deepseek-r1

深度求索

“A Profound Quest” in Reasoning AI

Feb 25, Ozgur Guler

Agenda

1. Reasoning - generalisation - RL
2. Why is deepseek-r1 a breakthrough?
3. Future potential for AI Automation
4. deepseek-r1 on Azure
5. deepseek-r1 vs OpenAI o1/o3
6. Reasoning SLM’s - deepseek-distilled
7. AI Safety & deepseek-r1

1. RL for Generalisation

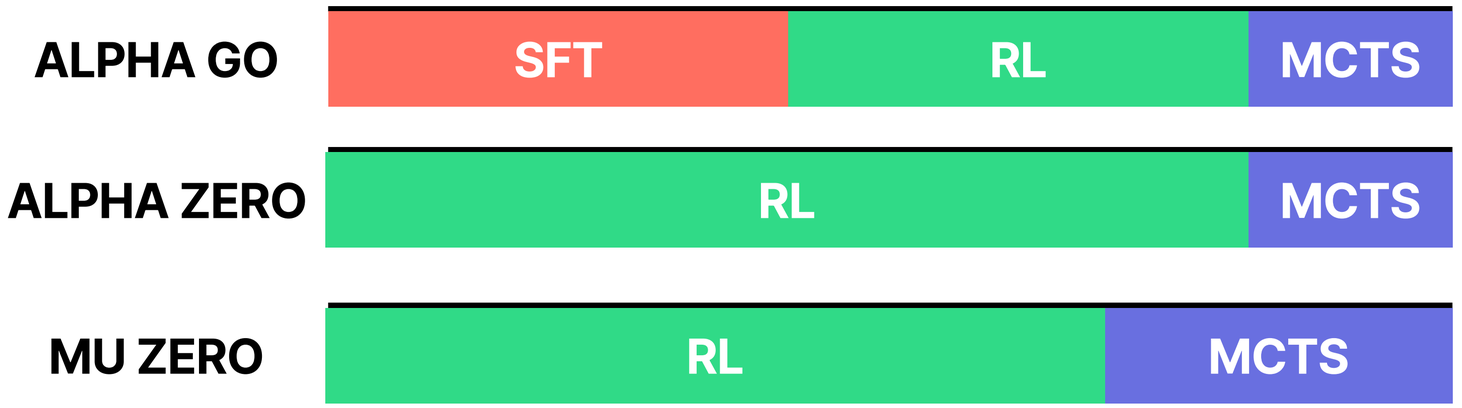
David Silver's RLC 2024 Keynote

August, 2024

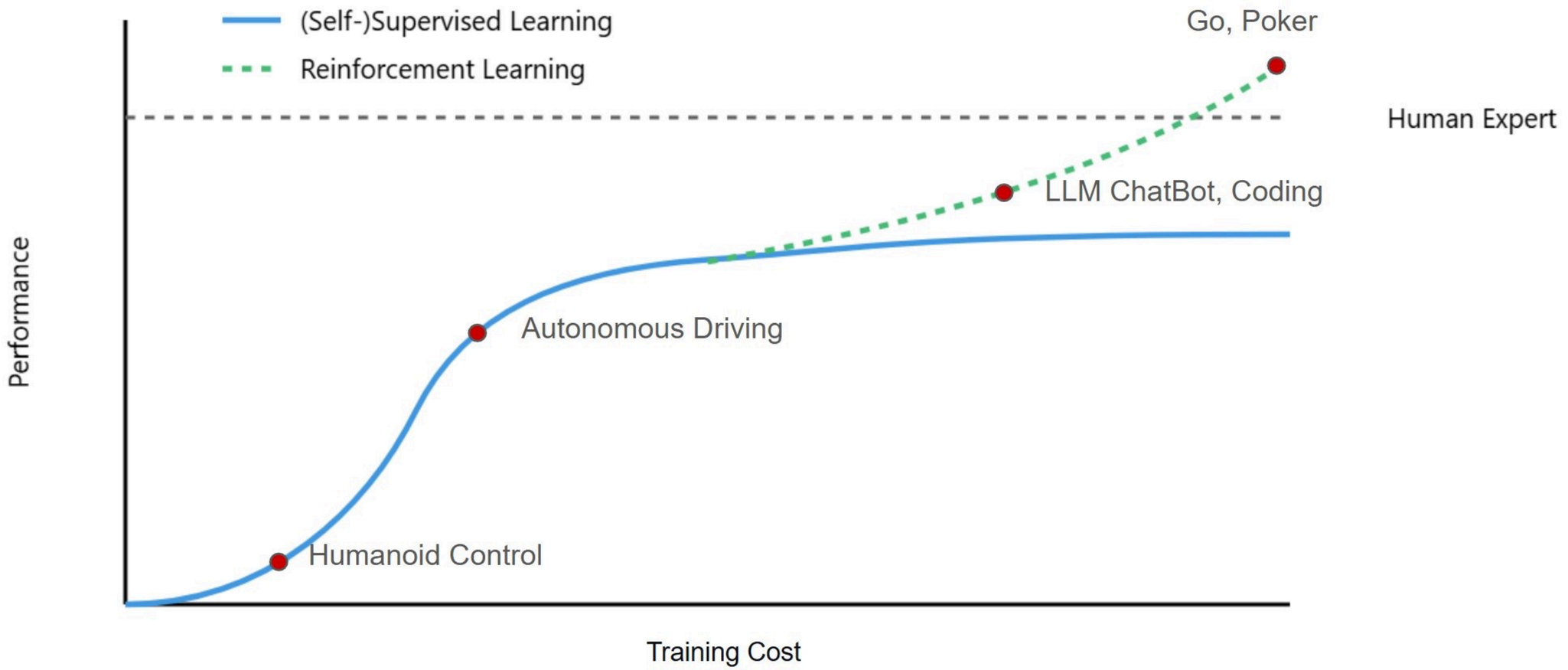
Google DeepMind
London
davidsilver@google.com

Professor of Computer Science
University College London
d.silver@cs.ucl.ac.uk

