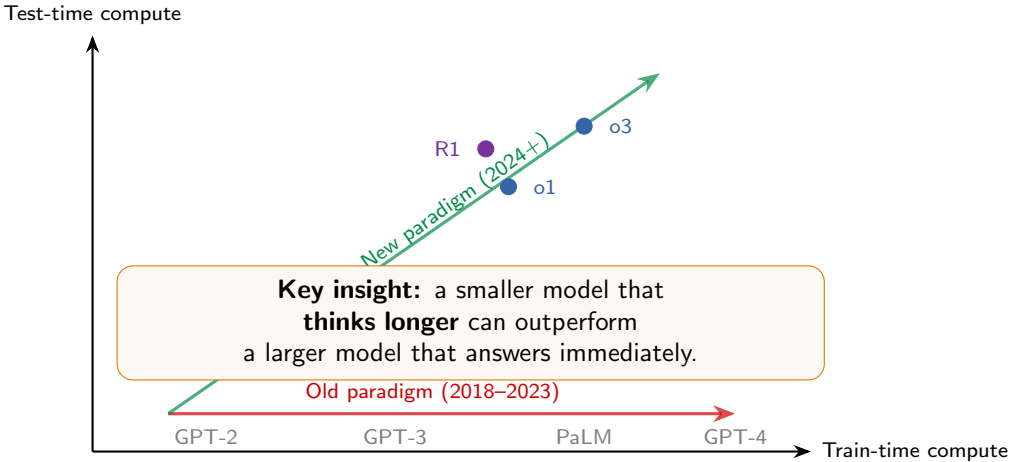
# Reasoning & Test-Time Compute

Chain-of-Thought · Search · Verification · Scaling Inference

# The two scaling axes



# Chain-of-Thought prompting

**Standard prompting**

Q: Roger has 5 balls.
He buys 2 cans of 3.
How many balls?

↓

**A: 11**

**Chain-of-Thought prompting**

Q: Roger has 5 balls.
He buys 2 cans of 3.
How many balls?

↓

Step 1: 2 cans $\times$ 3 balls $=$ 6 balls bought
Step 2: 5 original $+$ 6 new $=$ **11 balls**

**Few-shot CoT** (Wei et al., NeurIPS 2022):
provide exemplars with reasoning steps

**Zero-shot CoT** (Kojima et al., 2022): just add "*Let's think step by step*"

Emergent ability: benefits appear only at $\sim$100B+ parameters

# Why Chain-of-Thought works

**Decomposition**

Hard problem → sequence of easy sub-problems

Each step is within the model's ability

**Extra compute**

More output tokens = more FLOPs allocated per problem

Harder problems get more computation

**System 1 vs 2**

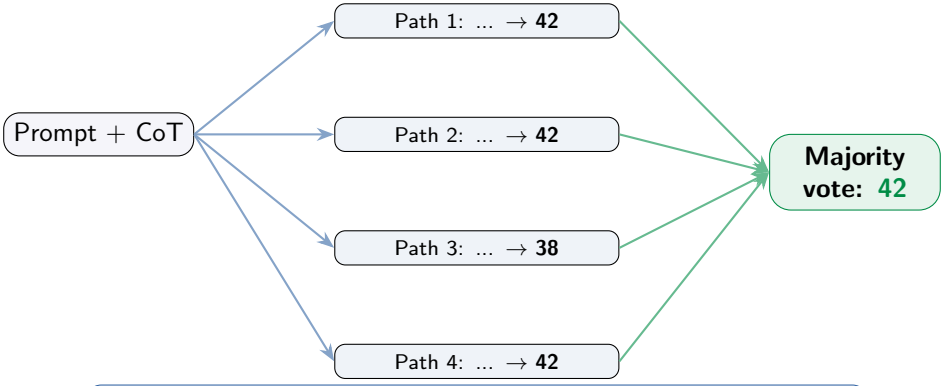
Direct answer = fast, intuitive (System 1)

Step-by-step = slow, deliberate (System 2)

CoT is the **simplest form** of test-time compute scaling:
generate more tokens ⇒ allocate more computation ⇒ better answers.

Caveat: some researchers argue CoT reflects structured inductive bias from training data, not true reasoning.

