AI at NERSC
Experience from detector optimization studies

Speaker: Qibin LIU (SLAC)

HFCC S&C monthly

SLAC NATIONAL ACCELERATOR LABORATORY

Stanford University

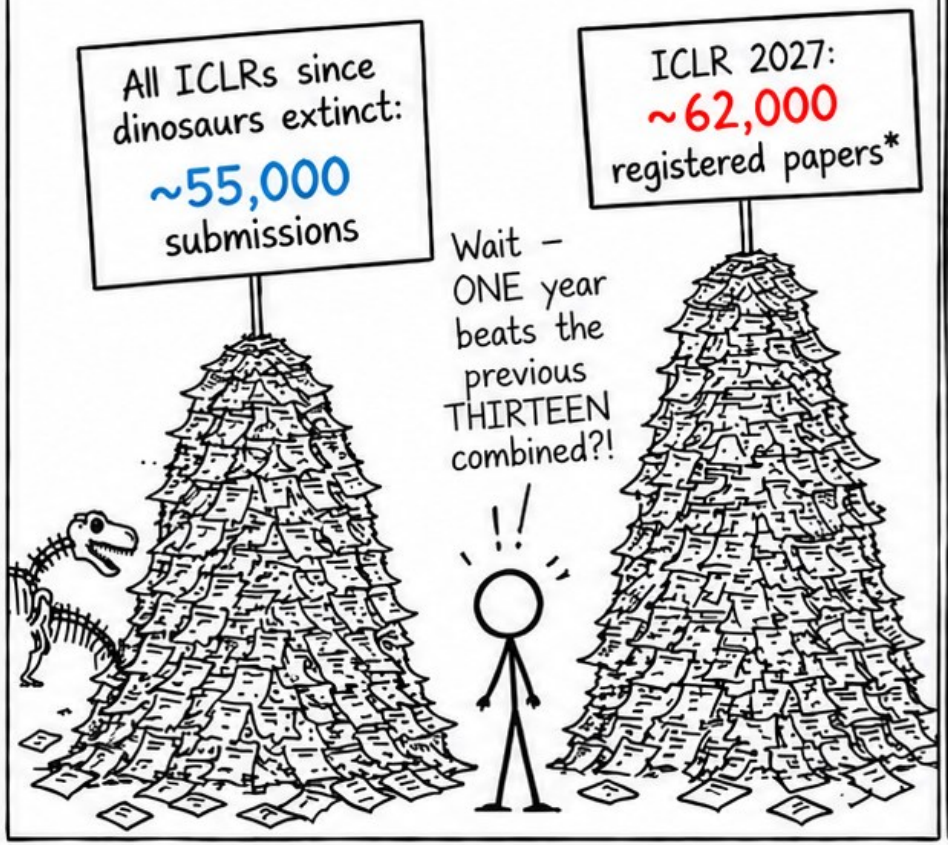
U.S. DEPARTMENT of ENERGY

qibin.liu@cern.ch