"Breaking through LLM Valley to AGI requires RL's self-improving intelligence, we must harness RL's power of self-improvement—learning beyond human limits through interaction..."



Pure-RL training is difficult at real world scenarios



Yann Le CUN's CHERRY CAKE

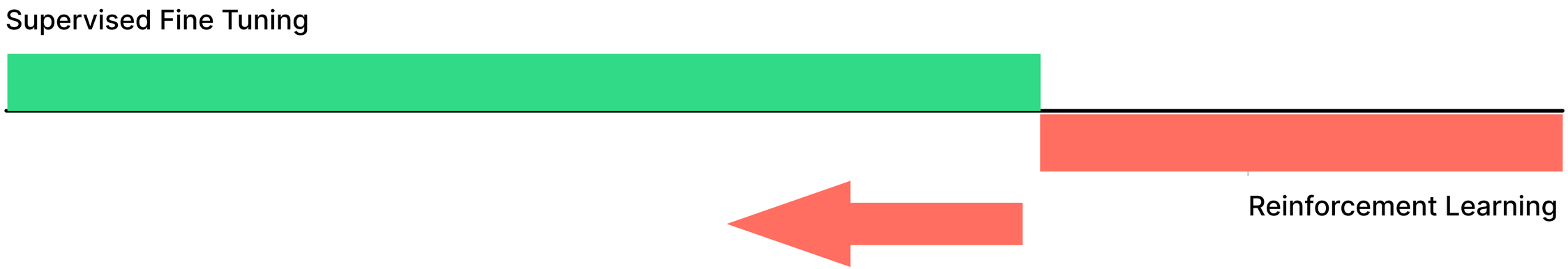
RLHF

SFT

SSL

Reasoning Requires Generalization—Reinforcement Learning Bridges the Gap
OOD - out of distribution - generalisation & output diversity

SFT vs RL “SFT memorizes, RL Generalizes” Arxiv 2501.17161
SFT vs RLHF “Understanding The Effects of RLHF on LLM Generalisation & Diversity” ArXiv:2310.06452v3



2. Why is deepseek-r1 a breakthrough?

Deepseek - Reinforcement Learning for Scalable Reasoning

How is deepseek-r1 different?