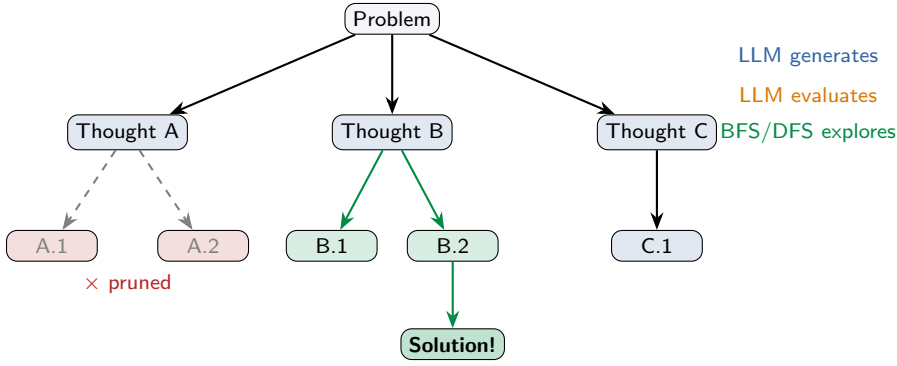
# Self-Consistency




Wang et al., ICLR 2023 · **No training needed** — purely a decoding strategy

GSM8K **+17.9%** SVAMP **+11%** AQuA **+12.2%** StrategyQA **+6.4%**

# Tree of Thoughts

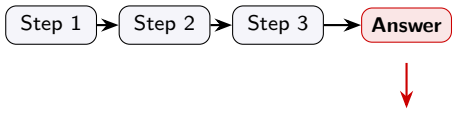



Yao et al., NeurIPS 2023 · **Game of 24:** CoT **4%** → ToT **74%**

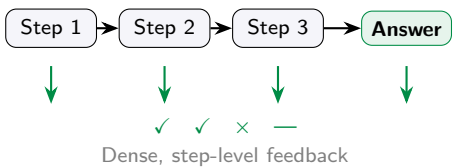
Bridges LLM generation with classical AI search (branching + backtracking)

# Process vs. Outcome Reward Models

**Outcome RM (ORM)**




**Process RM (PRM)**




Lightman et al., ICLR 2024
"Let's Verify Step by Step"

PRM **significantly outperforms** ORM on MATH dataset

**PRM800K:** 800K step-level human labels released

PRMs enable **fine-grained search**: score partial solutions to guide beam search / MCTS at inference

# Search at inference time

**Best-of-N**

Generate $N$ candidates

Score each with a verifier (ORM/PRM)

Return the best one

Simple but effective

Cost: $O(N)$ samples

**Beam search**

Expand promising partial solutions

PRM scores each step

Prune low-scoring branches

Finer-grained than Best-of-N

**MCTS**

Monte Carlo Tree Search over reasoning steps

Selection $\rightarrow$ expansion $\rightarrow$ rollout $\rightarrow$ backprop

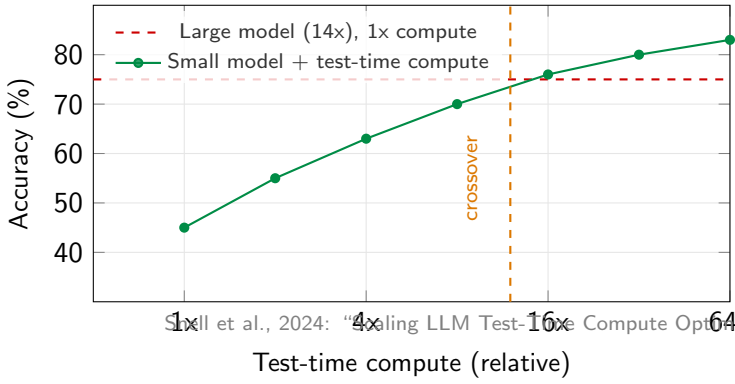
7B + MCTS: **83.4%**

math (beats 72B)

All methods share one principle: **trade more compute for better answers.**

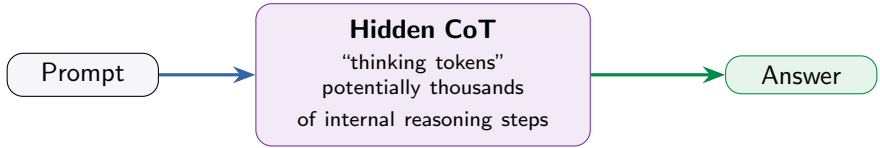
The key enabler is a **verifier** (PRM or self-evaluation) to distinguish good from bad reasoning.