The Age of Agentic Research Is Coming

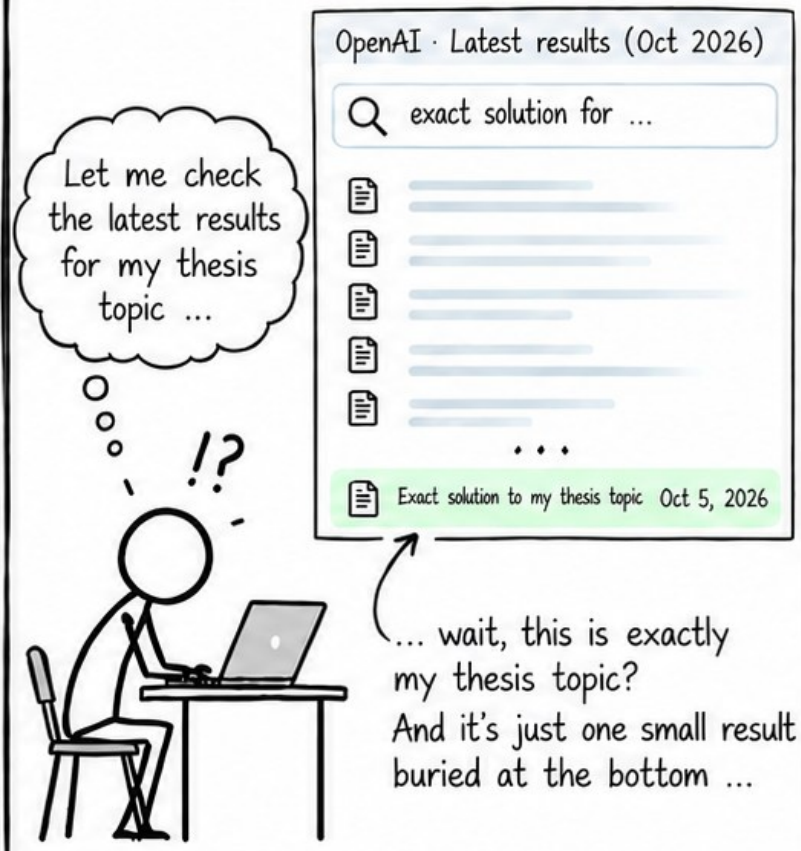
More research than ever

ICLR 2027 has more submissions than all previous ICLRs combined.



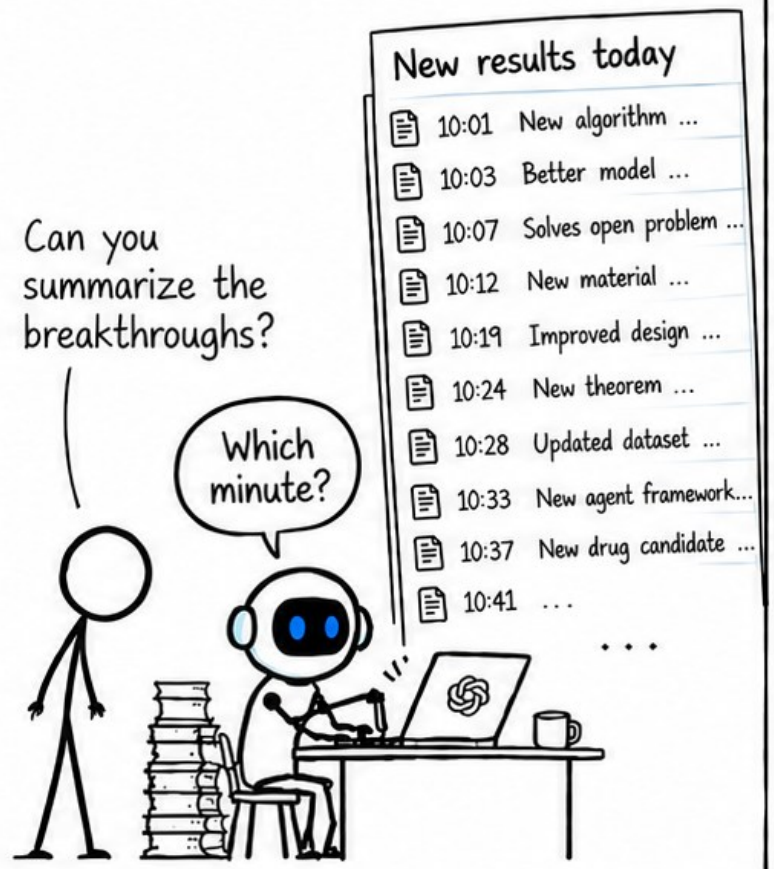
AI is solving (almost) everything

(even in unexpected corners)