1. Pure RL Approach (r1-zero)

- Learns like AlphaGo, by interacting with its own outputs
- Pure RL training (small SFT with cold-start data to improve readability and language mixing)
- Self-Evolution & smooth learning with GRPO

2. Emergence with Scale

- Self-verification, reflection, generating long CoT's & backtracking, allocating more time to a problem, emerge with increased test-time compute
- No PRM

3. Scalable as no Reward Models or Humans

- No trained reward models as in RLHF or RLAIF
- No reliance on humans
- Rule based reward system based on correctness (GRPO)
- Cheaper & computationally efficient as directly optimizes the responses relative to each other...

4. No inference-time scaling

- Even if model backtracks it is a single forward pass.
- We do not introduce loops or search during inference...

Generator / Verifier Gap

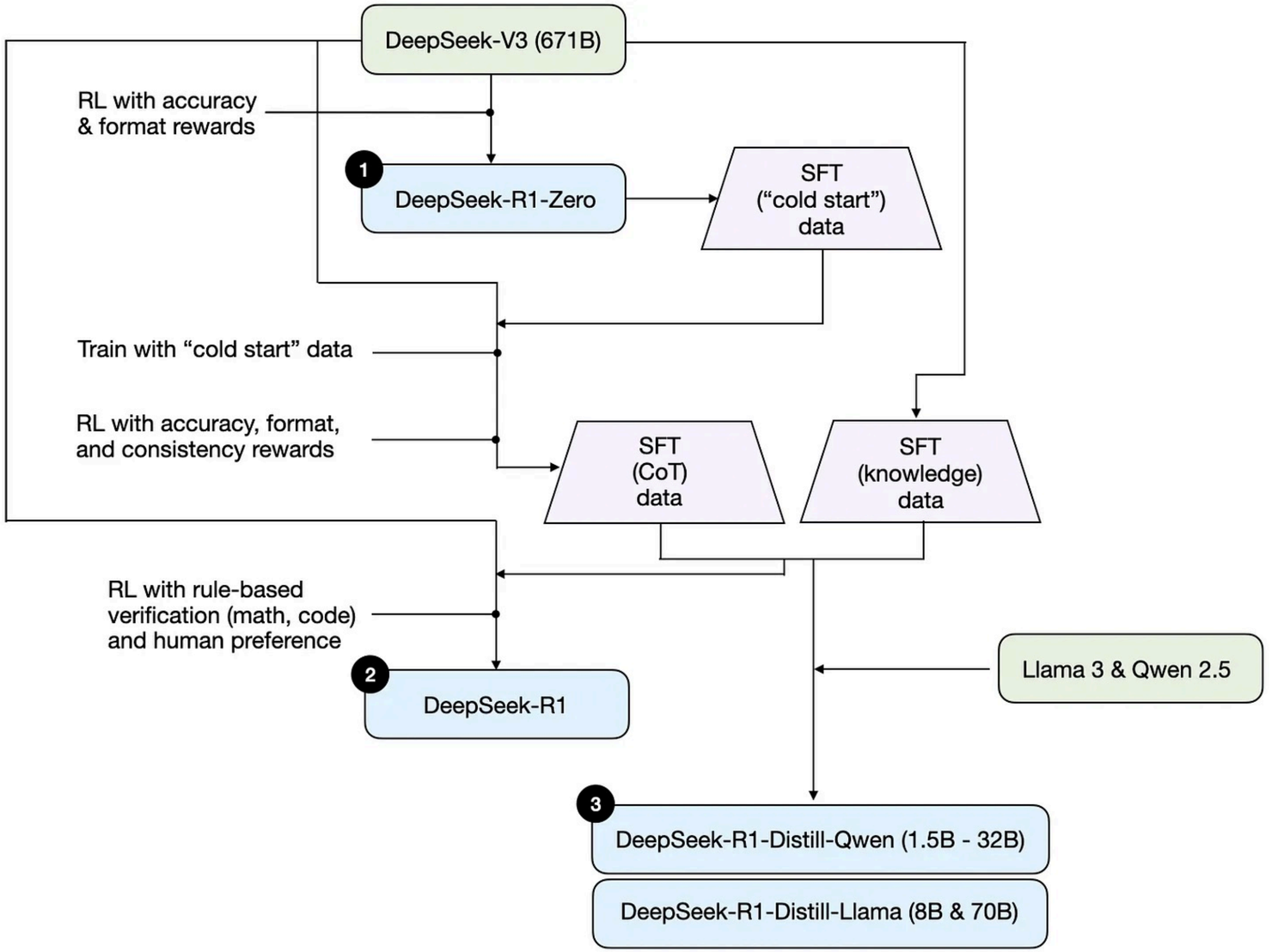
Generator / Verifier Gap
For some problems, verifying a good solution is easier than generating one

