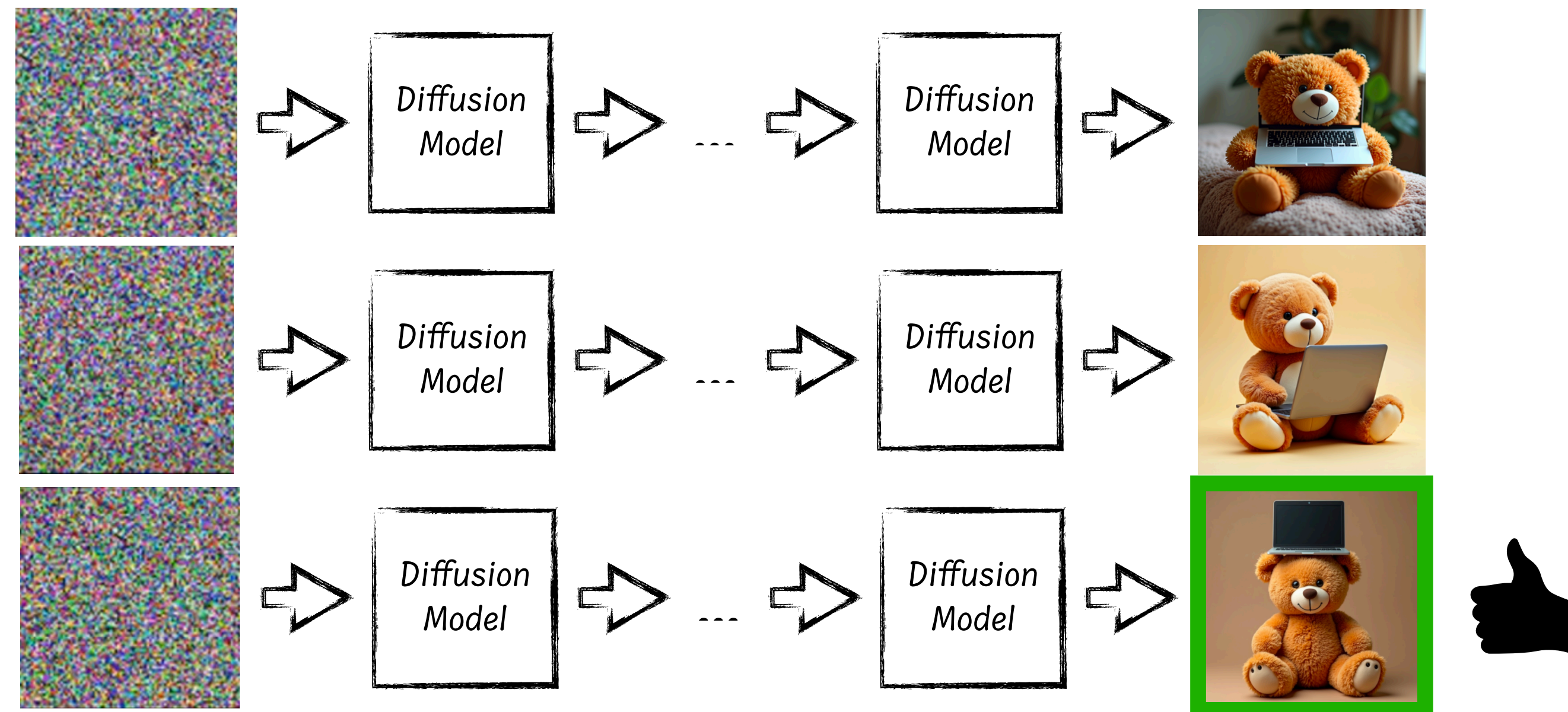
- Other Verifiers: Aesthetic Score, Imagereward, Verifier Ensemble



[Xu et al. ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation, 2024]

Search Algorithms

Prompt: A laptop **on top of** a teddy bear



1. Verifiers:

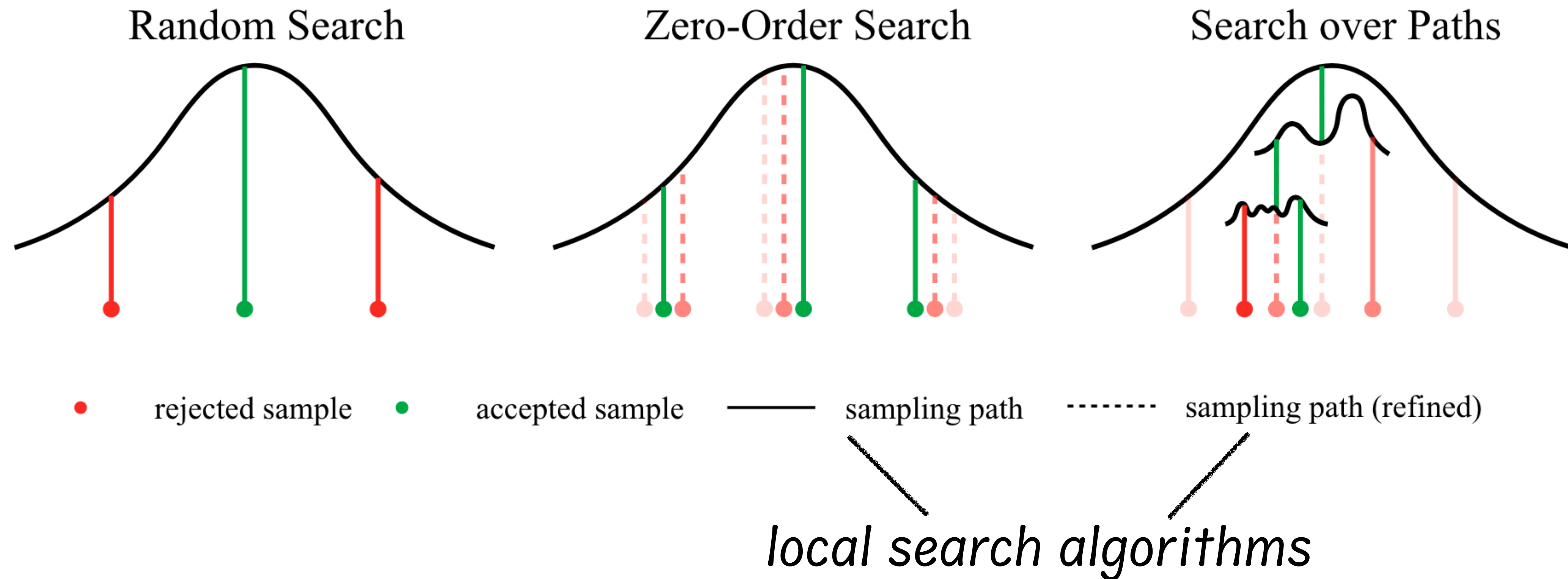
Evaluate the goodness of candidates

2. Search Algorithm:

Find better candidates based on the verifiers scores

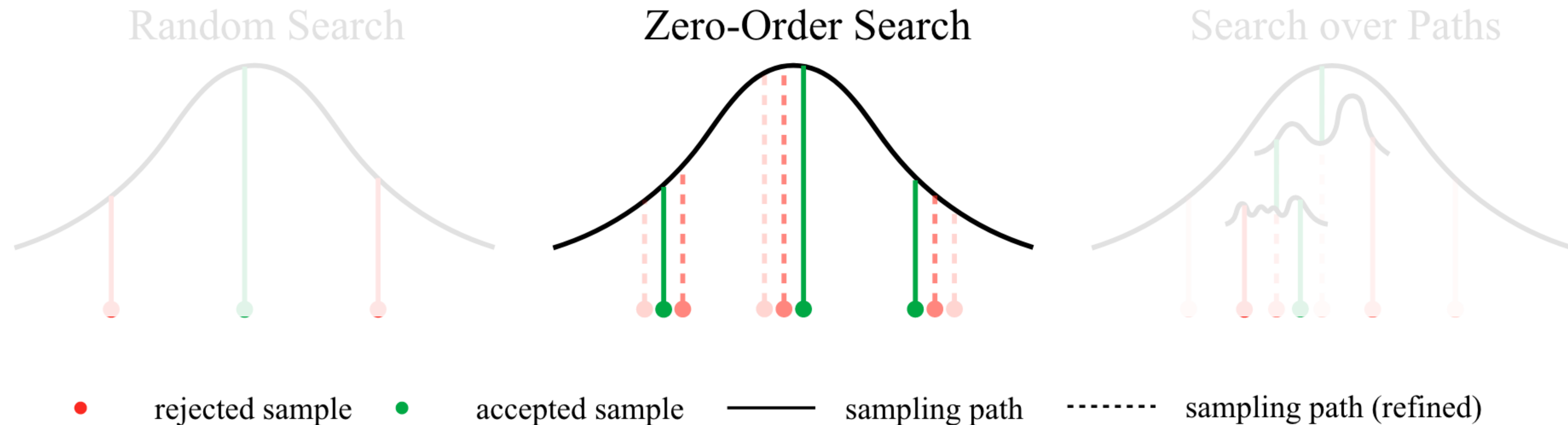
Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



Search Algorithms

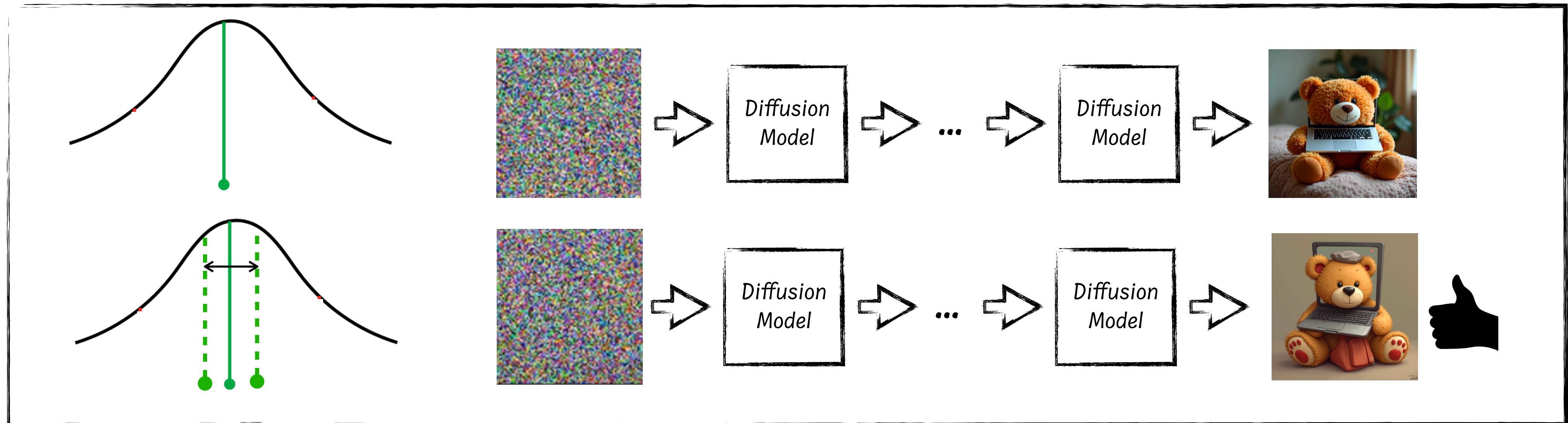
- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



Sample noises in a small neighborhood, find the best and iterate.

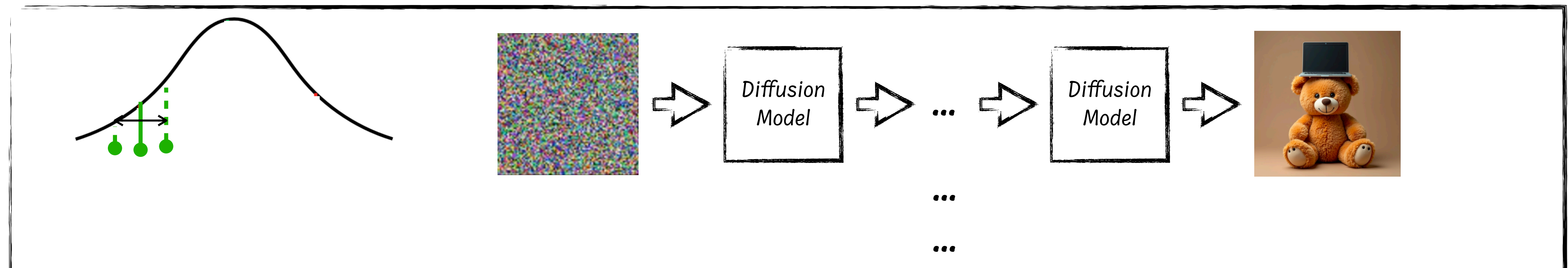
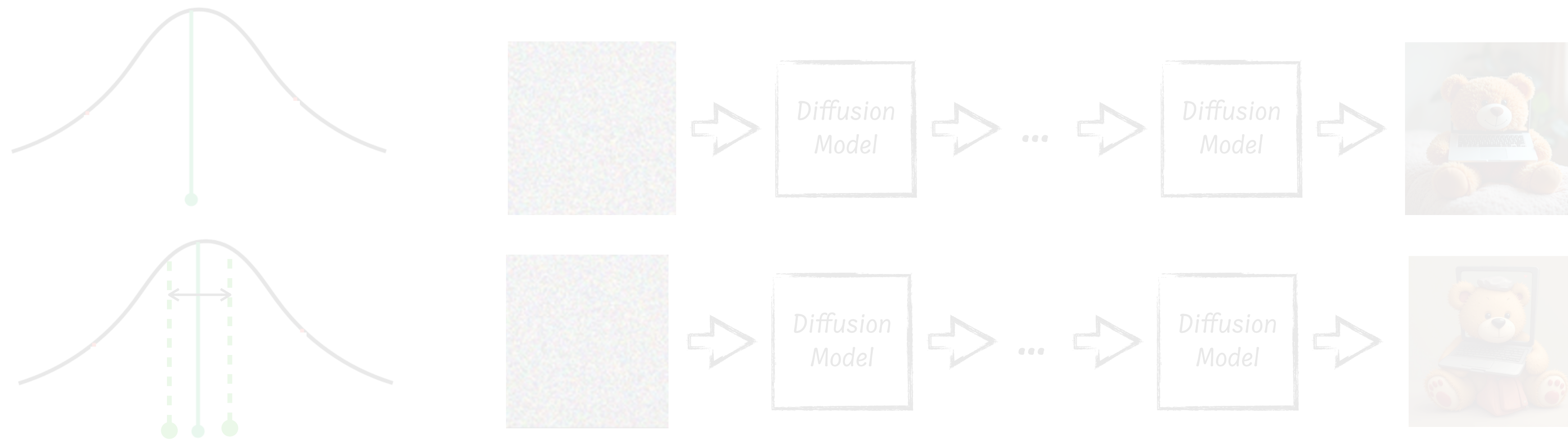
Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



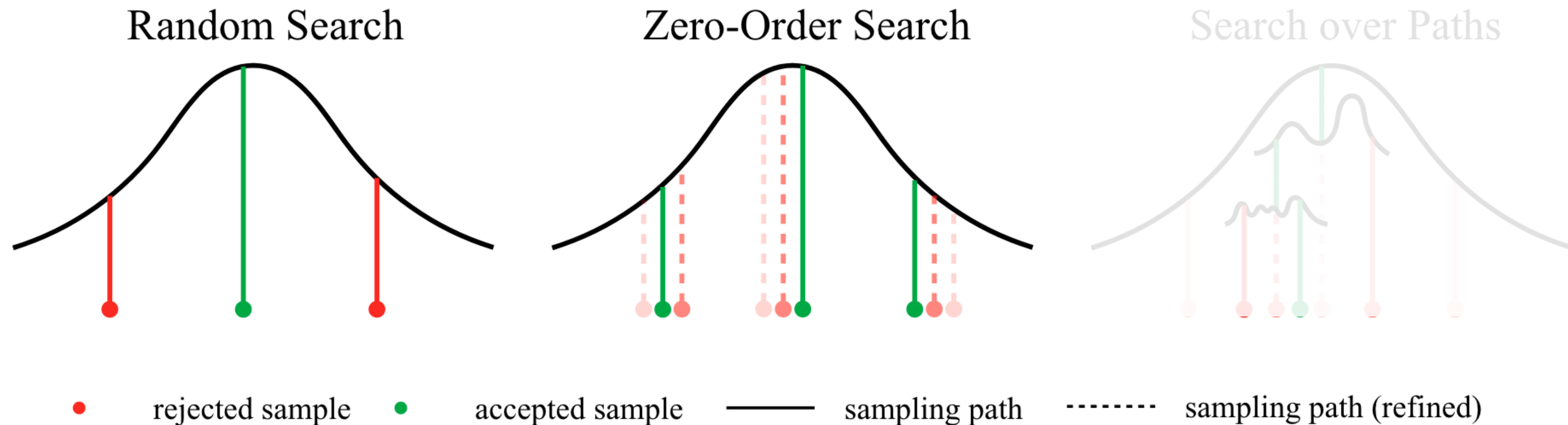
Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.

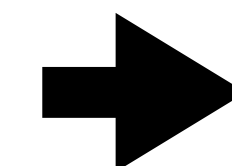


Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



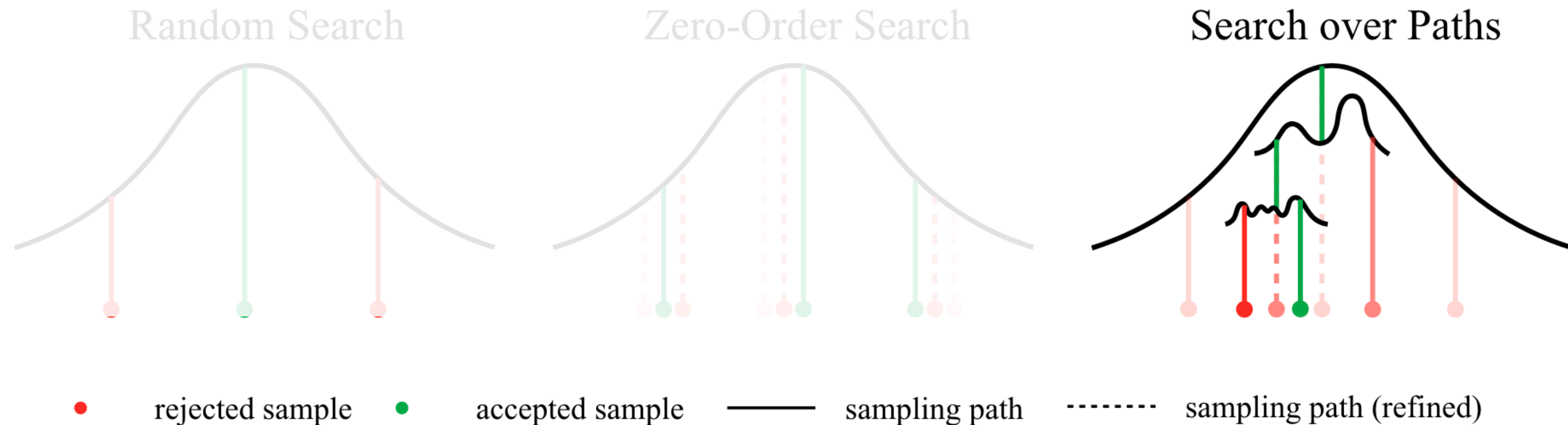
Find several local optimals.



Results are more diverse.