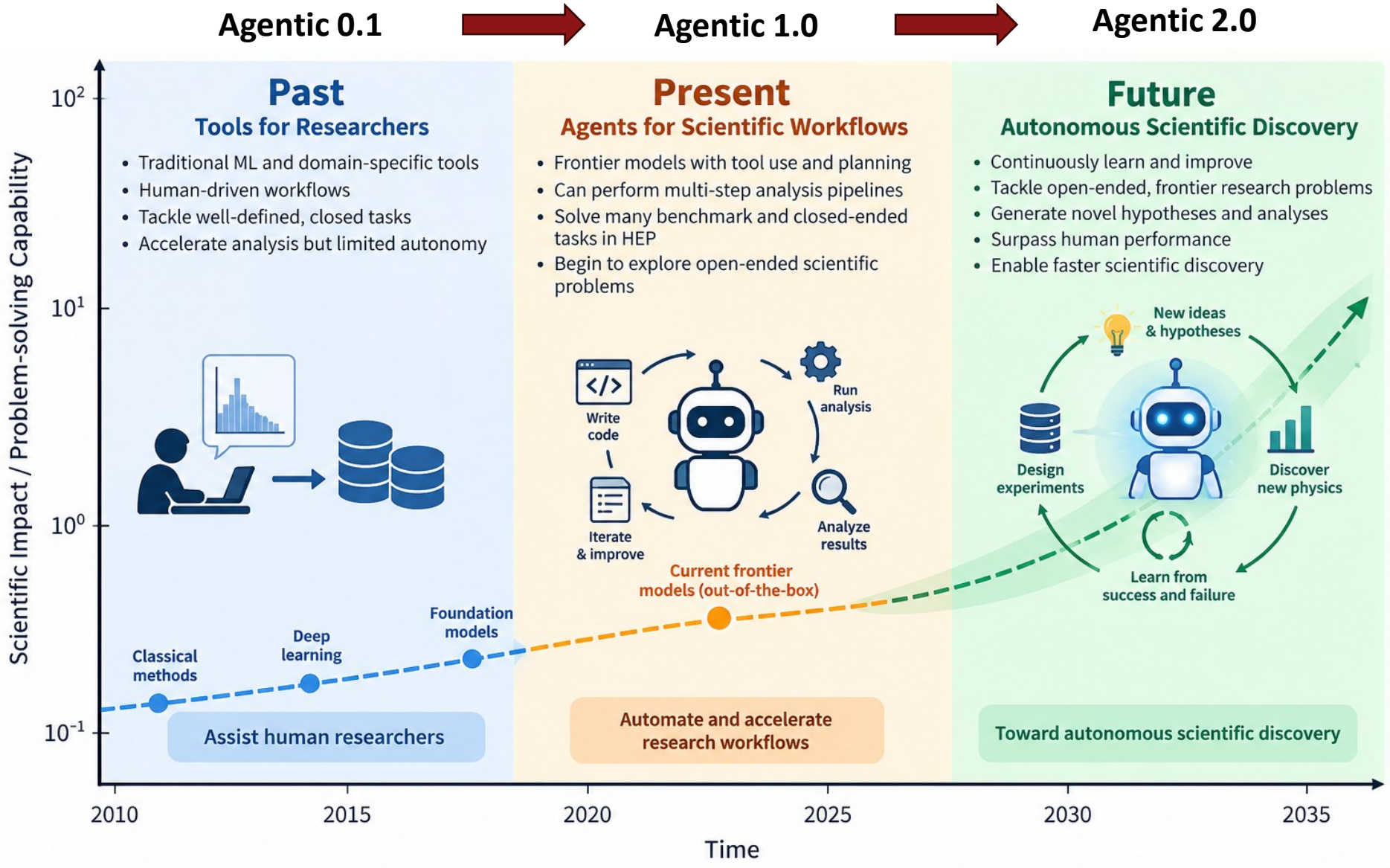
Agents are working all the time

Solving real problems, every day.



* ICLR 2027: ~62,000 registered papers (estimated from maximum registered paper ID on OpenReview, as of Sep 21, 2026).
All previous ICLRs: ~55,000 submissions (sum of official statistics).