A. Search Process

- The AI can generate multiple candidate solutions
- Quickly verify each solution's quality
- Use this rapid verification to guide the search toward better solutions

B. Iterative Improvement

- Start with an initial solution
- Make small modifications
- Verify if modifications improve the solution
- Keep the better versions



- deepseek-zero
- deepseek-r1
- depseek-distill

What didn’t work for Reasoning Training?

SFT

Provide CoT traces for known for problems with known solutions

PRM

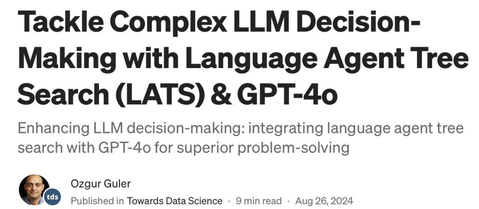
Process supervision with reward models…e.g. Jigsaw

Pure-RL

Reward hacking & instability

Search

MCTS, beam search LATS
Computationally expensive



SFT PPO - RLHF PRM **RL**

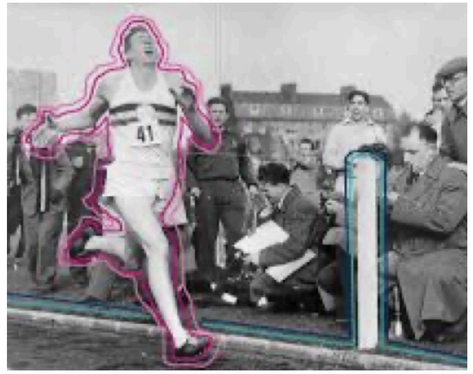
Why did it work this time?

Very good base model deepseek-v3

Add a quick description of each thing, with enough context to understand what's up.
Not Flan-t5?

Finding the right pipeline is difficult...

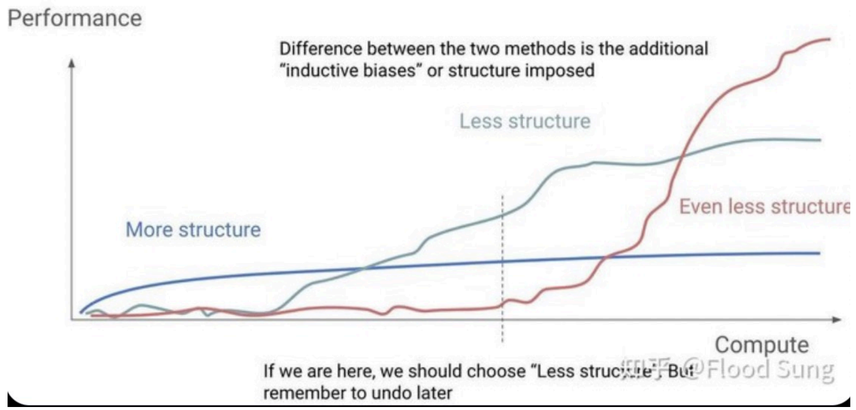
This is where Deepseek's Engineering shines...