Search Algorithms

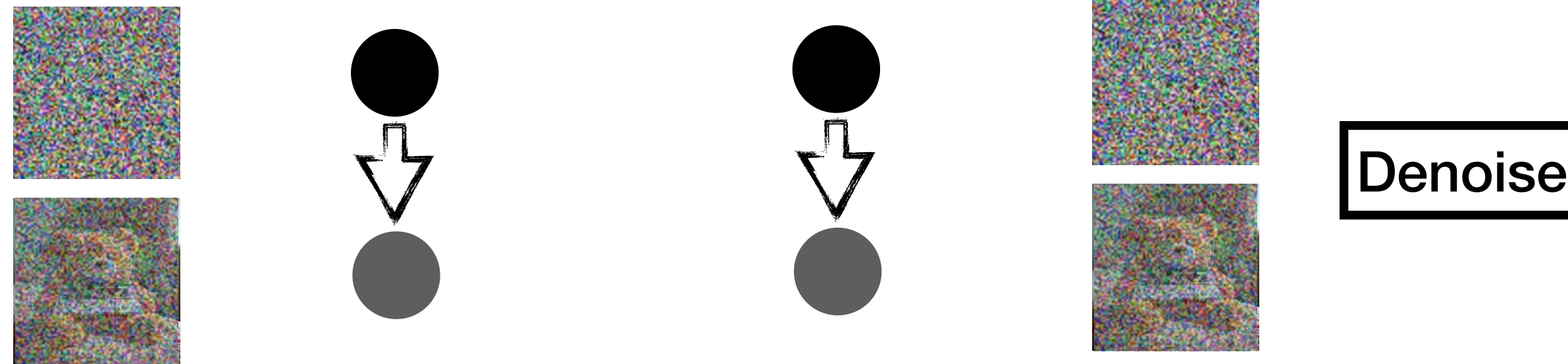
- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



Search along the trajectory at intermediate steps.

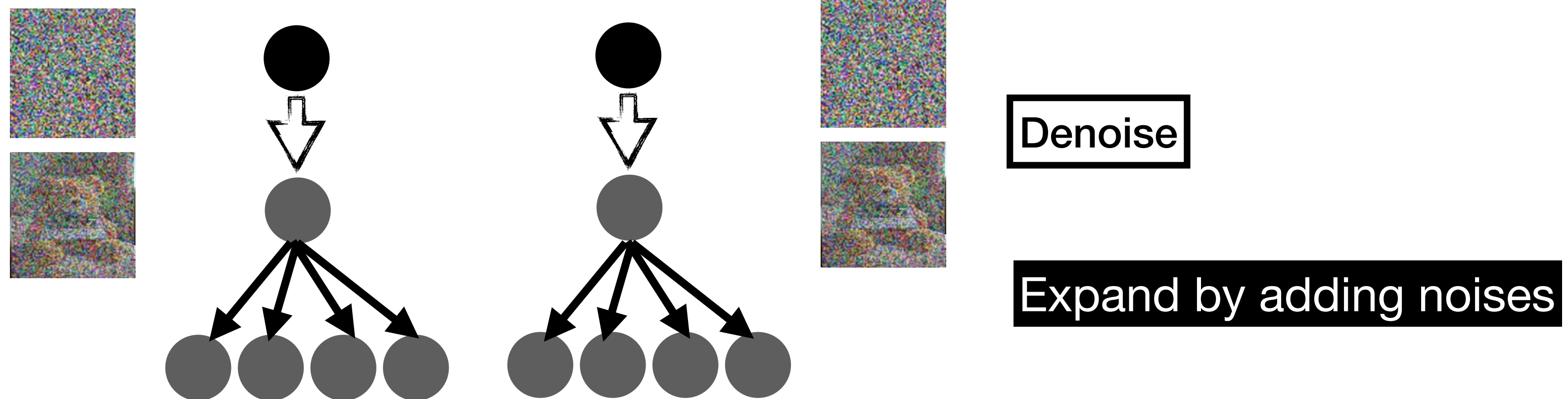
Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



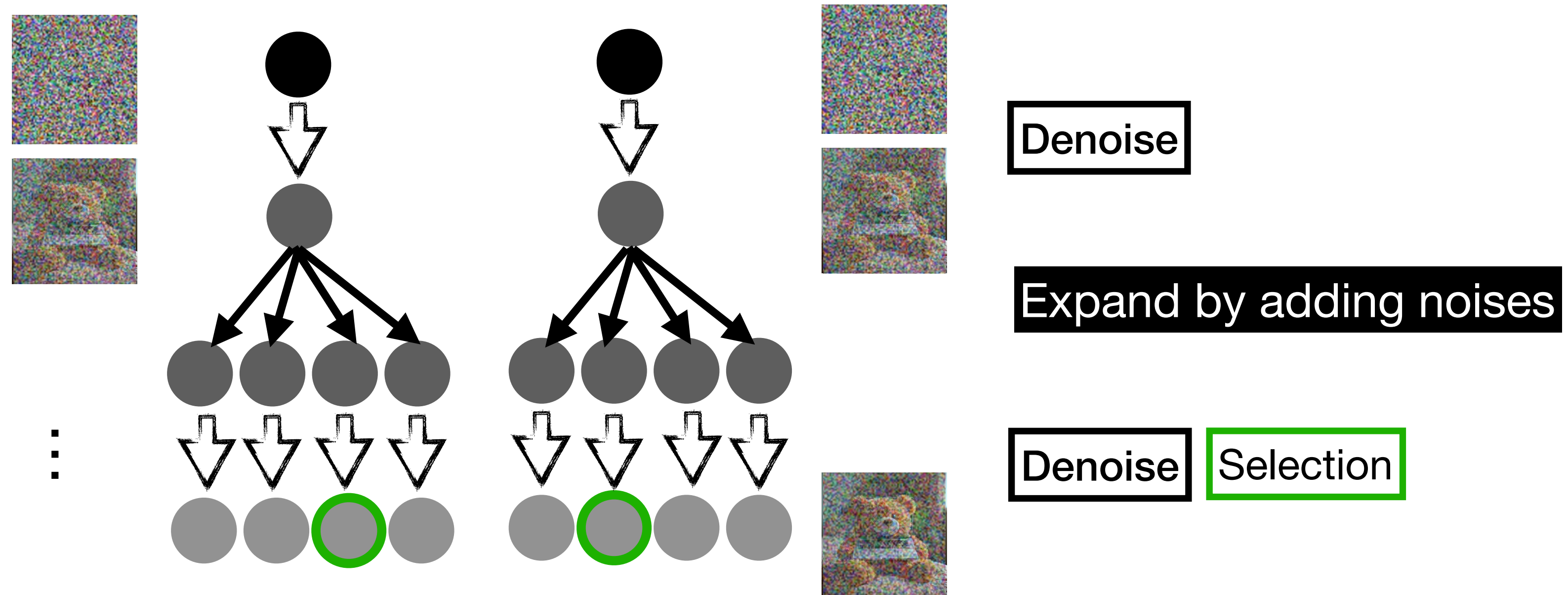
Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



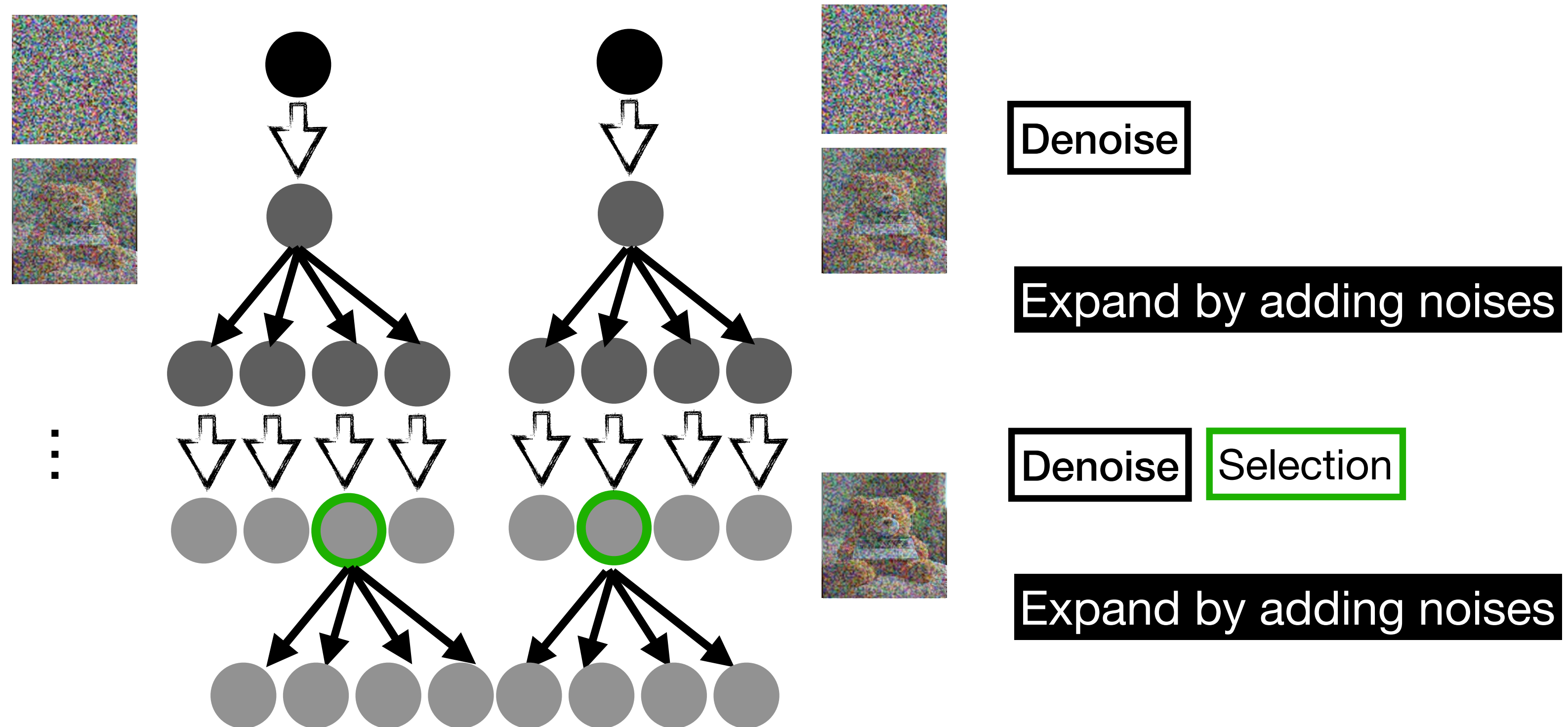
Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



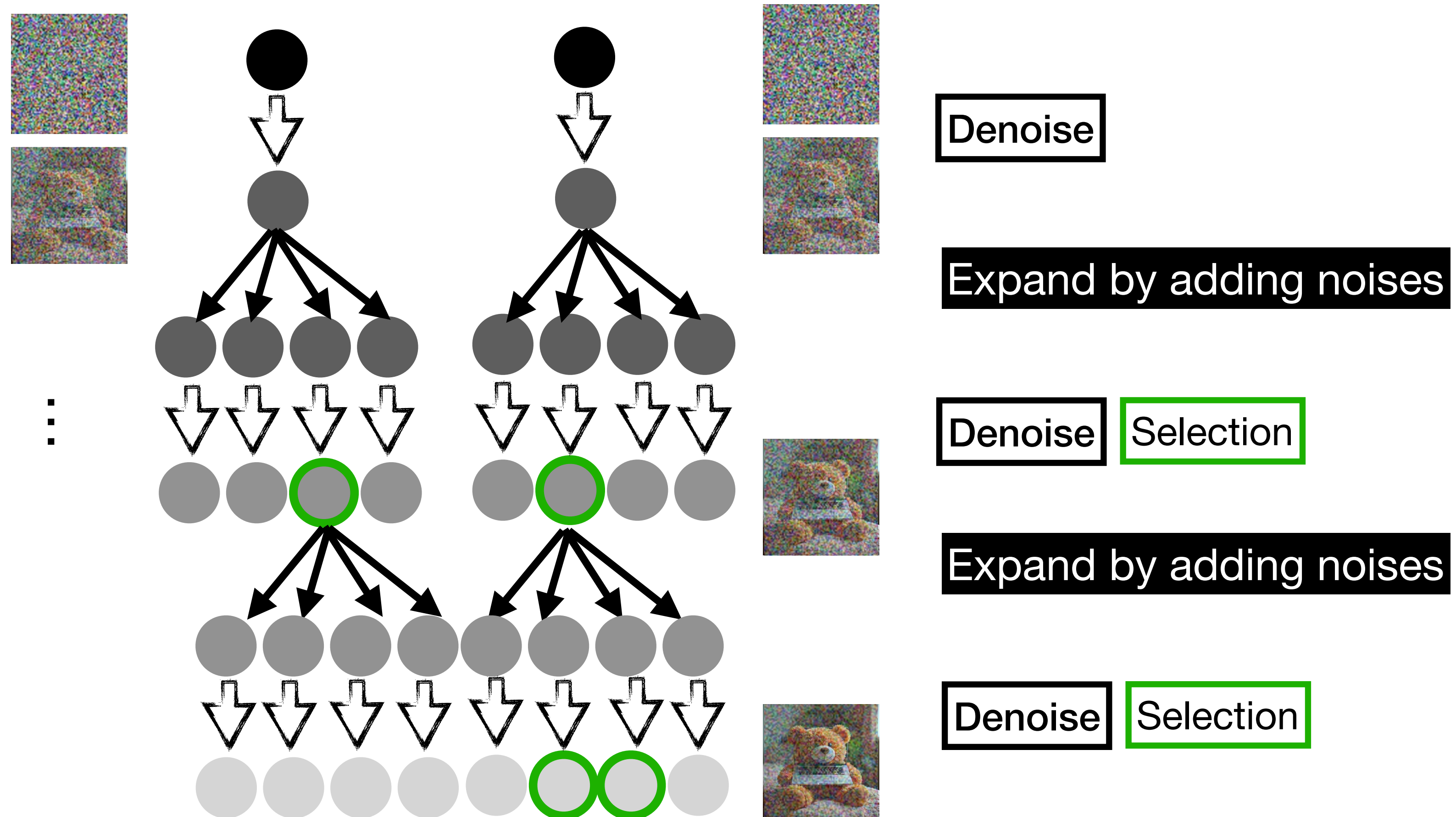
Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.

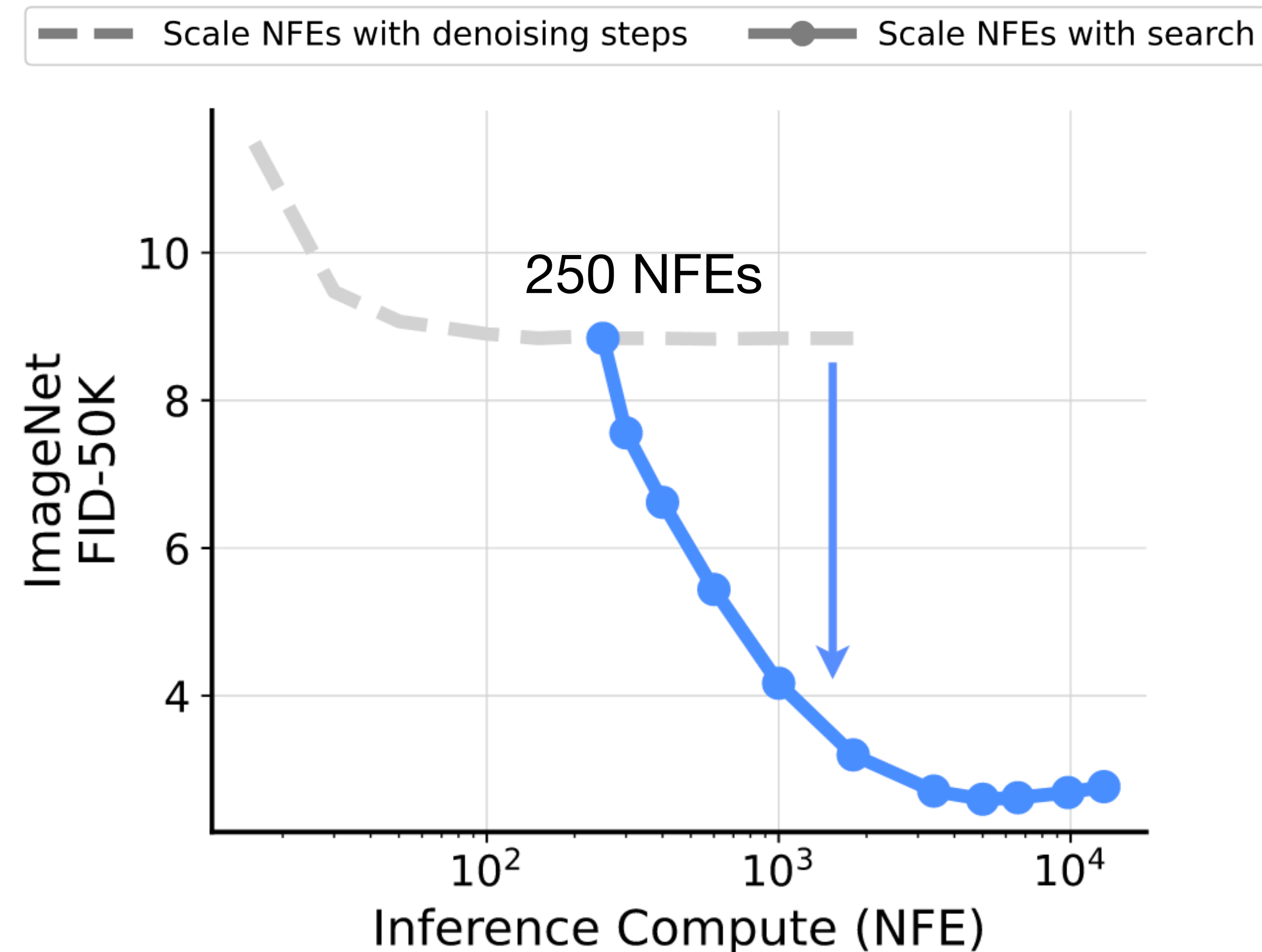


Search Algorithms

- **Search Algorithms:** Random Search, Zero-Order Search, and Search over Paths.



Main Results: Search enables better scaling behavior.