# Test-time compute scaling laws




Snell et al., 2024: "Scaling LLM Test-Time Compute Optimally"

# OpenAI o1: reasoning via reinforcement learning




Trained with **large-scale RL** to produce internal reasoning chains.

Through RL, the model learns to: refine strategies, recognize mistakes, break down hard steps, and try alternative approaches.

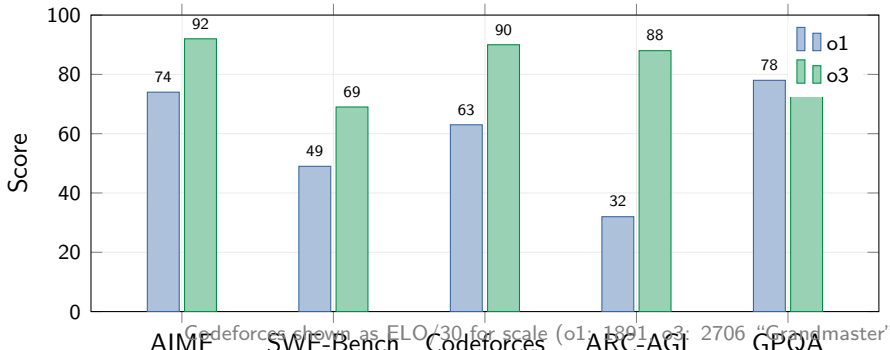
**MATH:** 94.8% (consensus@64) **AIME 2024:** 83%

**GSM8K:** ∼ceiling

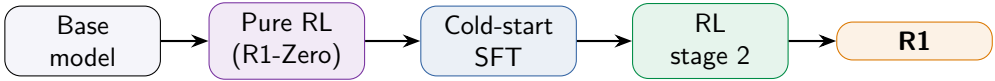
Hidden CoT is not shown to users — only a model-generated summary is displayed.

OpenAI, "Learning to Reason with LLMs," September 2024

# OpenAI o3: scaling inference further



# DeepSeek-R1: open-source reasoning




**R1-Zero:** reasoning emerges from **pure RL** (GRPO), no SFT needed!

Emergent behaviors:
self-verification, "aha moments,"
extended chain-of-thought

**Matches o1** at ∼**95% less cost**

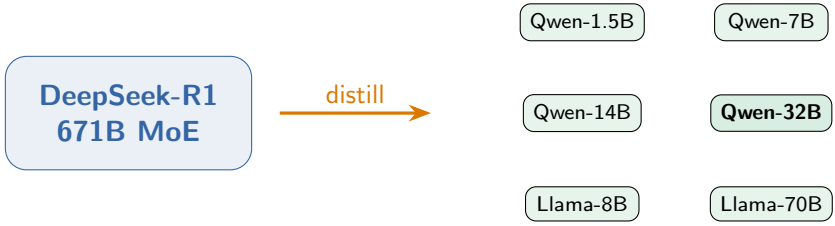
MATH: **97.3%** AIME: **79.8%**

Fully open-source:
model, data, and code

**The "aha moment":** during RL training, the model spontaneously learned to step back, spot its own mistakes, and try alternative approaches.

DeepSeek-AI, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via RL," January 2025

# Distillation of reasoning




**Key finding:** distilling reasoning from a large model into small models works **better** than applying RL directly to small models.

**R1-Distill-Qwen-32B** outperforms **o1-mini** across benchmarks.

Reasoning capability is transferable — you don't need to run RL on every model.

# Budget forcing & adaptive compute

**Truncation**

Model exceeds token budget ⇒ forcefully end thinking

Prevents overthinking on easy questions

**Extension**

Model tries to stop early ⇒ append "**Wait**" token

Forces double-checking, often fixes errors!