Proof of existence - 4 minute-barrier of AI

o1 report explicitly says it was trained Large Scale RL..
Kimi k1.5

kimi k1.5



Parables on the Power of Planning in AI: From Poker to Diplomacy: Noam Brown

Test-time search but not MCTS...Search being baked into the model...

Hyung Won Chung's "Don't Teach, Incentivize"

Do RL with exact rewards! Don't be constrained by Reward Models.

Recreate Efforts

HG open-r1

Kimi k1.5

AI21 Labs Tulu3

Only GRPO trials

Berkeley r1-zero

3. AI Automation

Flexibility & Tool use

AI’s next leap: When reasoning Powers AI Automation

Reasoning Models

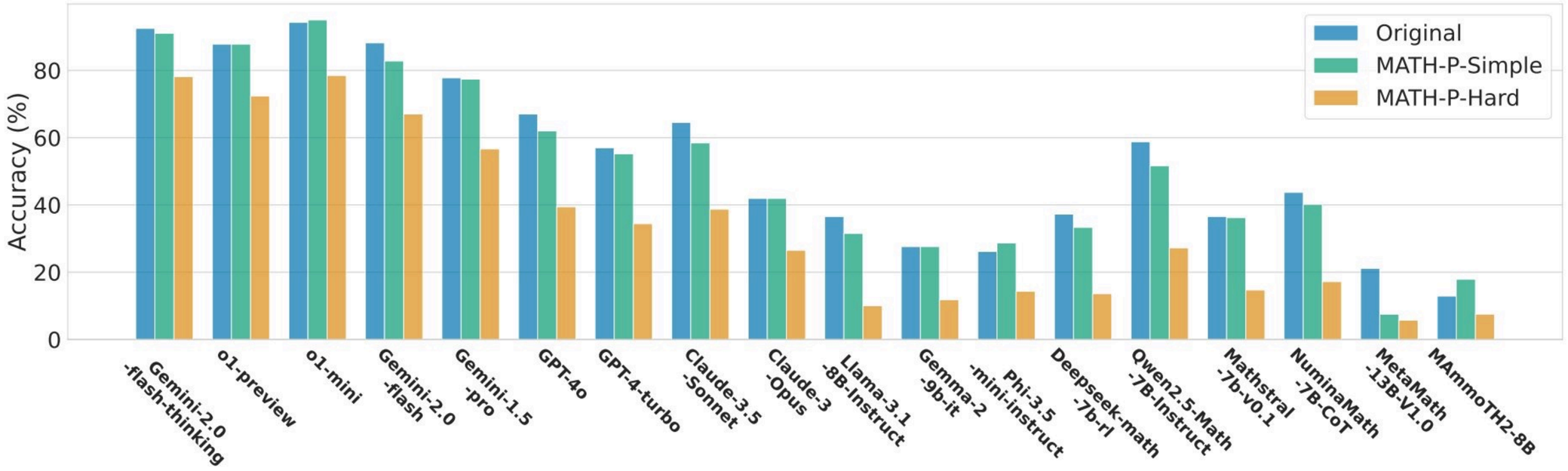
Planning

- Break down complex tasks
- Plan & execute autonomously
- Can backtrack & autocorrect
- Uses general reasoning approaches

Tools

- Perfection instruction following, function calling, structured outputs…
- Create & combine tools
- Orchestrate other models
- Self-modify & optimise code

MATH-Perturb: Benchmarking LLMs' Math Reasoning Abilities against Hard Perturbations
arXiv:2502.06453



UNDERSTANDING THE EFFECTS OF RLHF ON
LLM GENERALISATION AND DIVERSITY link

Building Agents is difficult!