Number of Function Evaluations = denoising steps + search cost

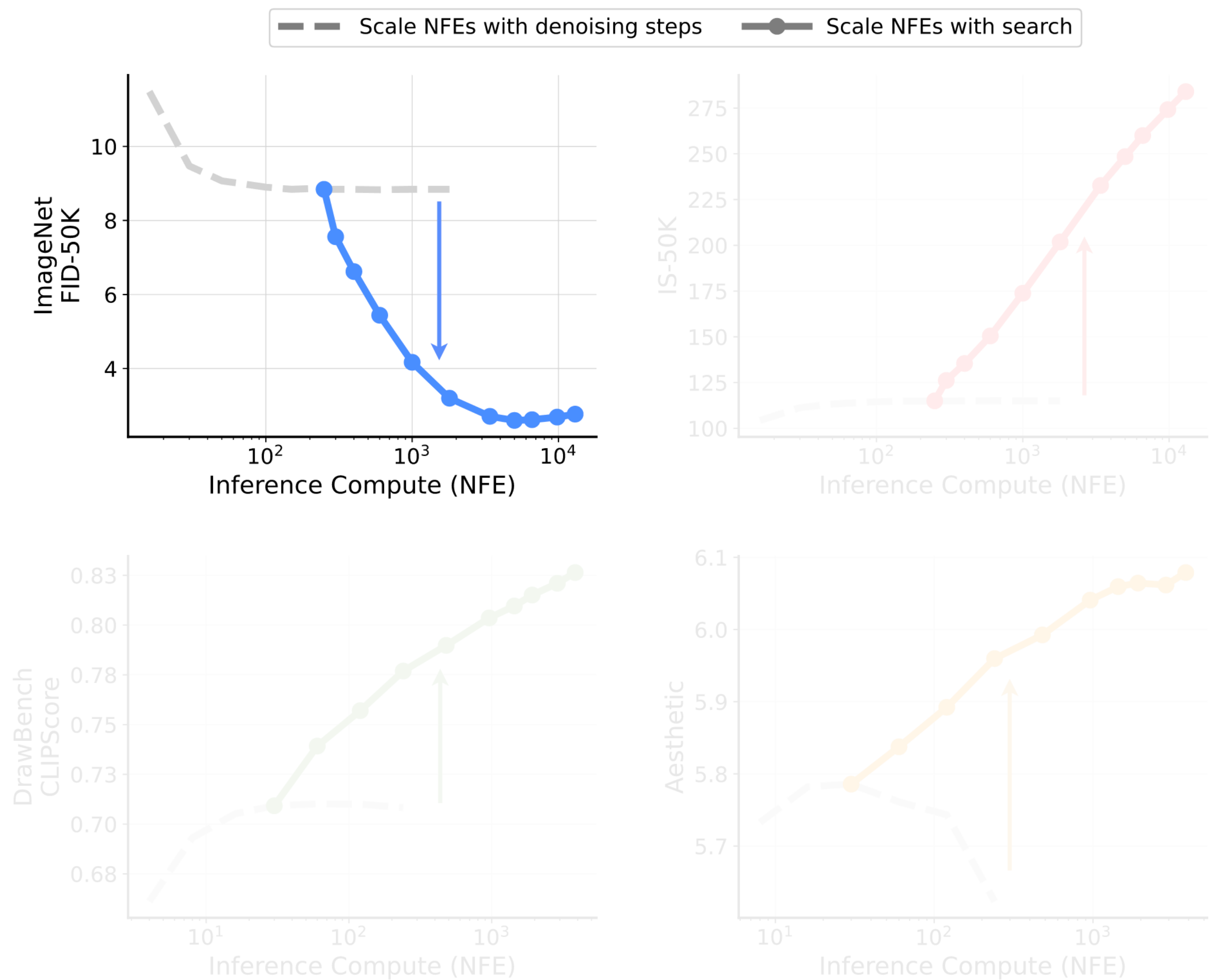
Model: SiT-XL_[1] and FLUX.1-dev _[2].

(Additional forward pass)

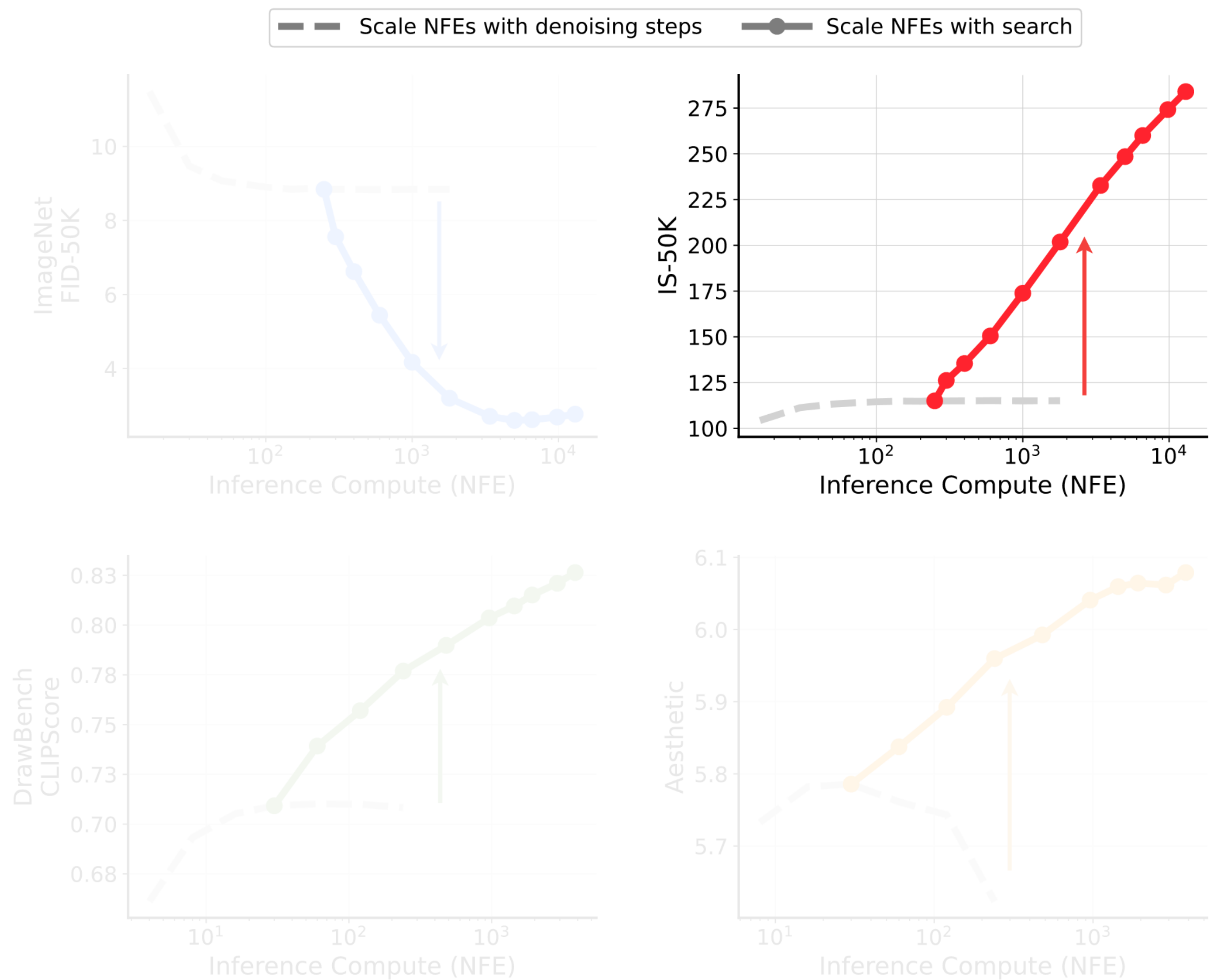
[1] Ma et al. SiT: Exploring Flow and Diffusion-based Generative Models with Scalable Interpolant Transformers, 2024

[2] B. F. Labs. Flux.1 [dev]. <https://blackforestlabs.ai/>.

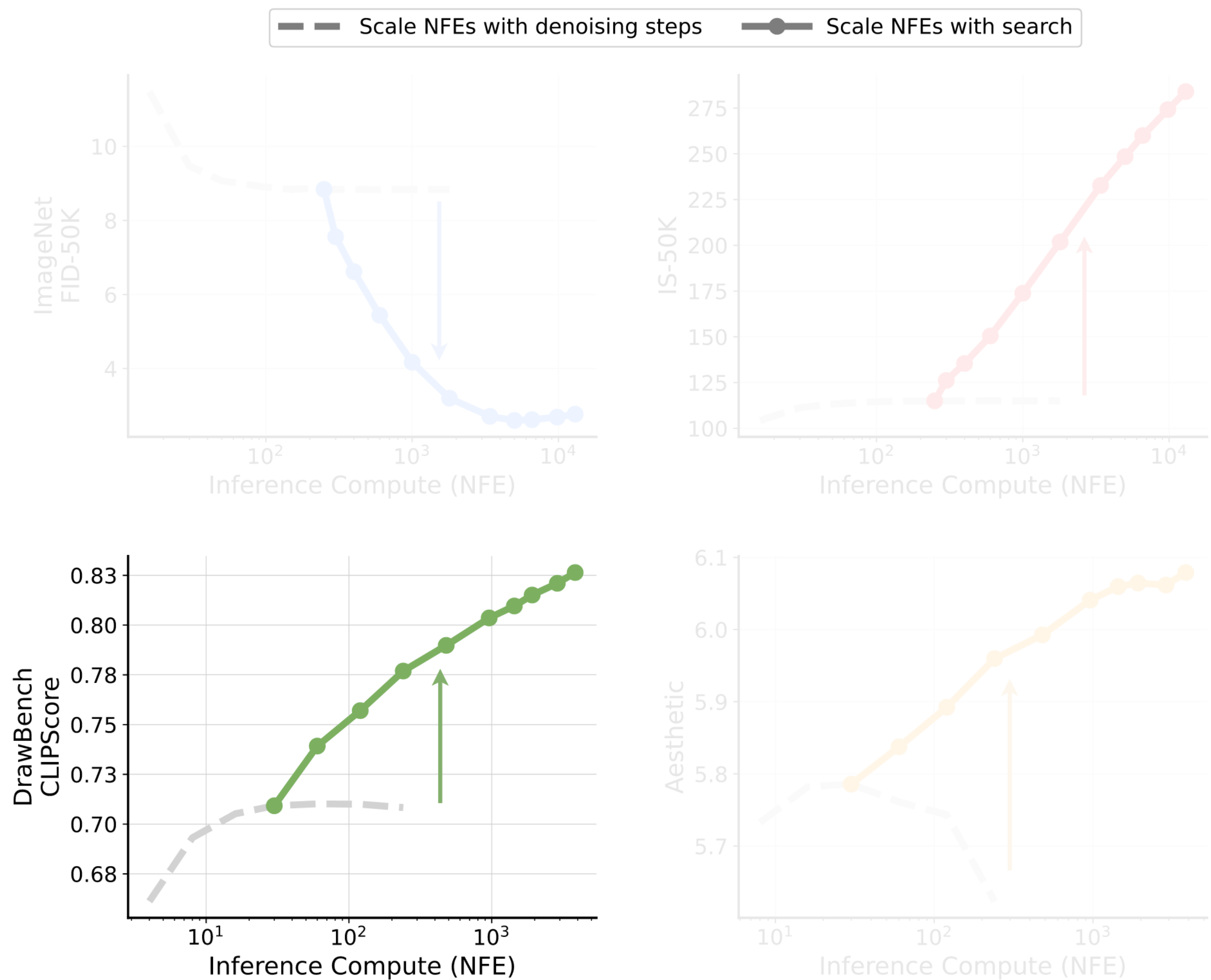
Main Results: Search enables better scaling behavior.



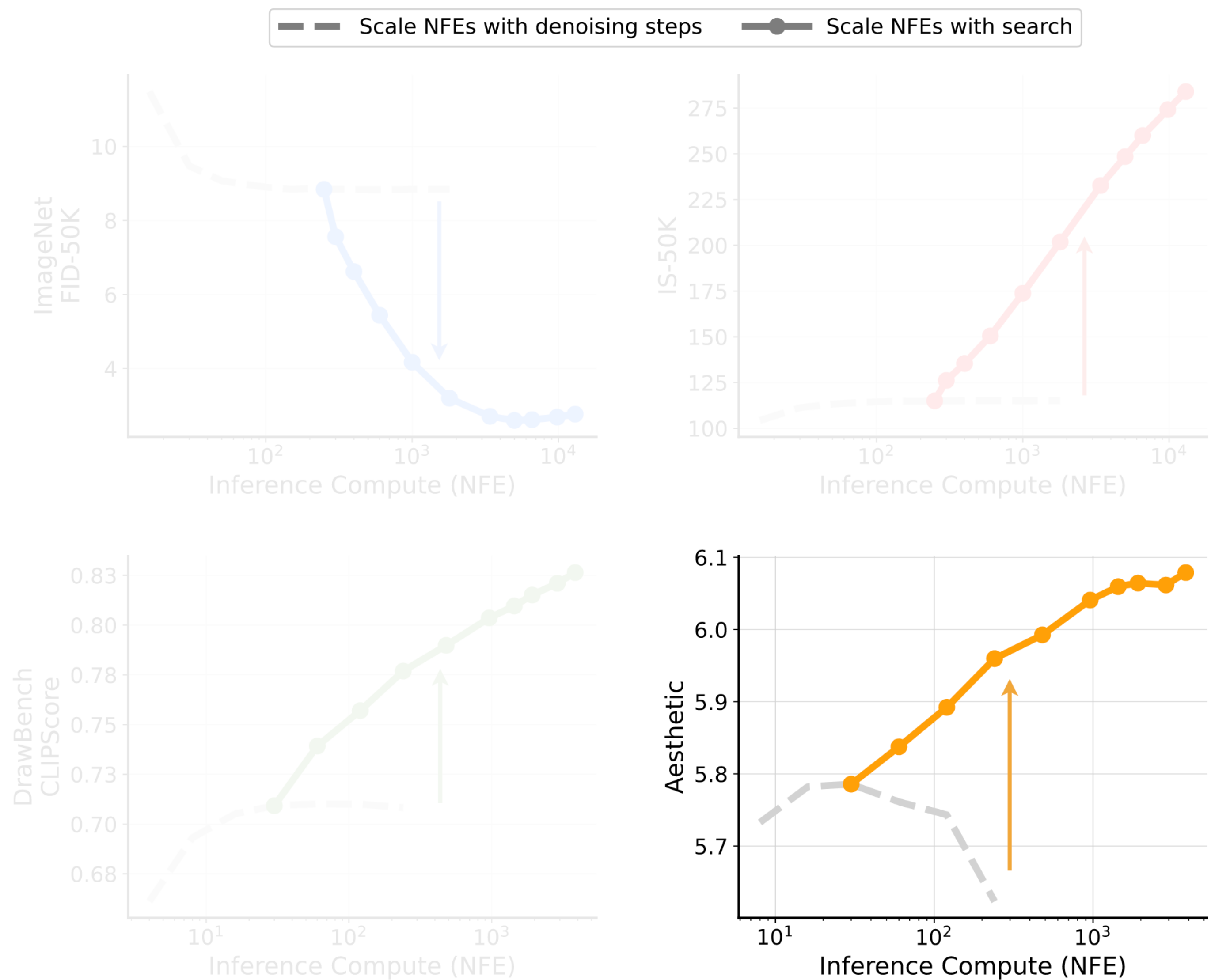
Main Results: Search enables better scaling behavior.



Main Results: Search enables better scaling behavior.



Main Results: Search enables better scaling behavior.



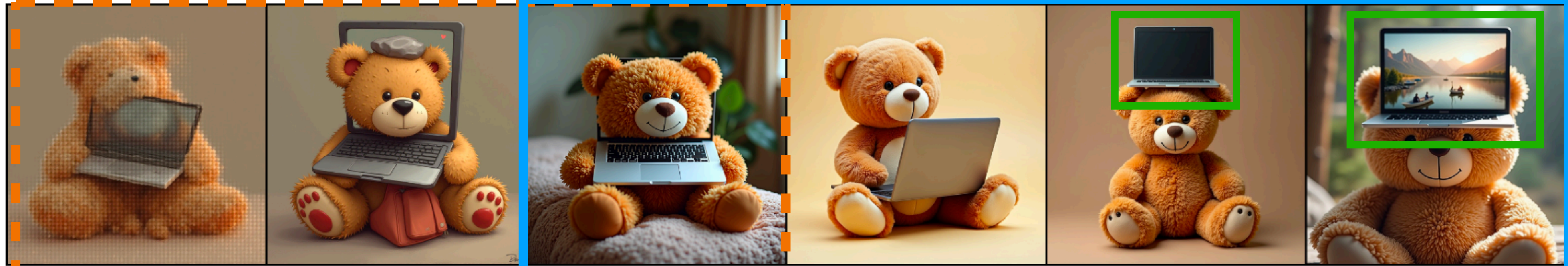
Visualization

Search leads to a better image-text alignment.

FLUX-1.dev [41]

Increasing NFEs in denoising steps

Increasing NFEs in search

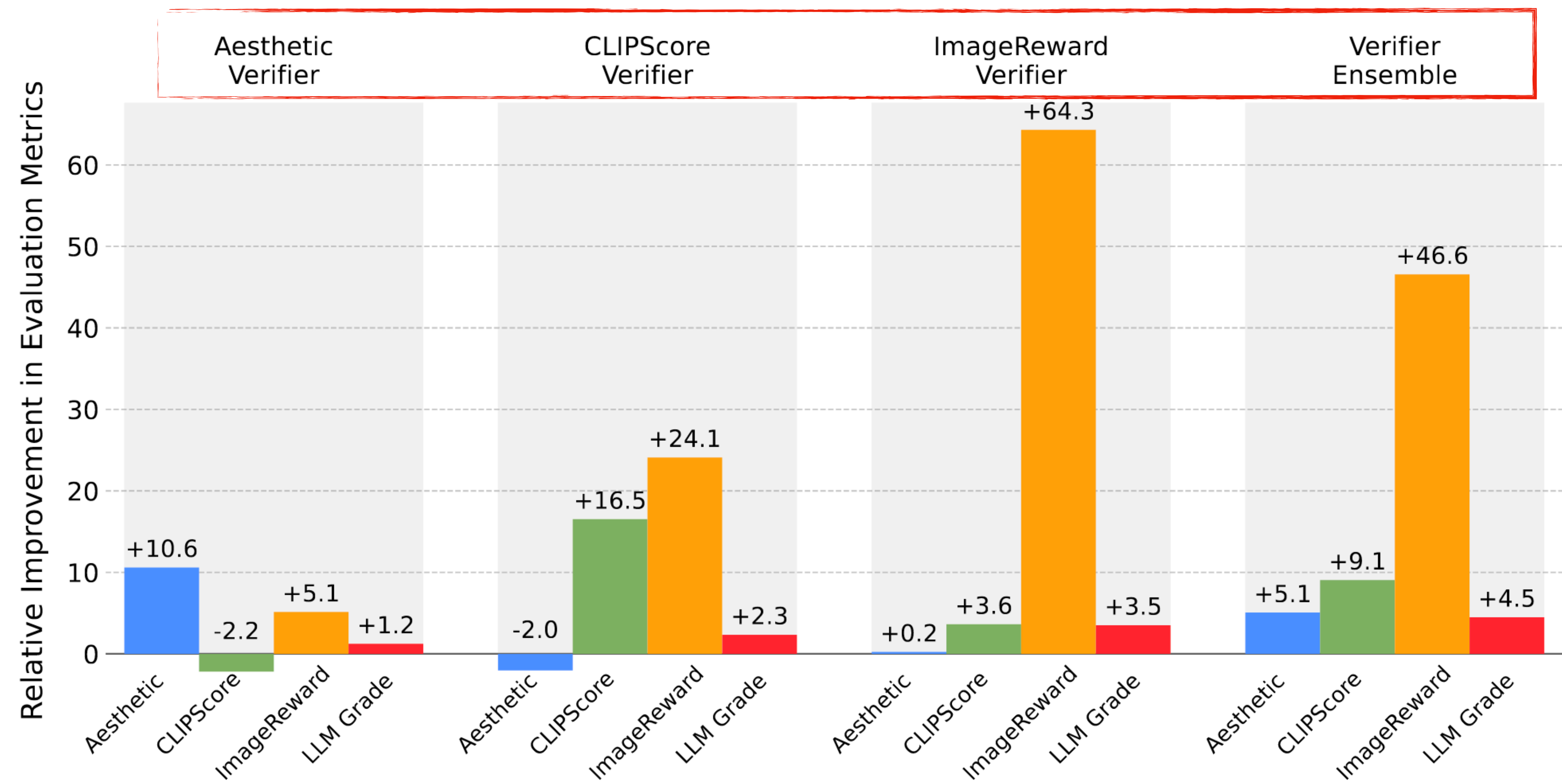


“A laptop on top of a teddy bear.”