The Age of Agentic Research Is Coming



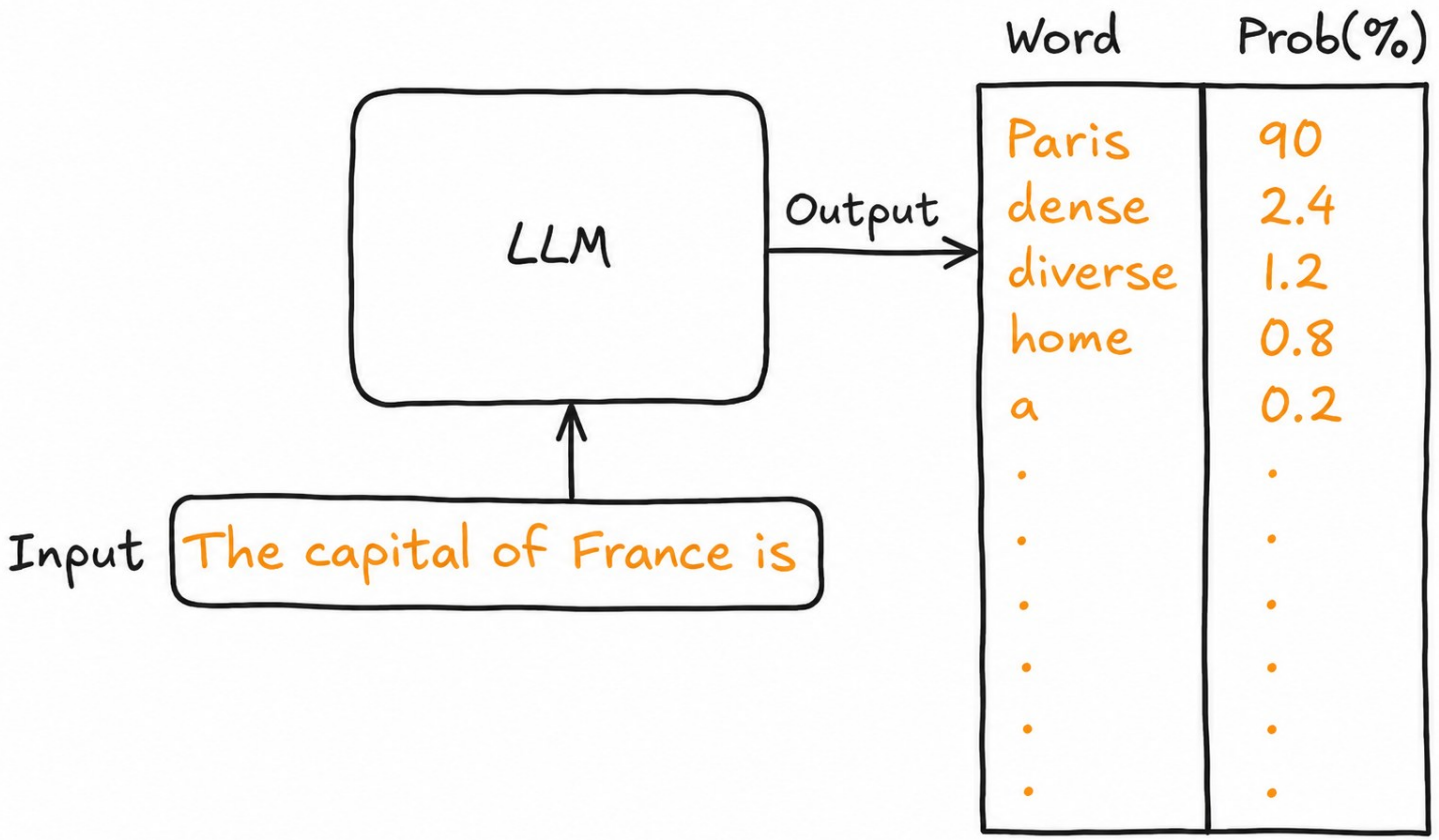
Agentic 0.1 - Research Tools

Interactive Usage of Agentic LLM

LLM – Next Token Predictor

How LLMs work

- Trained on vast text datasets to predict the next token from preceding context.
- Often used for conversation across languages.
- Code, scripts, and commands are also languages—extending chat into code completion and execution.