**s1** (Muennighoff et al., Jan 2025): trained on just **1,000 questions**

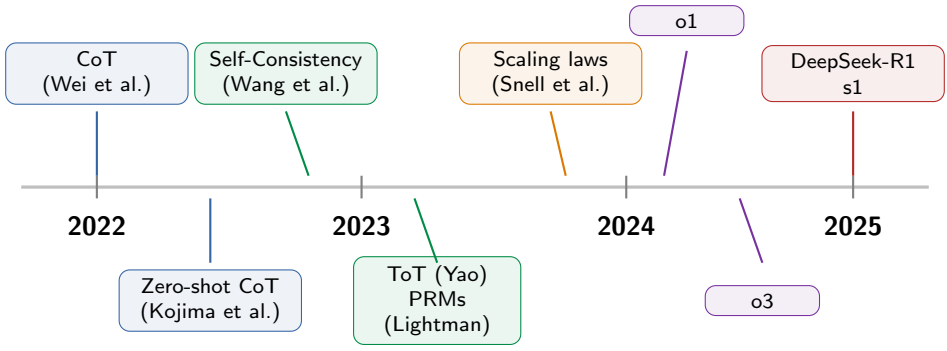
Beats **o1-preview** by up to **27%** on competition math (MATH, AIME)

**The problem:** reasoning models **overthink** easy queries and **underthink** hard ones.

**Adaptive Budget Forcing:** monitor confidence/entropy in real time, allocate compute dynamically.

Claude 3.7: user-controllable "thinking token budget" · o1: multiple compute modes

# The reasoning landscape



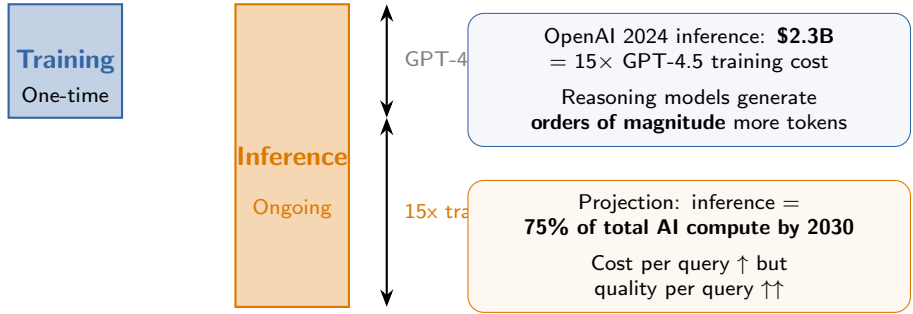

From "prompt engineering" (2022) to "inference-time scaling" (2024+) — in just 3 years.

# Key reasoning benchmarks

| Benchmark | What it tests | Scale | Frontier SOTA |
|---|---|---|---|
| **GSM8K** | Grade-school math | 1K test | Saturated (>95%) |
| **MATH** | Competition math | 5K test | R1: **97.3%** |
| **AIME** | Hard competition (15 Q/yr) | 15/year | o3: **91.6%** |
| **ARC-AGI** | Fluid intelligence | ∼400 tasks | o3: **88%** |
| **Codeforces** | Competitive programming | ELO rating | o3: **2706** (GM) |
| **SWE-Bench** | Real GitHub issues | 2.3K tasks | o3: **69.1%** |
| **GPQA** | PhD-level science | 198 Q | o3: **87.7%** |

**Benchmark saturation** is a growing problem: GSM8K went from "hard" (2021) to "solved" (2024). New benchmarks (ARC-AGI-2, FrontierMath) keep raising the bar.

# The economics of thinking




The economic balance has shifted: **inference cost now dominates training cost.**

This changes how we think about model deployment, pricing, and hardware planning.

# Practical guide

**Simple factual Q?**
→ Standard LLM
(no extra reasoning)

**Multi-step reasoning?**
→ CoT + self-consistency
(cheap, effective)

**Math / code contest?**
→ Reasoning model
(o1/o3/R1) or
MCTS+PRM

**Latency-sensitive?**
→ Budget forcing
(truncate/extend
adaptively)

**Open-source needed?**
→ DeepSeek-R1 or
distilled variants

**Want best quality?**
→ Best-of-N + PRM
(scale compute
at inference)

**Rule of thumb:** start with CoT/self-consistency (free, universal).
Upgrade to reasoning models or search only when the task demands it.

# Open questions

**Efficiency**

Can we get reasoning quality **without** the overhead of thousands of hidden tokens?

**Allocation**

How to optimally decide **how much** to think for each query?

**Scaling**

Will reasoning ability keep improving with more test-time compute, or plateau?

**Generality**

Is RL-discovered reasoning truly general, or specialized to math/code domains?

# Questions?

Next: Long Context & Efficient Attention