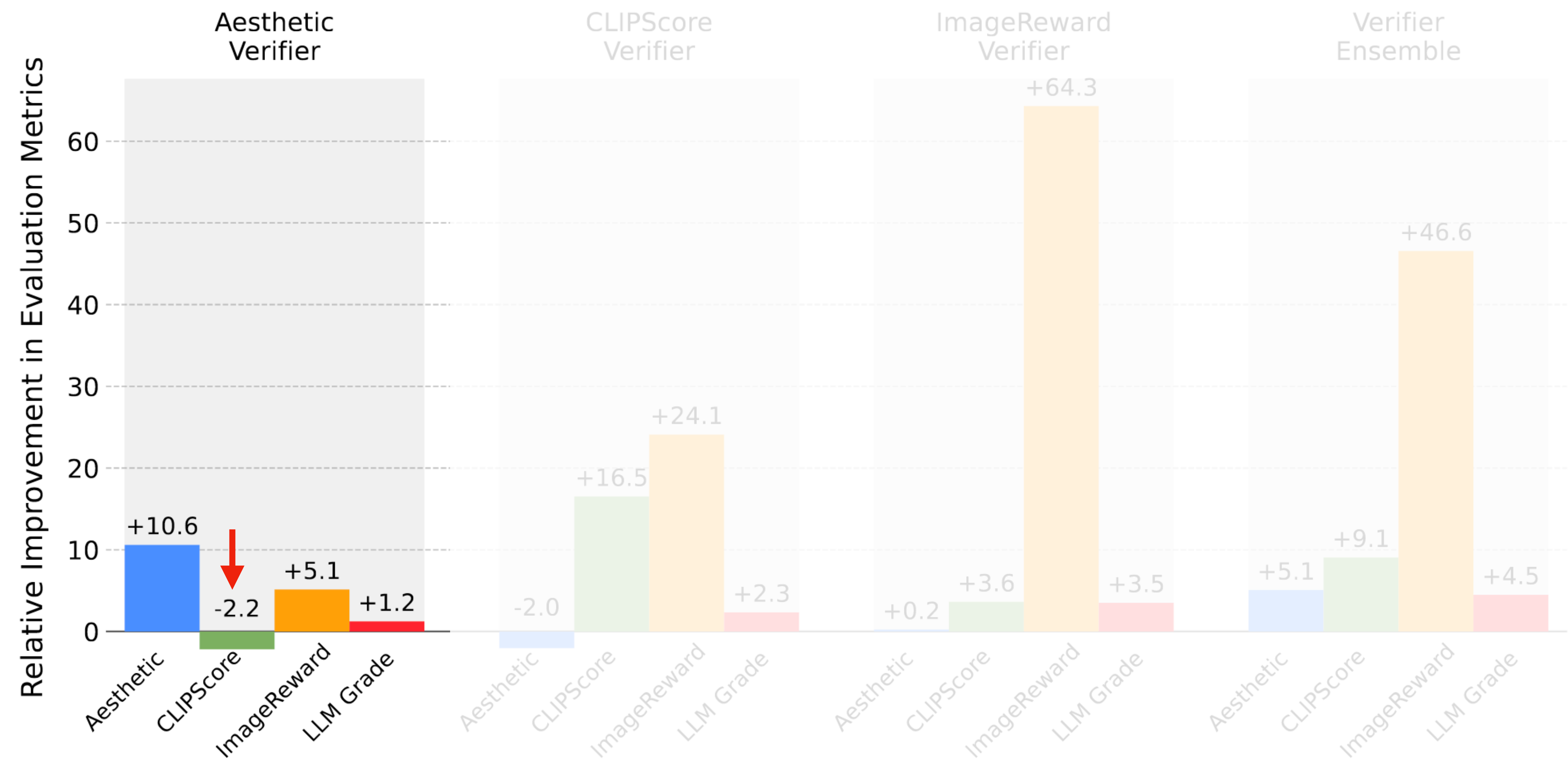


“Photo of an athlete cat explaining it’s latest scandal at a press conference to journalists.”

Verifier Results

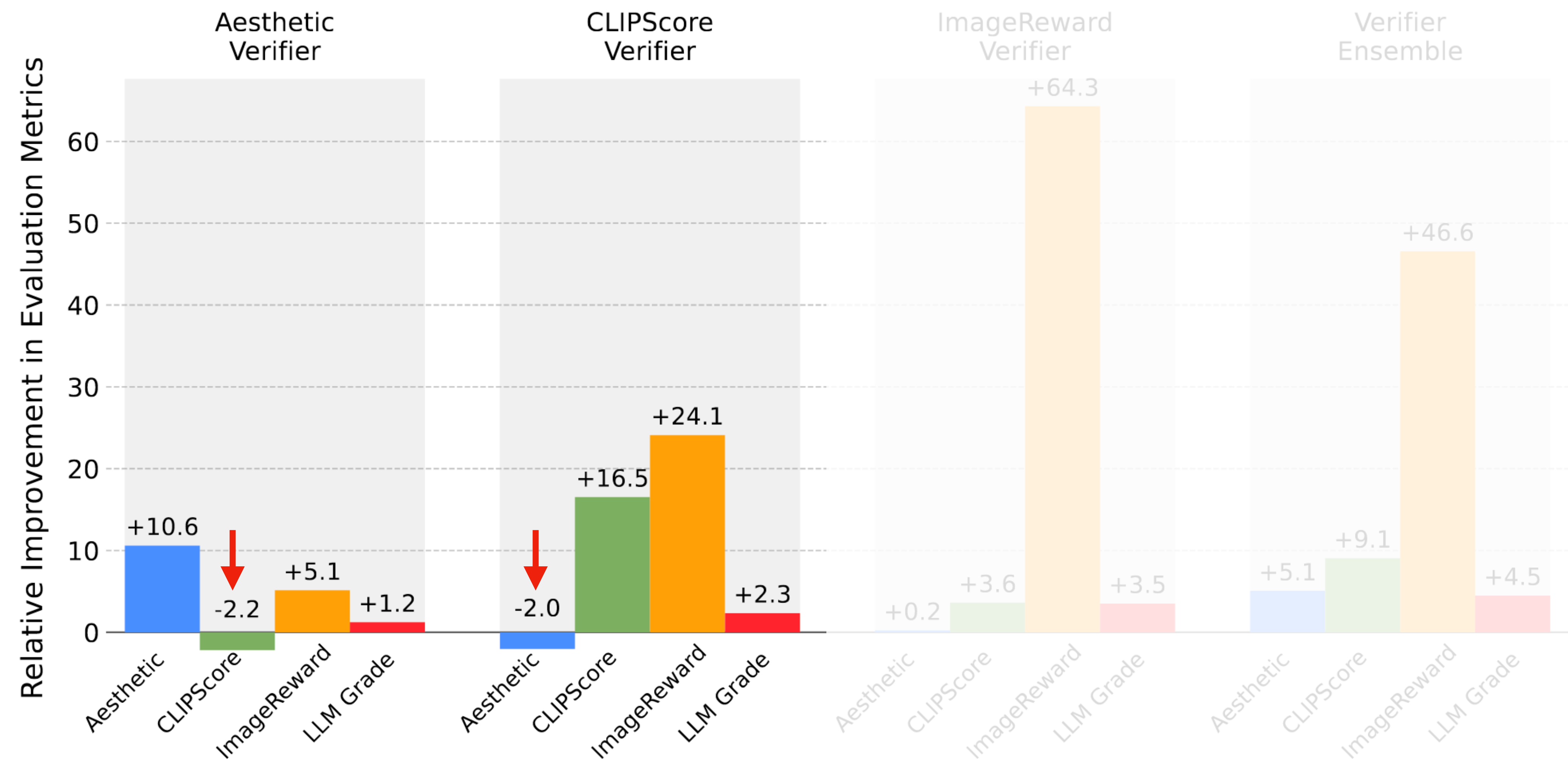


Verifier Results

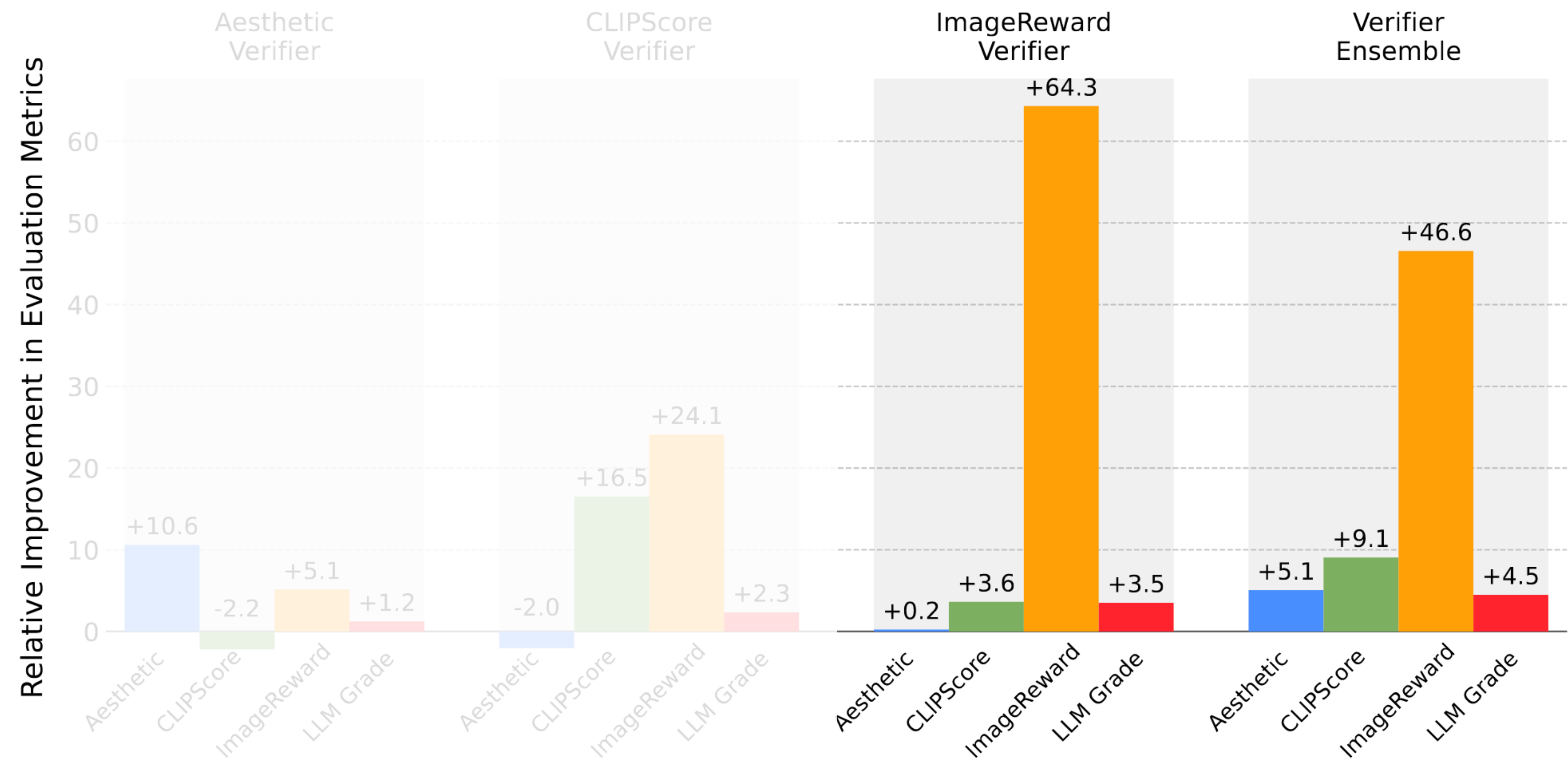


Verifier Results

Verifier Hacking

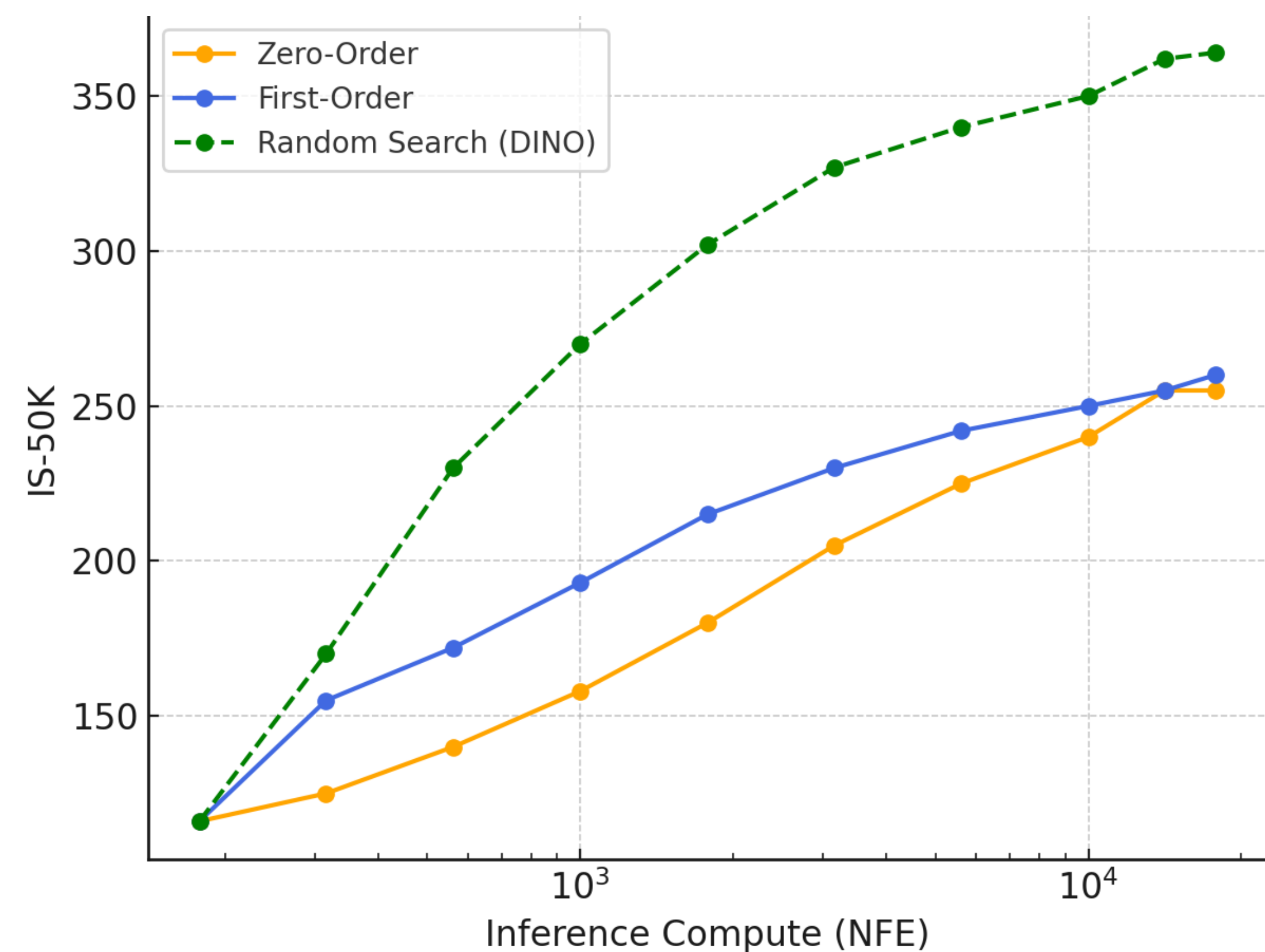
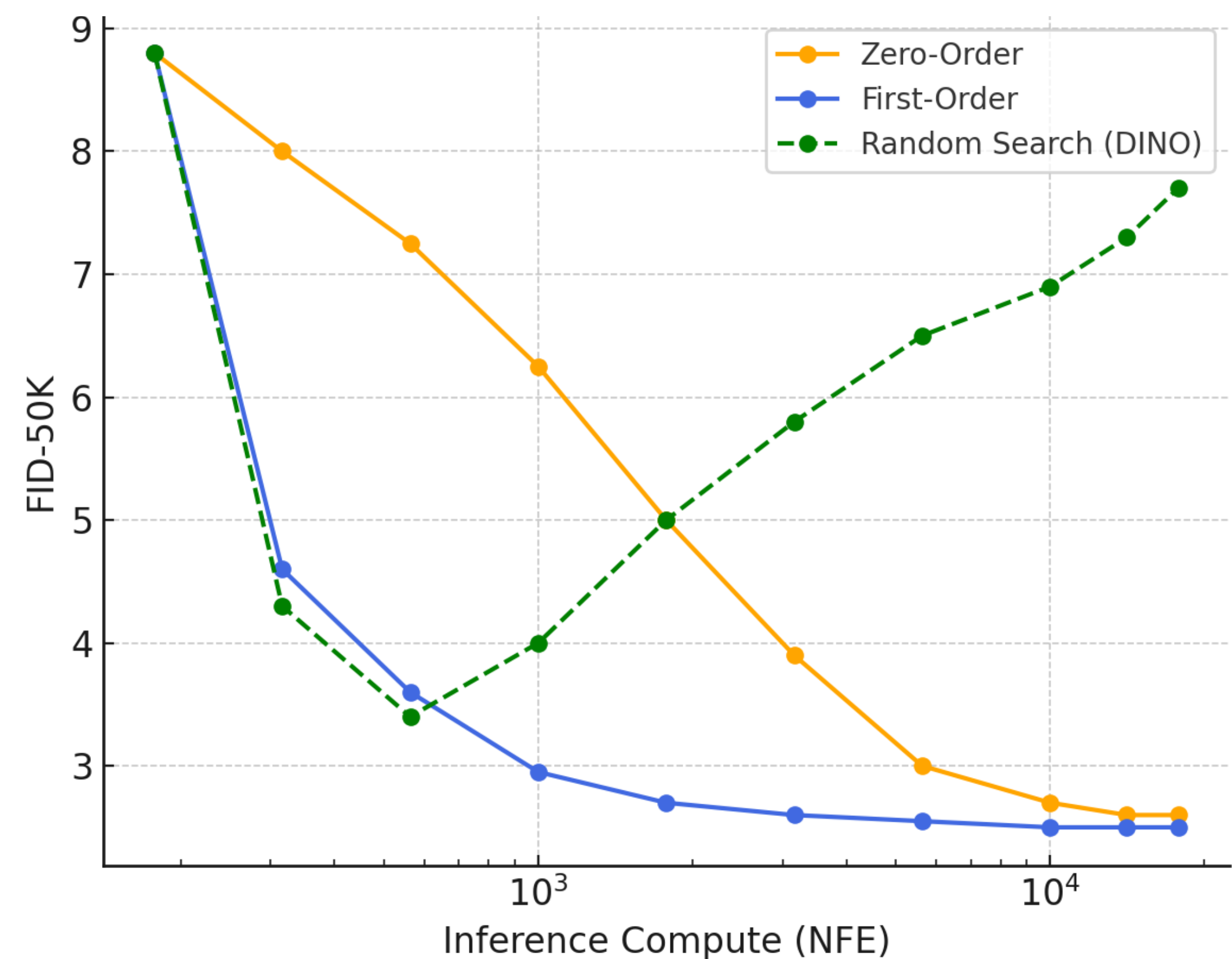