Compound Mistakes

- Multi-step errors are compounded
- Agents can get stuck in loops

Higher Stakes

- Agency requires write access
- Cost of errors are higher in domains like HCLS, FinTech, LegalTech...

Multi-agent coordination

- Difficult to get it right...

Function calling not fully solved

- multi-step & constrained function calling, which requires long-parameter filing, parameter value reasoning, and 128k long context.

Ethical Concerns

ComplexFuncBench: Exploring Multi-Step and Constrained Function Calling under Long-Context Scenario

1-Search Flights

```
{
  "name": "Search_Flights",
  "arguments": {
    "departDate": "2024-12-23",
    "origin": "Washington",
    "destination": "New York"
  }
}
```

The API returns multiple options, including flight numbers, prices, arrival times, and baggage allowances.

2-Retrieve Flight Details

```
{
  "name": "Get_Flight_Details",
  "arguments": {
    "token": "d7699_H4sIAAAA..."
  }
}
```

This step fetches baggage allowance and detailed flight info.

3-Book a taxi

```
{
  "name": "Search_Taxi",
  "arguments": {
    "pick_up_time": "23:00",
    "pick_up_place": "LaGuardia Airport",
    "drop_off_place": "Times Square"
  }
}
```

The correct pickup time is determined dynamically based on the flight's arrival time.

(a) Simple Function Calling

Query: Book a flight from Washington to New York on December 23, 2024.

(b) Complex Function Calling

Query: Help me find the cheapest flight from Washington to New York on the day before Christmas Eve 2024. I want to know the flight's baggage allowances and also arrange a taxi to Times Square 1 hour after landing.

Step 1

Function Calls:
{"name": "Search_Flights", "arguments": {"departDate": "2024-12-23", ...}}
API Response:
[{"number": "d7699", "price": "1287 USD", "arriveTime": "13:00", "arriveAirport": "John F. Kennedy International Airport", ...},
{"number": "d7740", "price": "1025 USD", "arriveTime": "22:00", "arriveAirport": "LaGuardia Airport", "token": "d7699_H4sIAAAA..."}, ...]

Step 2

Function Calls:
{"name": "Get_Flights_Details", "arguments": {"token": "d7699_H4sIAAAA..."}}
{"name": "Search_Taxi", "arguments": {"pick_up_time": "23:00", "pick_up_place": "LaGuardia Airport", ...}}
API Response: [...]

128k context length

Assistant Response: The cheapest flight is ...

Why is this task difficult...

- Multi-step in a single turn - The LLM needs to execute multiple API calls in sequence, handling dependencies between them.
- Implicit Parameter Inference: The request does not explicitly provide the flight arrival time, requiring the model to extract this from the API response and use it to calculate the taxi booking time.
- Long-Context Dependencies: The model must maintain the state across multiple steps and correctly extract details such as baggage information.
- Disambiguation of API Responses: The flight search might return multiple flights, and the model must decide which one to select based on the lowest price.
- Error Propagation and Handling: If the flight details API fails, the model should retry or provide an alternative response.

Why do LLM's fail at this task...

- Incorrect Parameter Extraction - Many LLMs struggle with correctly extracting and passing parameters across multiple steps, leading to invalid API calls
- Premature Stopping: Some models generate a final response before all necessary function calls have been executed, failing to gather all required details 2501.10132v1.
- Hallucinated Parameter Values: LLMs sometimes generate invalid or hallucinated parameters that do not match the API schema, resulting in failed calls2501.10132v1.
- Lack of Robust State Management: Since most LLMs do not maintain an explicit state across function calls, they struggle with tracking information through multiple steps.

More...

- OpenAI has a perfect RLHF pipeline...
- deepseek has no alignment...
- deepseek-r1 has a very simple optimiser. Debatable weather it will generalise as well...
- o1 & o3-mini have function calling + structured outputs
- o1 & o3-mini have “deliberative alignment”, can reason about safety policies in context when responding to potentially unsafe prompts... & “instruction hierarchy enforcement”.