Verifier Results



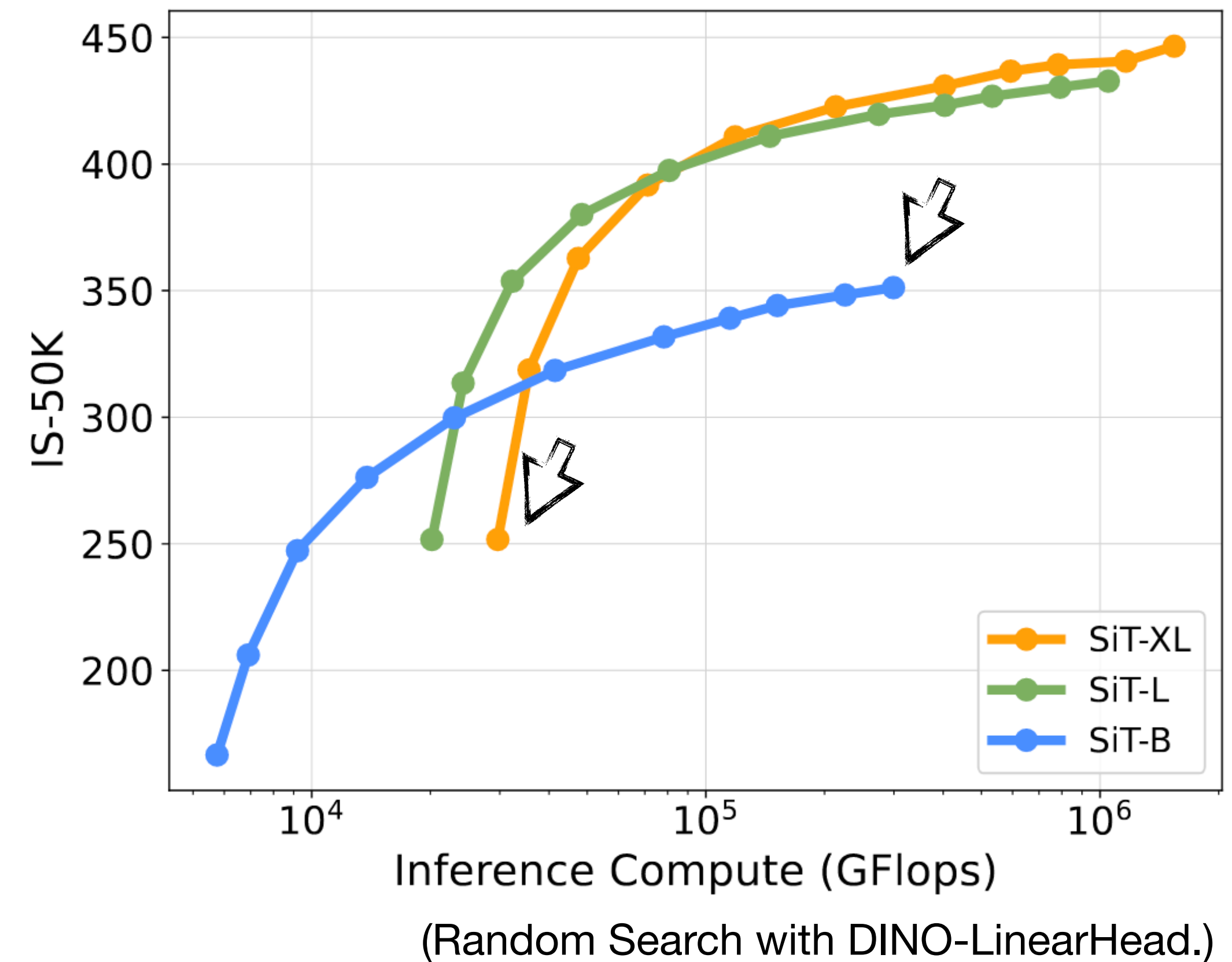
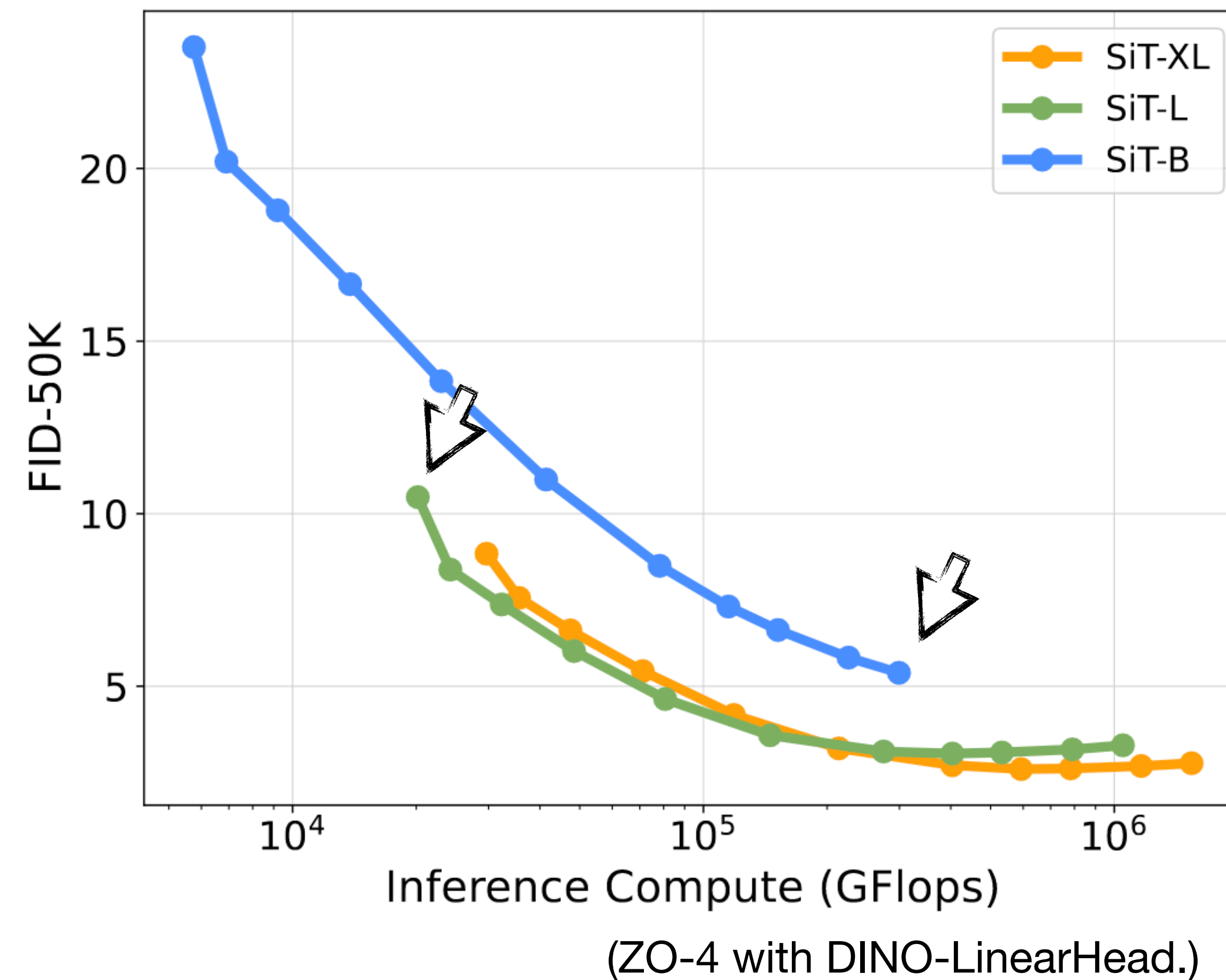
Search Algorithms Comparison

All algorithms use DINO as the verifier. Results on Imagenet.



Training-time Compute v.s. Test-time Compute

Performance of search methods across **different model sizes** (SiT-{B,L,XL}) on ImageNet.



Takeaways



- Search **works** in the image generation domain!



- **Limited Scope** to Diffusion Models.
- Lacks advanced verifiers and more sophisticated search algorithms..

Overview

Background

Introduce “test-time scaling” and “search”

1st paper

“Mastering the game of Go with deep neural networks and tree search” (AlphaGo)

Game Play

2nd paper

“DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning”

Large Language Models

3rd paper

“Inference-Time Scaling for Diffusion Models beyond Scaling Denoising Steps”

Image Generation

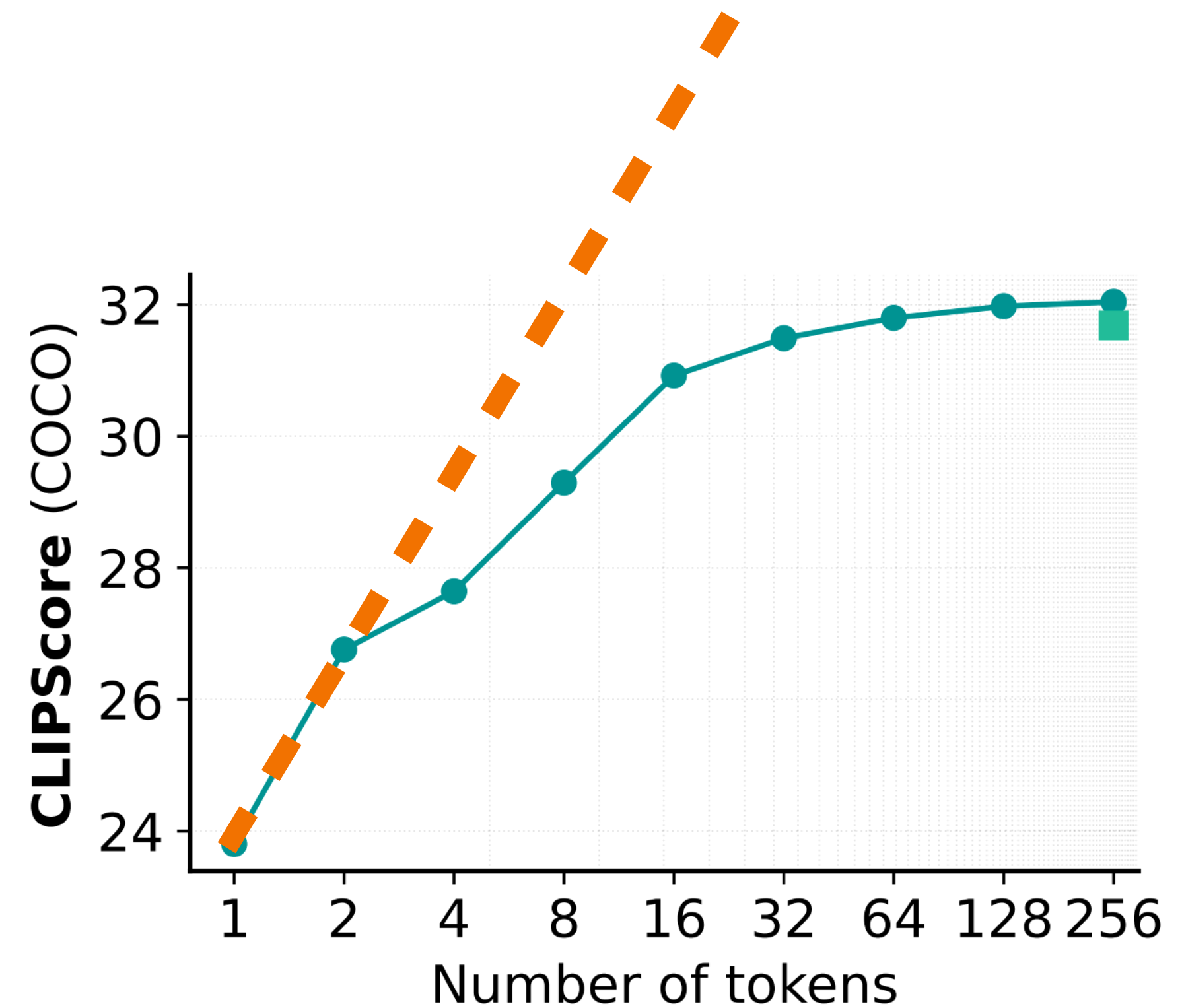
Discussion & Research Proposal

Ongoing explorations and future work

Main Idea: Visual Test-time Scaling via Search

How about Search?

Can we find a better test-time scaling method for vision generative models?

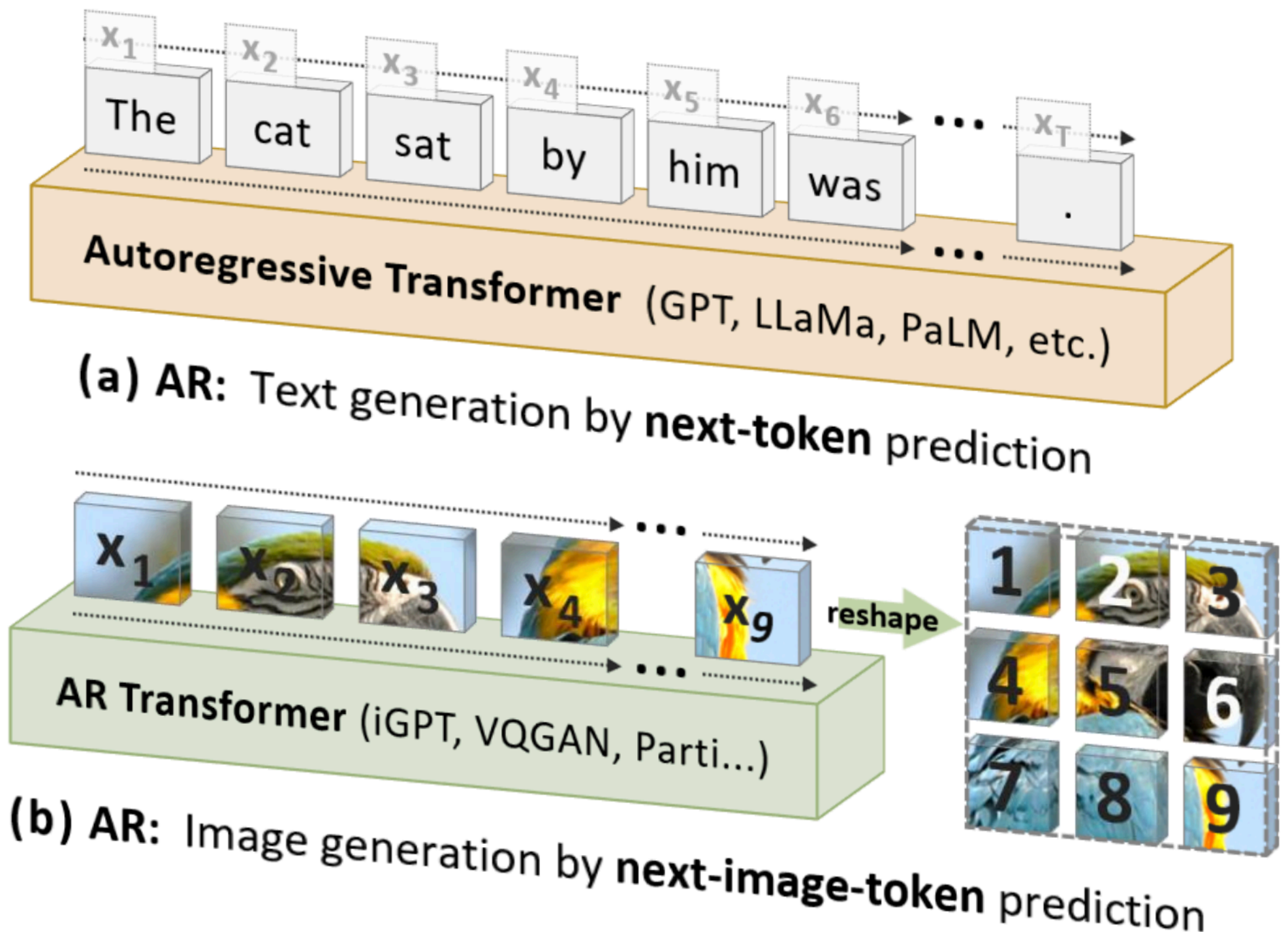


Main Idea: Visual Test-time Scaling via Search

How about Search?

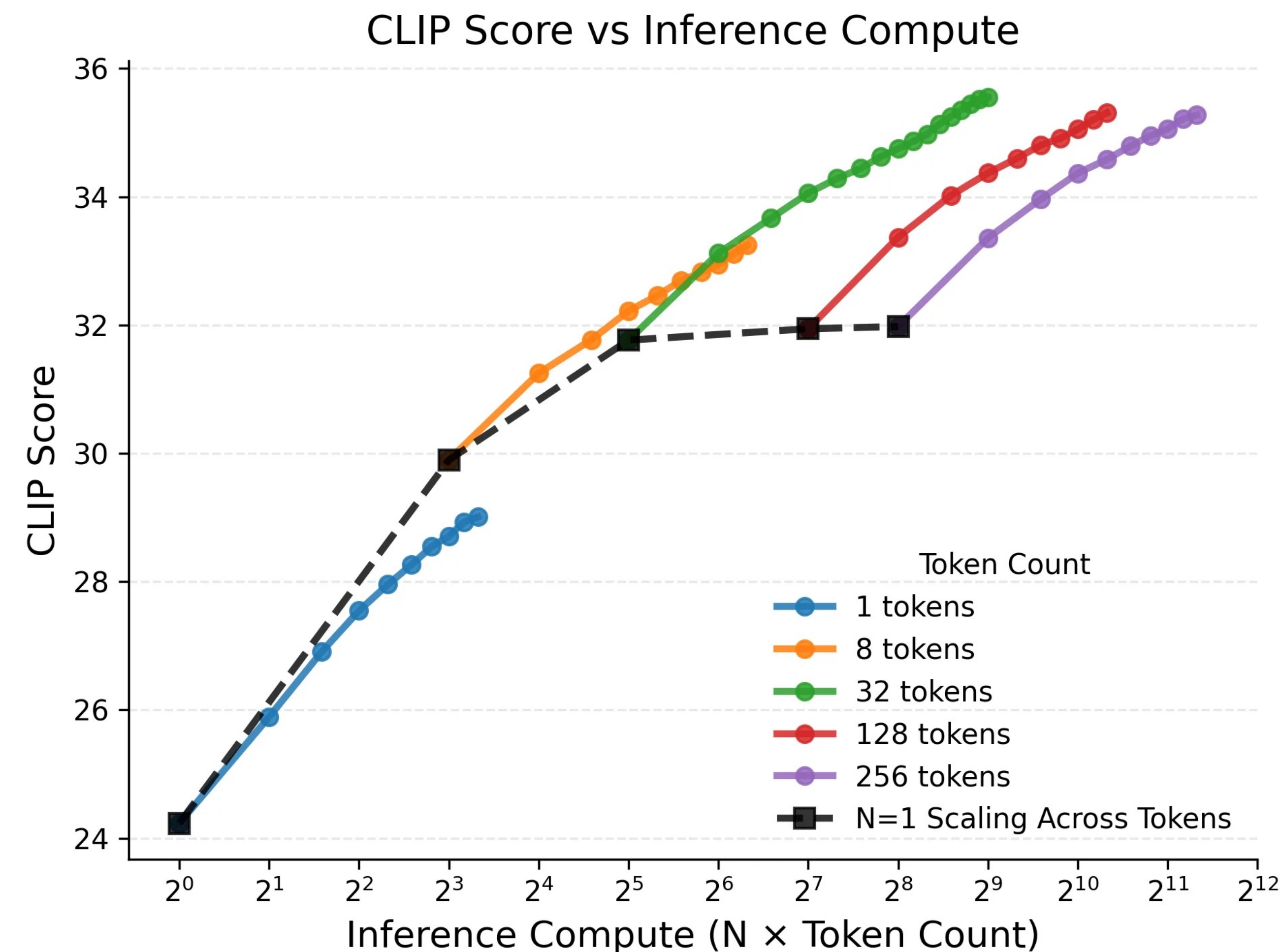
Can we find a better test-time scaling method for vision generative models?

We start with
**autoregressive image
generation models**



[Tian et al. Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction, 2024.]

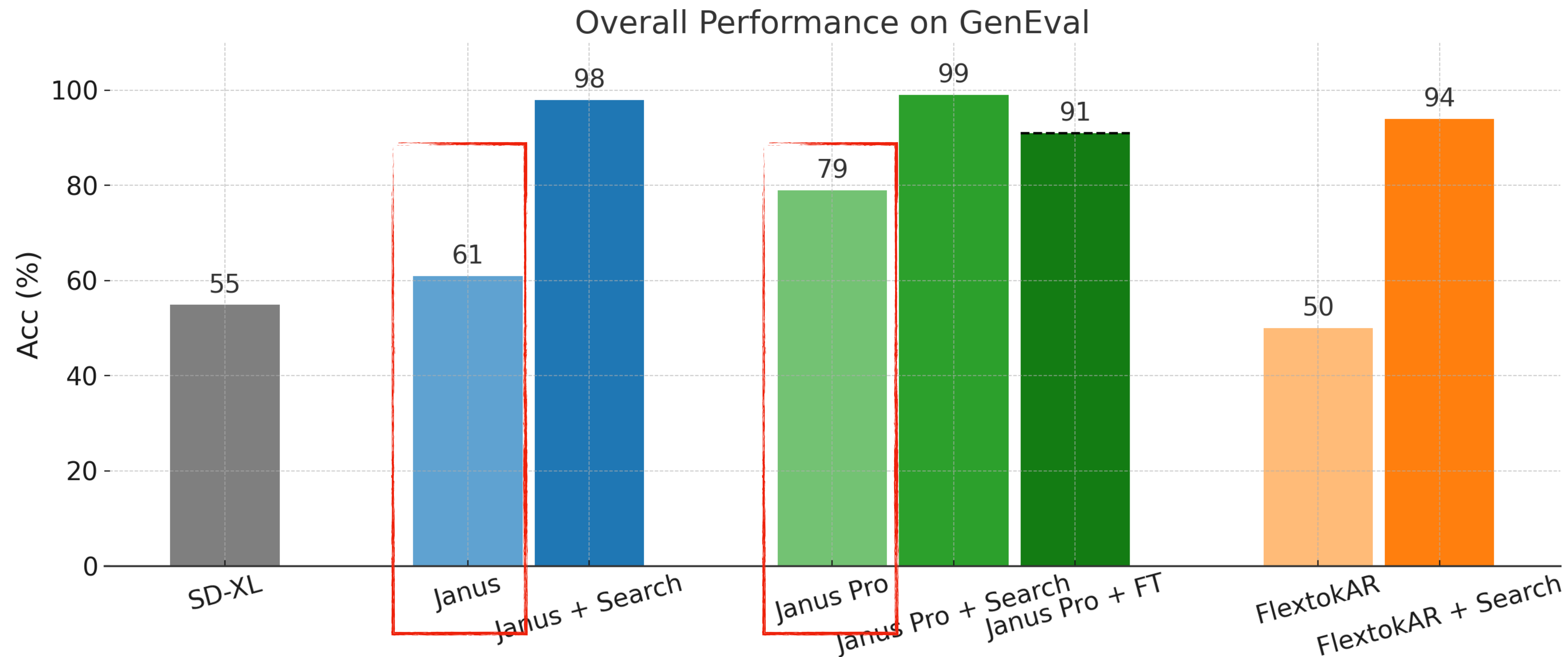
Scaling by Search is Better



Scaling by **searching** is better than scaling over **token number**.

[Bachmann et al. FlexTok: Resampling Images into 1D Token Sequences of Flexible Length, 2025]

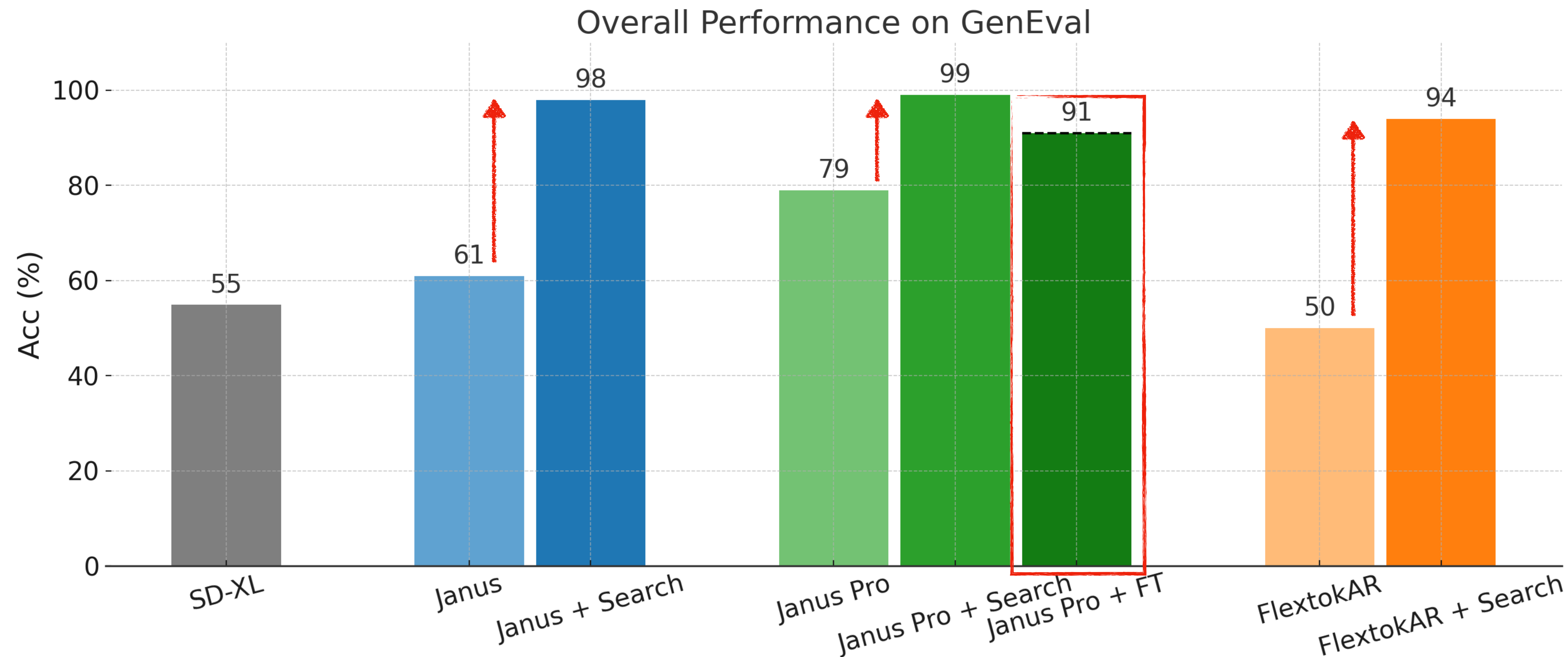
How about other AR models?



[Deepseek AI. Janus: Decoupling Visual Encoding for Unified Multimodal Understanding and Generation, 2024]

[Deepseek AI. Janus-Pro: Unified Multimodal Understanding and Generation with Data and Model Scaling, 2025]

How about other AR models?



[Deepseek AI. Janus: Decoupling Visual Encoding for Unified Multimodal Understanding and Generation, 2024]

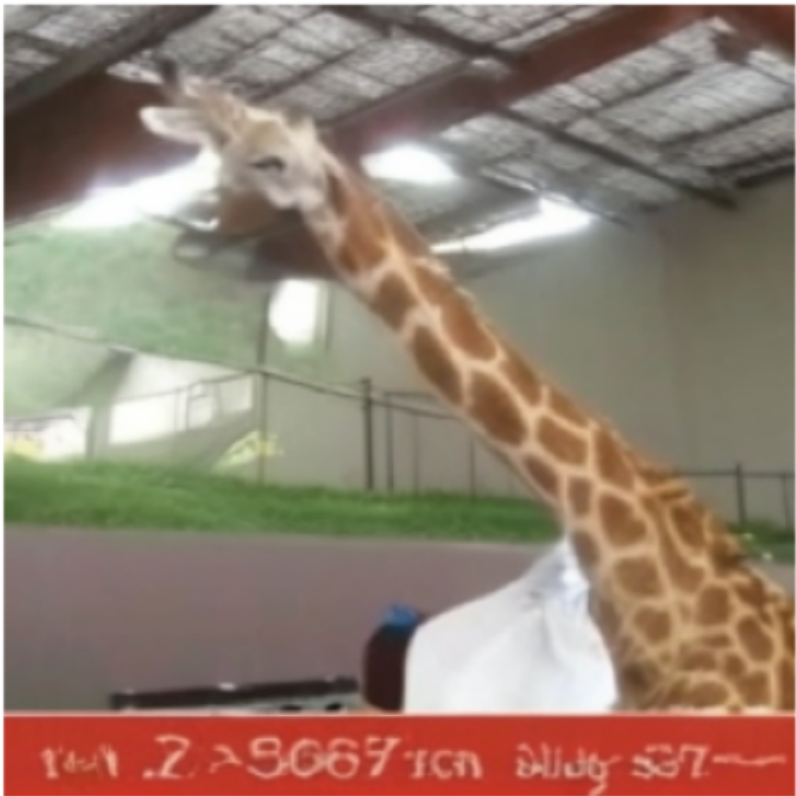
[Deepseek AI. Janus-Pro: Unified Multimodal Understanding and Generation with Data and Model Scaling, 2025]

What else can we achieve by Search?

Flexible control of the generation

An image of a giraffe in a enclosed area is watched by some people

AR



CLIP: 29.344
Aesthetic: 3.703
ImageReward: -0.581

CLIP Verifier



CLIP: 41.438
Aesthetic: 4.977
ImageReward: 1.374

Aesthetic Verifier



CLIP: 35.250
Aesthetic: 6.114
ImageReward: 0.379

Combined Verifier



CLIP: 37.406
Aesthetic: 6.056
ImageReward: 0.808

Image Reward Verifier



CLIP: 41.594
Aesthetic: 5.006
ImageReward: 1.679

An image of A kitchen with a slanted ceiling and skylight.

AR



CLIP: 32.438
Aesthetic: 5.577
ImageReward: 1.092

CLIP Verifier



CLIP: 37.719
Aesthetic: 4.565
ImageReward: 0.721

Aesthetic Verifier



CLIP: 30.297
Aesthetic: 6.316
ImageReward: 0.820

Combined Verifier



CLIP: 34.500
Aesthetic: 6.140
ImageReward: 0.912

Image Reward Verifier

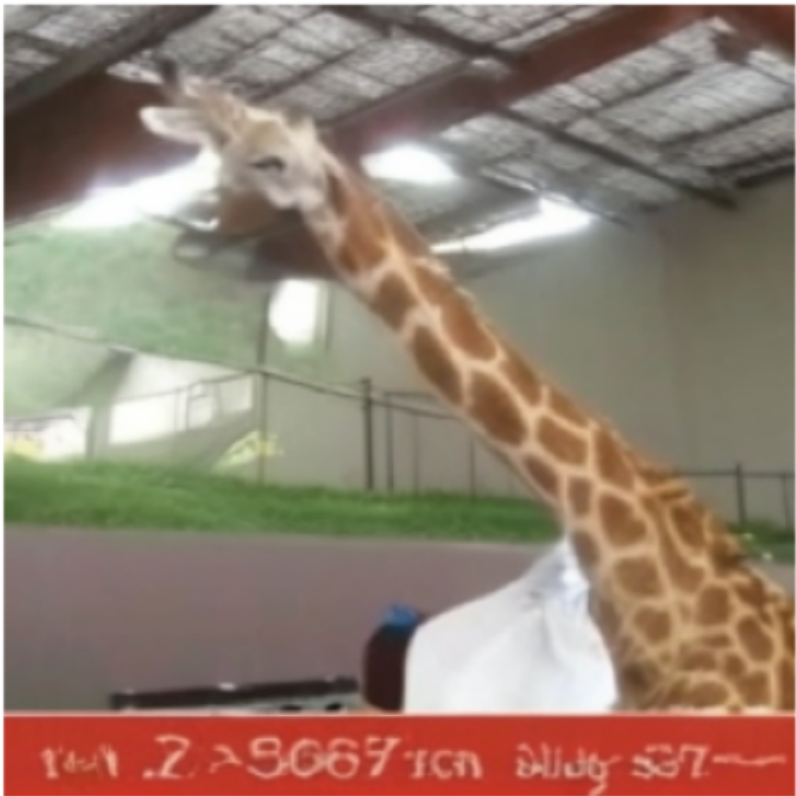


CLIP: 35.219
Aesthetic: 5.302
ImageReward: 1.301

Flexible control of the generation

An image of a giraffe in a enclosed area is watched by some people

AR



CLIP: 29.344
Aesthetic: 3.703
ImageReward: -0.581

CLIP Verifier



CLIP: 41.438
Aesthetic: 4.977
ImageReward: 1.374

Aesthetic Verifier



CLIP: 35.250
Aesthetic: 6.114
ImageReward: 0.379

Combined Verifier



CLIP: 37.406
Aesthetic: 6.056
ImageReward: 0.808

Image Reward Verifier



CLIP: 41.594
Aesthetic: 5.006
ImageReward: 1.679

An image of A kitchen with a slanted ceiling and skylight.

AR



CLIP: 32.438
Aesthetic: 5.577
ImageReward: 1.092

CLIP Verifier



CLIP: 37.719
Aesthetic: 4.565
ImageReward: 0.721

Aesthetic Verifier



CLIP: 30.297
Aesthetic: 6.316
ImageReward: 0.820

Combined Verifier



CLIP: 34.500
Aesthetic: 6.140
ImageReward: 0.912

Image Reward Verifier

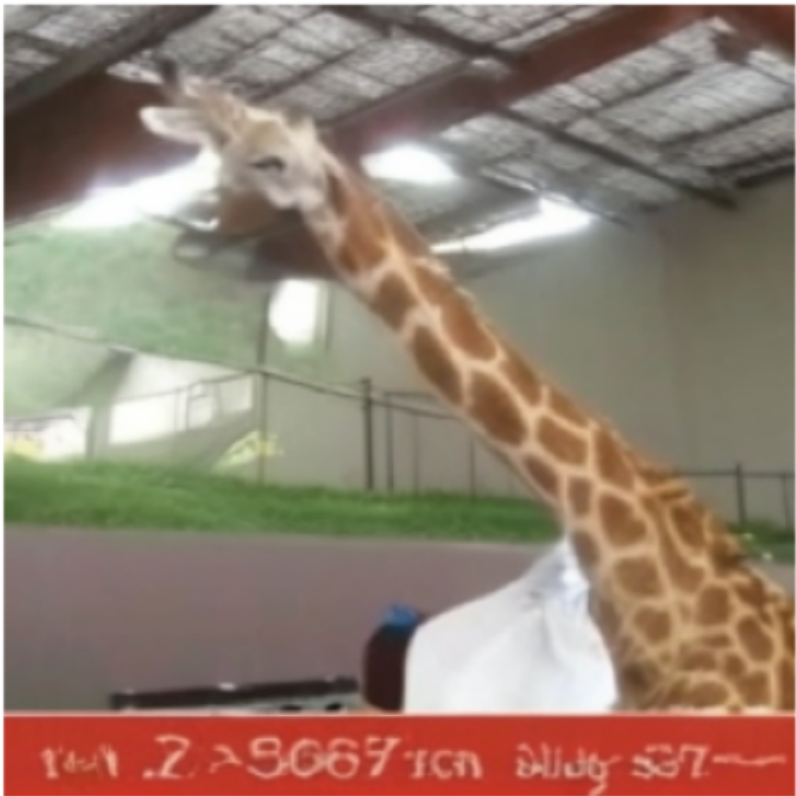


CLIP: 35.219
Aesthetic: 5.302
ImageReward: 1.301

Flexible control of the generation

An image of a giraffe in a enclosed area is watched by some people

AR



CLIP: 29.344
Aesthetic: 3.703
ImageReward: -0.581

CLIP Verifier



CLIP: 41.438
Aesthetic: 4.977
ImageReward: 1.374

Aesthetic Verifier



CLIP: 35.250
Aesthetic: 6.114
ImageReward: 0.379

Combined Verifier



CLIP: 37.406
Aesthetic: 6.056
ImageReward: 0.808

Image Reward Verifier



CLIP: 41.594
Aesthetic: 5.006
ImageReward: 1.679

An image of A kitchen with a slanted ceiling and skylight.

AR



CLIP: 32.438
Aesthetic: 5.577
ImageReward: 1.092

CLIP Verifier



CLIP: 37.719
Aesthetic: 4.565
ImageReward: 0.721

Aesthetic Verifier



CLIP: 30.297
Aesthetic: 6.316
ImageReward: 0.820

Combined Verifier



CLIP: 34.500
Aesthetic: 6.140
ImageReward: 0.912

Image Reward Verifier



CLIP: 35.219
Aesthetic: 5.302
ImageReward: 1.301

Flexible control of the generation

An image of a giraffe in a enclosed area is watched by some people

AR



CLIP: 29.344
Aesthetic: 3.703
ImageReward: -0.581

CLIP Verifier



CLIP: 41.438
Aesthetic: 4.977
ImageReward: 1.374

Aesthetic Verifier



CLIP: 35.250
Aesthetic: 6.114
ImageReward: 0.379

Combined Verifier



CLIP: 37.406
Aesthetic: 6.056
ImageReward: 0.808

Image Reward Verifier



CLIP: 41.594
Aesthetic: 5.006
ImageReward: 1.679

An image of A kitchen with a slanted ceiling and skylight.

AR



CLIP: 32.438
Aesthetic: 5.577
ImageReward: 1.092

CLIP Verifier



CLIP: 37.719
Aesthetic: 4.565
ImageReward: 0.721

Aesthetic Verifier



CLIP: 30.297
Aesthetic: 6.316
ImageReward: 0.820

Combined Verifier



CLIP: 34.500
Aesthetic: 6.140
ImageReward: 0.912

Image Reward Verifier



CLIP: 35.219
Aesthetic: 5.302
ImageReward: 1.301

Flexible control of the generation

Use a segmentation or object detection model as a verifier (Spatial)

a photo of a vase above a fire hydrant



Global Verifier



Spatial Verifier

a photo of a cow right of a laptop



Global Verifier



Spatial Verifier

a photo of a chair left of a zebra



Global Verifier



Spatial Verifier

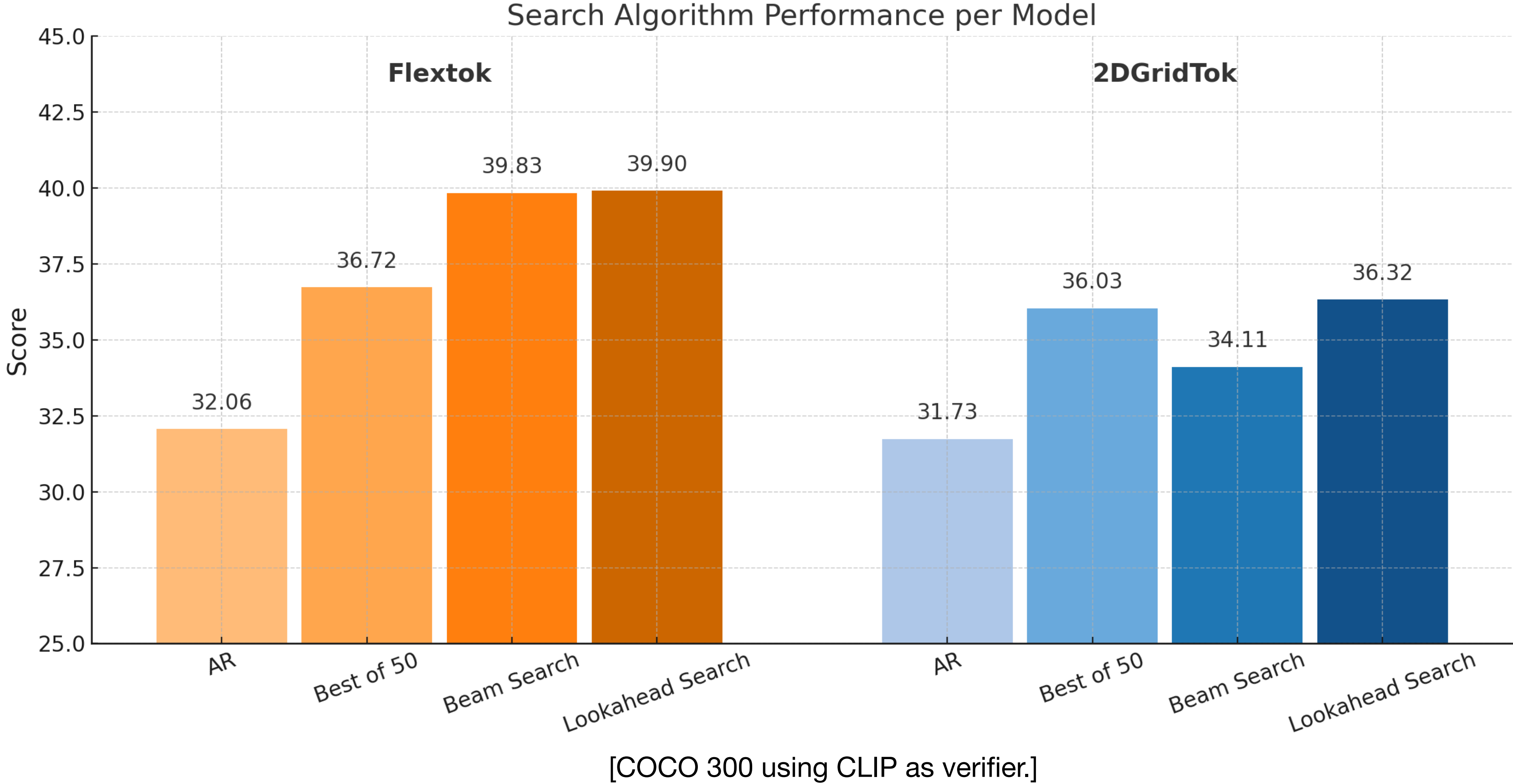
Flexible control of the generation

Use image-image similarity as a verifier.

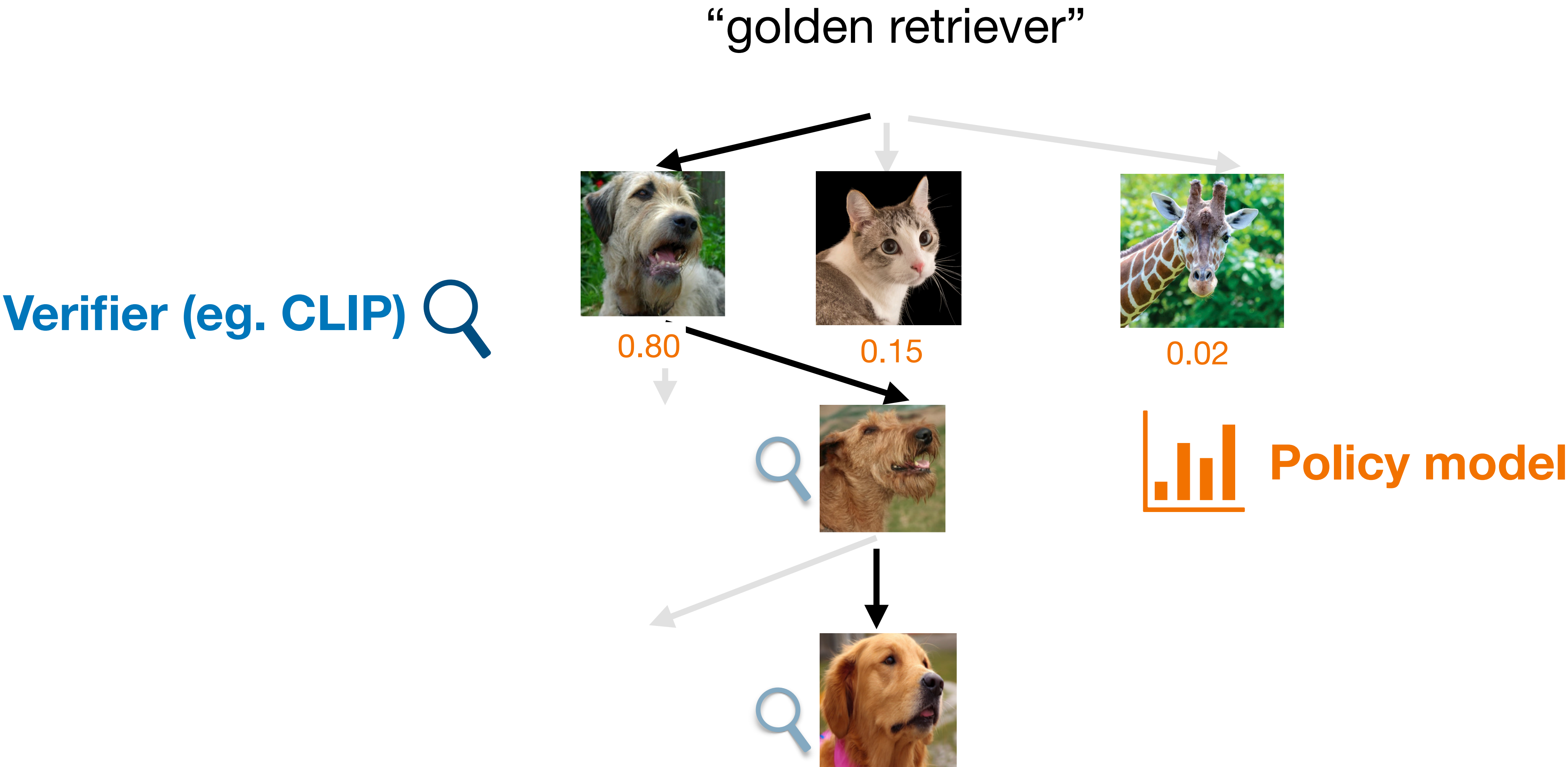
Text control Image control	"A photo of a cat"	"A photo of a cat next to a tree"
		
		

Zero-shot Multimodal Controlling

Search algorithms



Can we do training-free generation only through search?



Can we do training-free generation only through search?

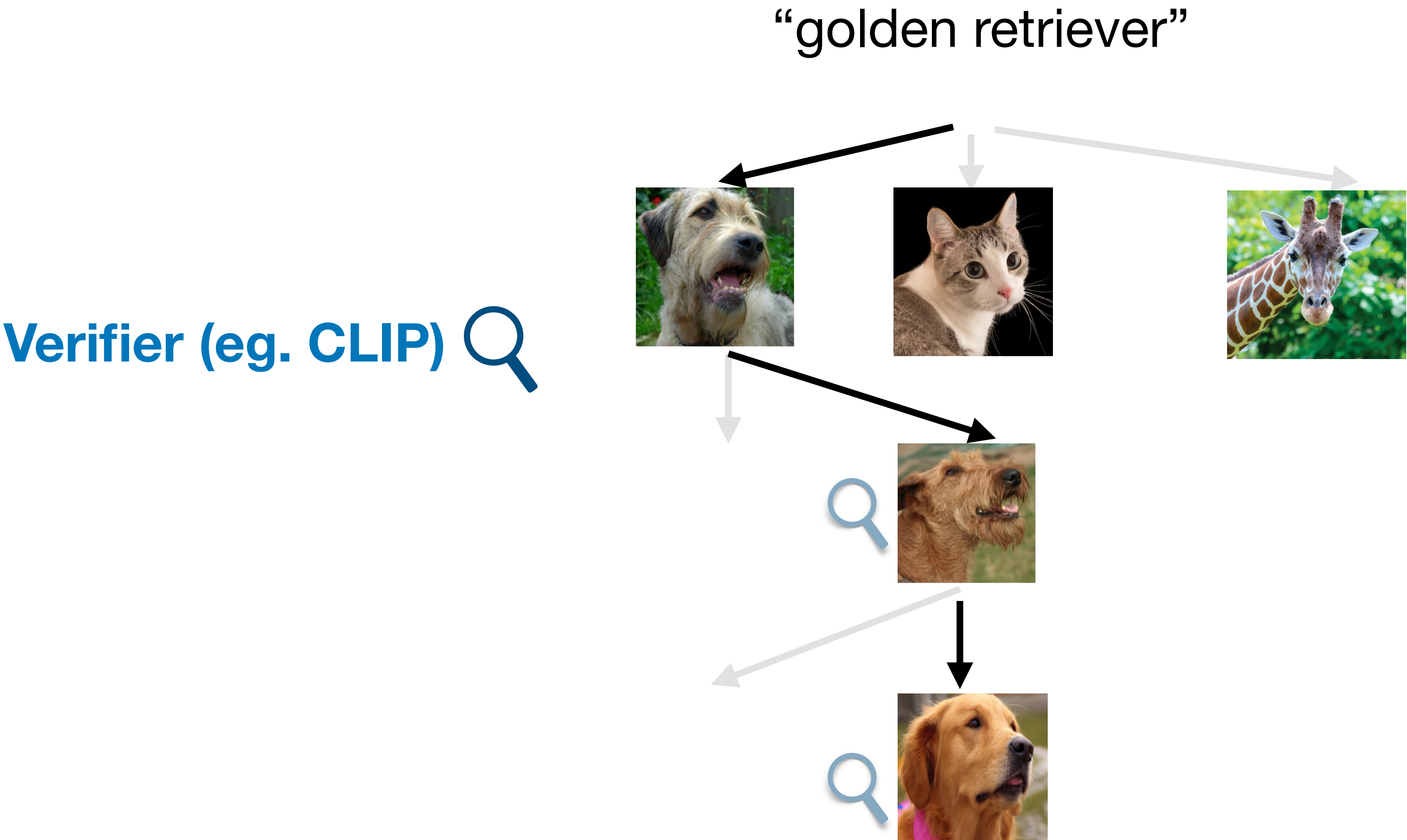
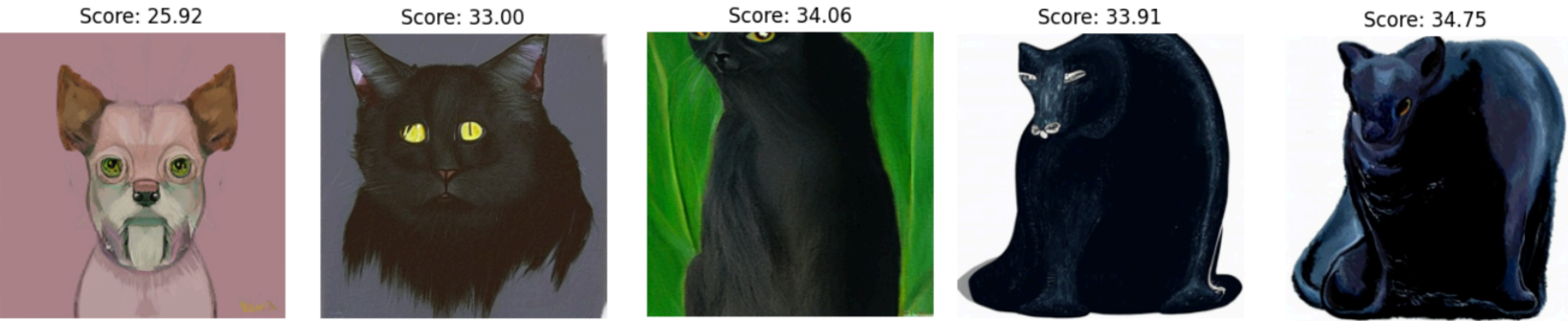


Image generation by pure search

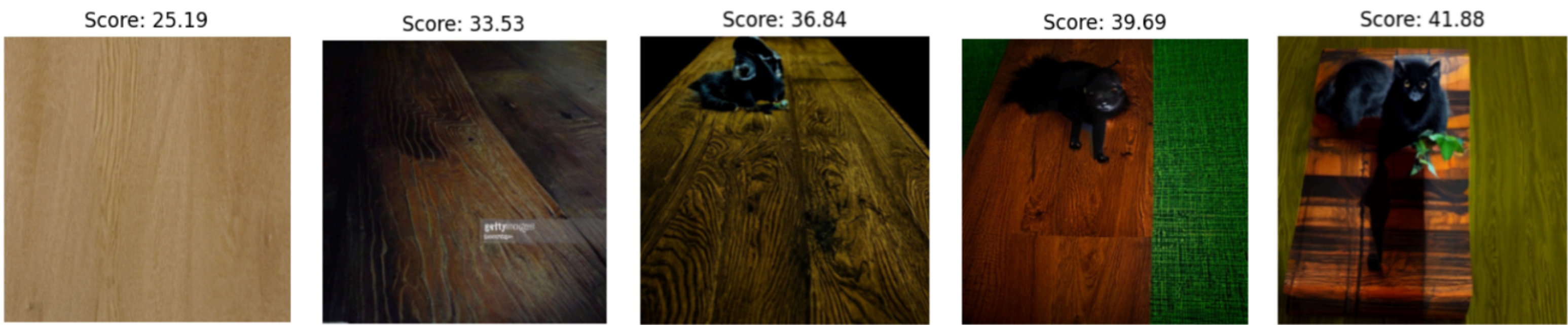
“cat”



“A black cat”



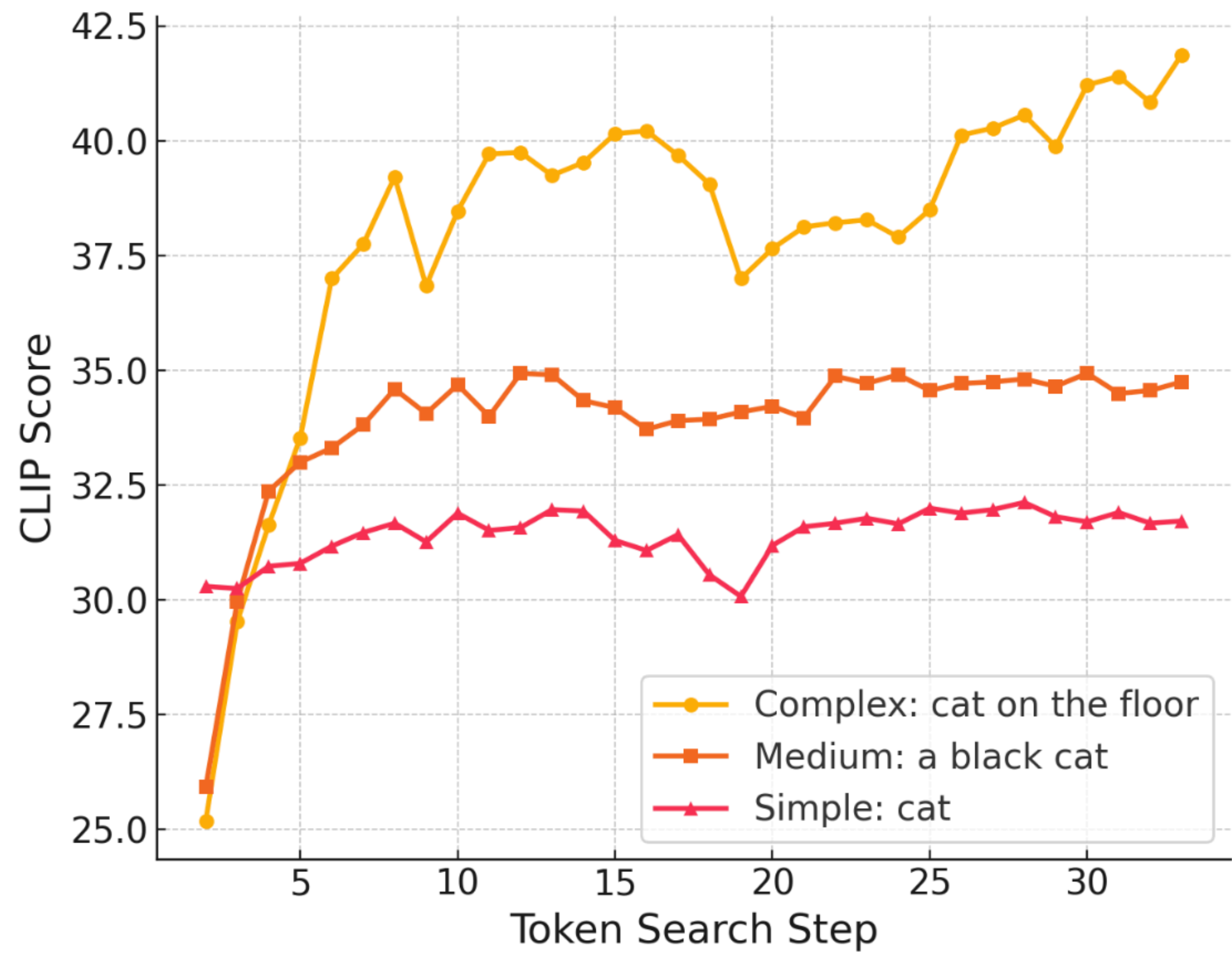
“A detailed photo of a black cat with green eyes sitting on a wooden floor, natural lighting, high quality”



Search Token Step

1 4 8 16 32

Image generation by pure search



More complex prompts -> Larger improvement by searching!

Next

Token Structure

2D token; 2D ordered Token;
1D token; 1D ordered Token, ...

Verifier (eg. CLIP)

Aesthetic Score

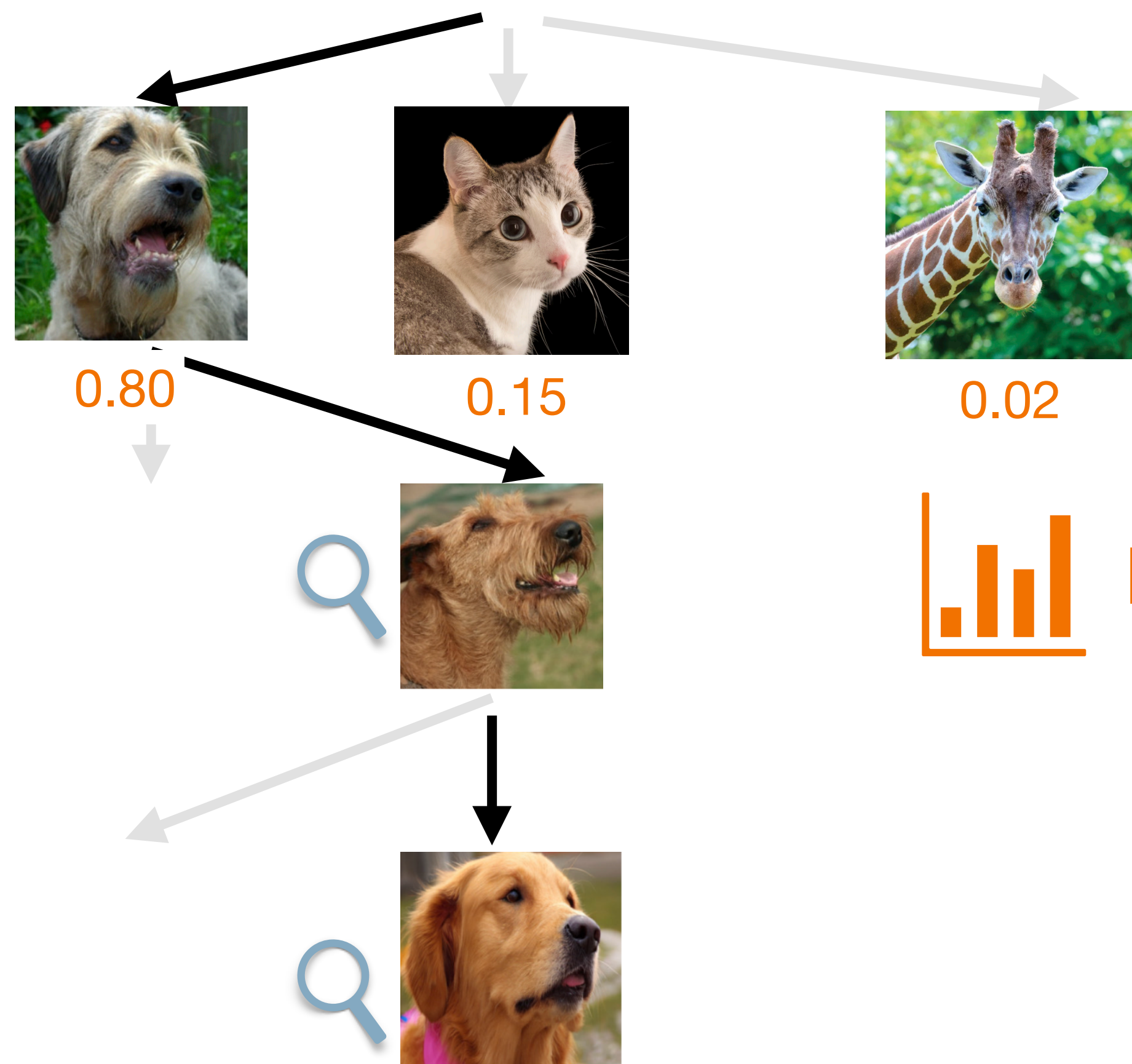
ImageReward

Spatial verifier

Image-image similarity

VLMs

“golden retriever”



Search Algorithm

Beam Search

Lookahead Search

MCTS,

....



Policy model

Conditional AR,

Unconditional AR,

Randomized AR, ...

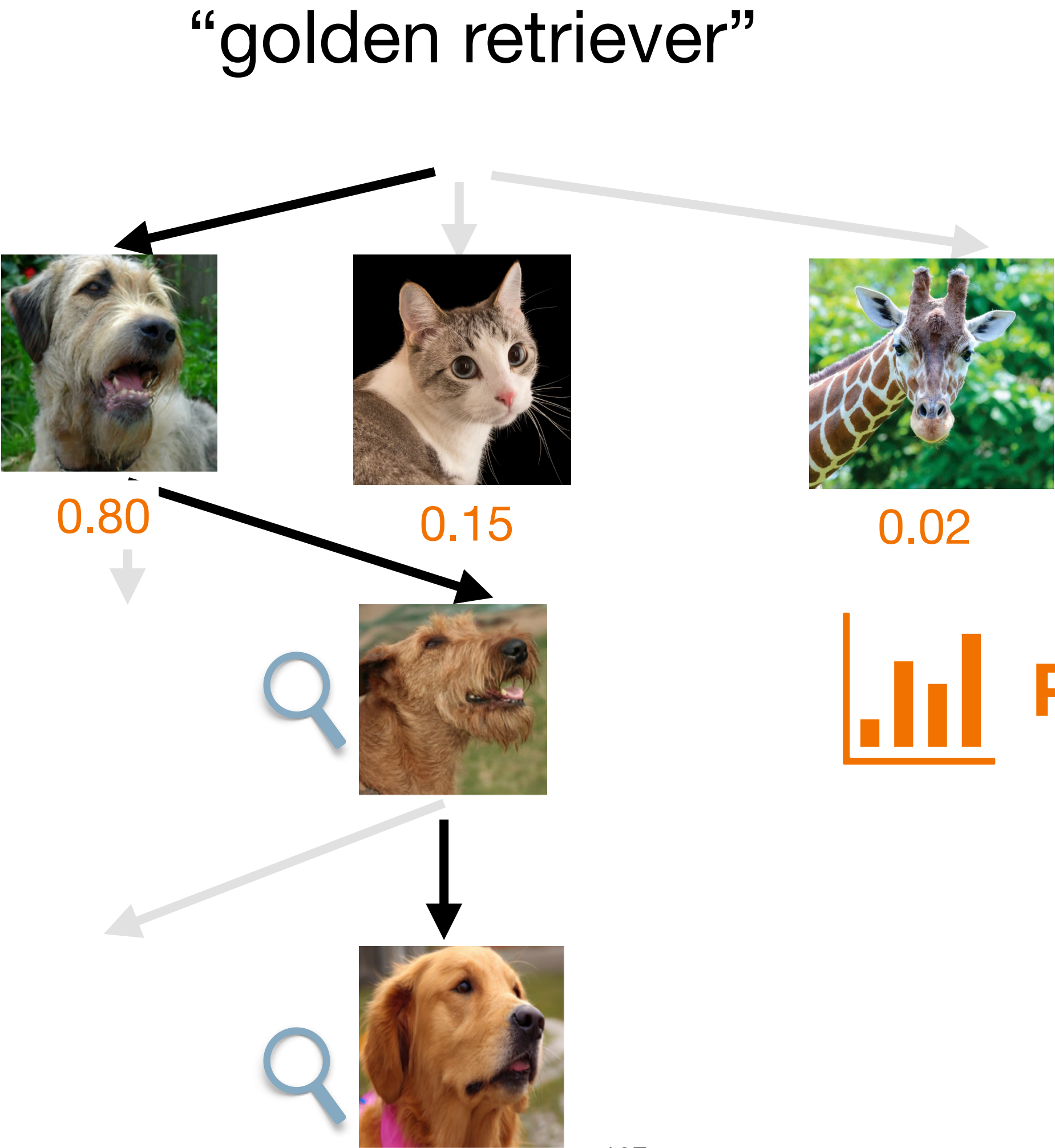
Next

Token Structure

2D token; 2D ordered Token;
1D token; 1D ordered Token, ...

Verifier (eg. CLIP)

- Aesthetic Score
- ImageReward
- Spatial verifier
- Image-image similarity
- VLMs



Search Algorithm

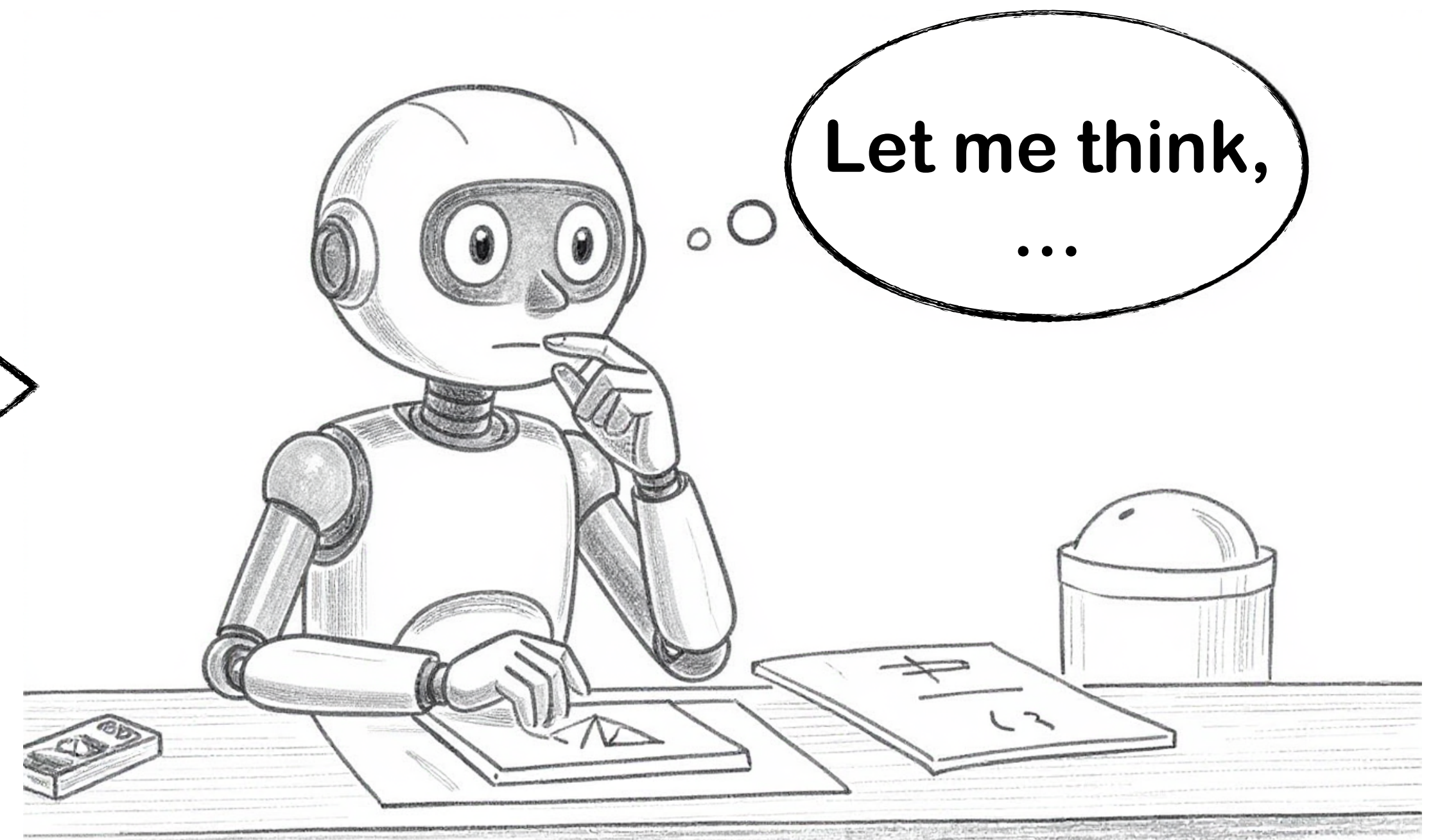
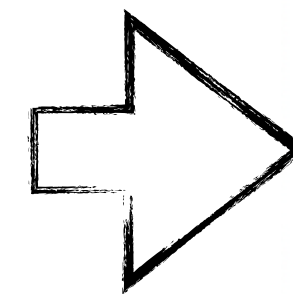
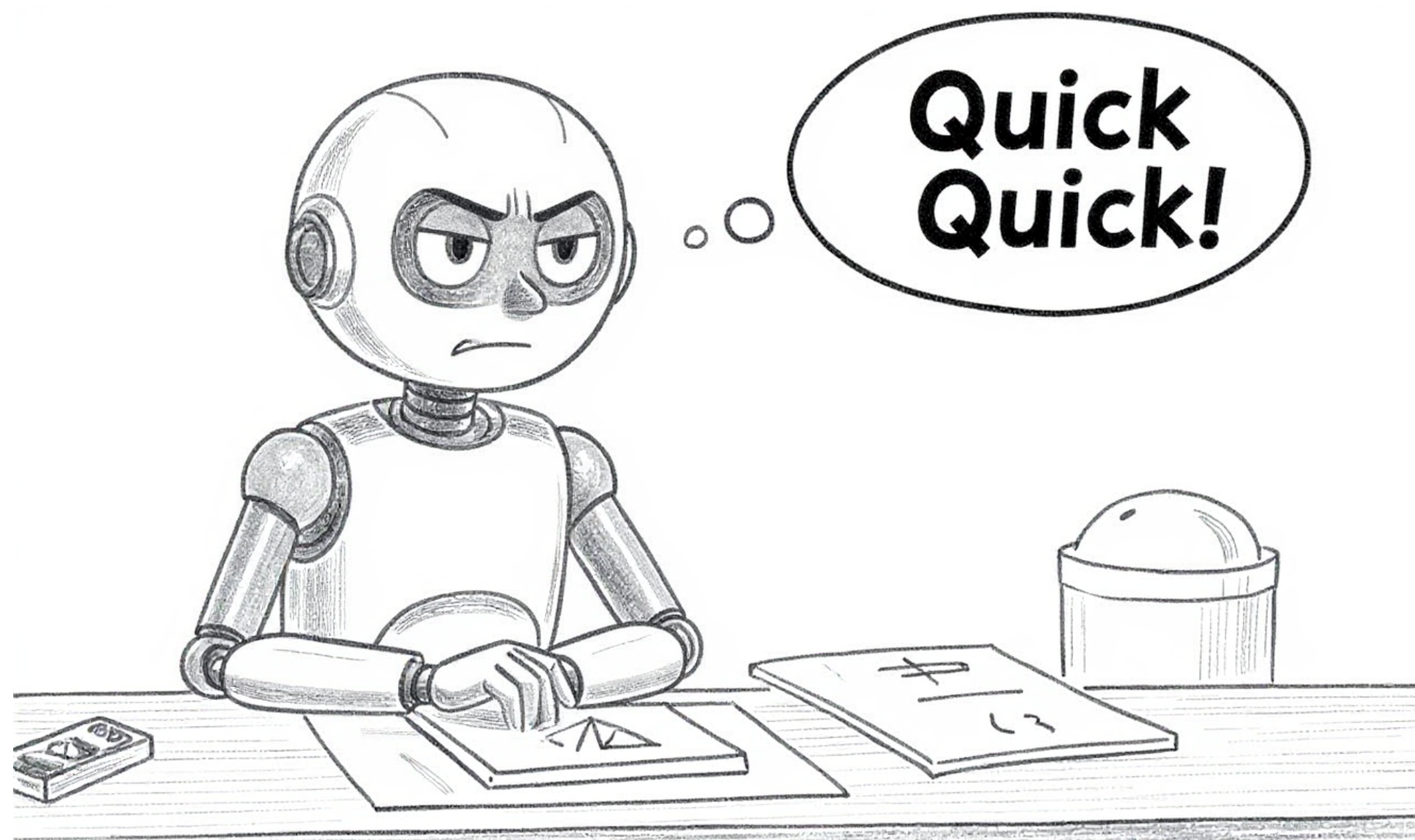
Beam Search
Lookahead Search
MCTS,
....

Policy model

Conditional AR,
Unconditional AR,
Randomized AR, ...

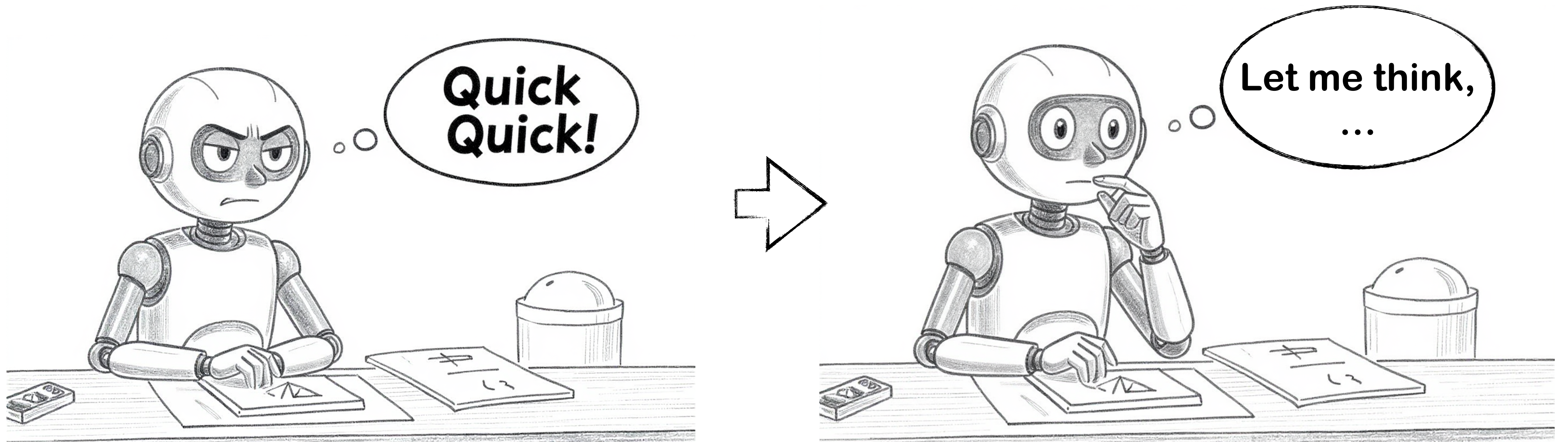
Future Work

- Can we further teach a vision model ‘how to search’ during training?



Future Work

- Can we further teach a vision model ‘how to search’ during training?
- Search for different modalities.



Future Work

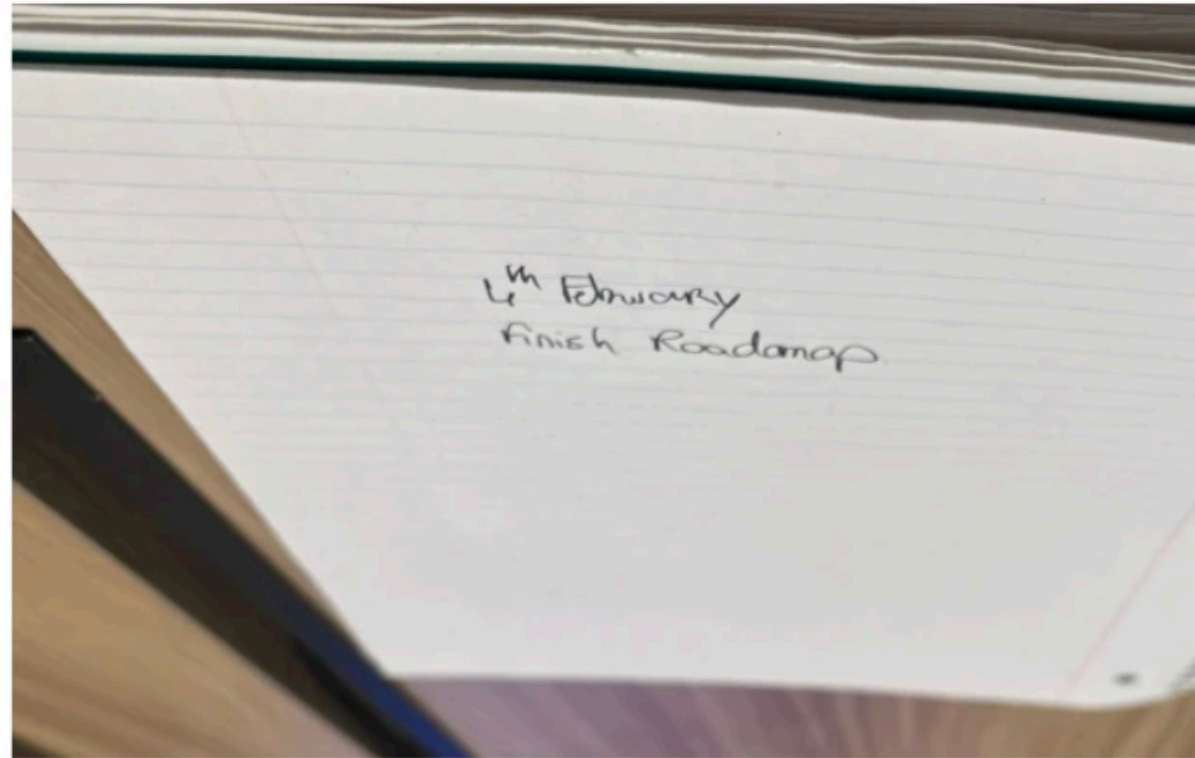
- Can we further teach a vision model ‘how to search’ during training?
- Search for different modalities.
- Test-time scaling and search in **visual perception**.

What is written on the notebook?



[OpenAI, Thinking with Image, 2025.]

- The user shared an image found at /mnt/data/own_notebook.jpg and asked me to read the text on it. I'll load the image and zoom in to see the writing clearly. It turns out the text is upside down, so I'll rotate it for easy reading. Afterward, I'll crop the lower part of the image to focus on the relevant area and rotate the cropped section to ensure the text is right side up.



Done

It says: "4th February – finish roadmap."

Thanks for listening!