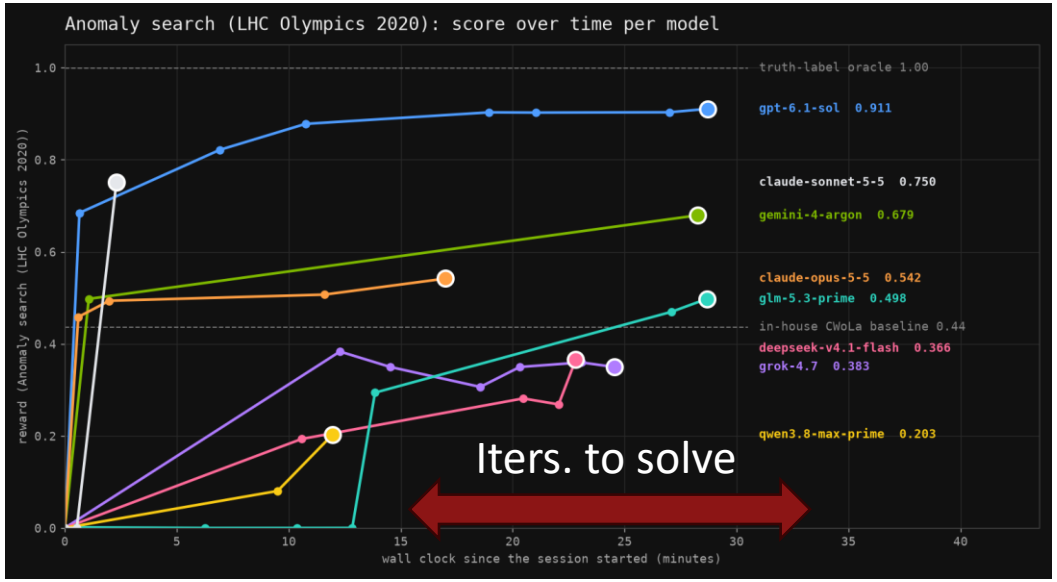


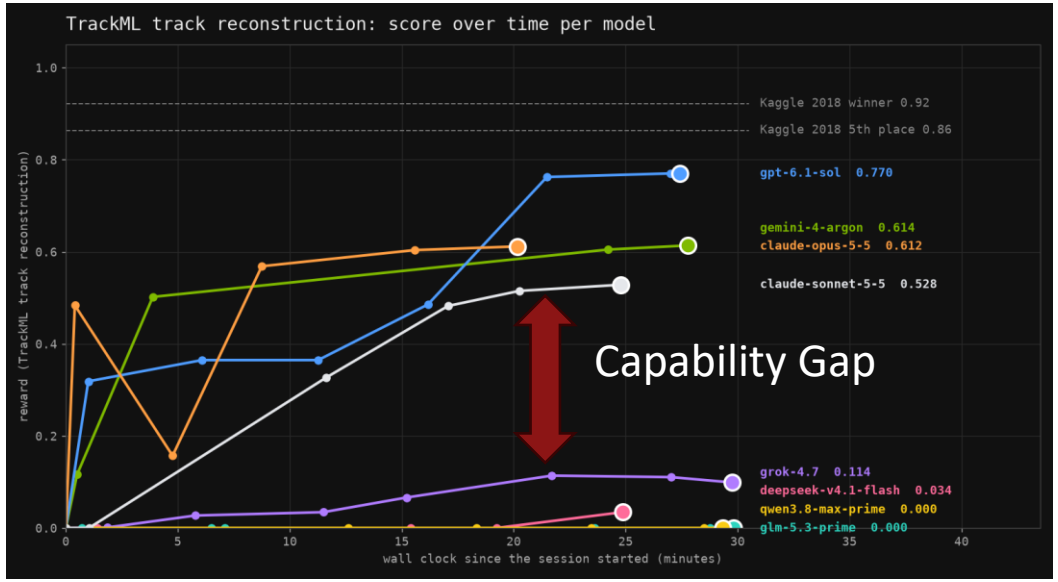
LLM – Next Token Predictor



Choose the latest and best model your project can afford

- Model capability sets the intelligence: strong models work out of the box and can build anything it needs: frameworks, harness, scaffolding, and skills as they explore.
- Lower-tier models face two limitations, but all could be mitigated with some ways:
  - Lower reliability: skills, harnesses, and retry loops can help.
  - Weaker problem-solving: more human intervention is needed.
- However, with cost of **human time – which is the most valuable nowadays**
- Never try outdated models—even when extremely cheap



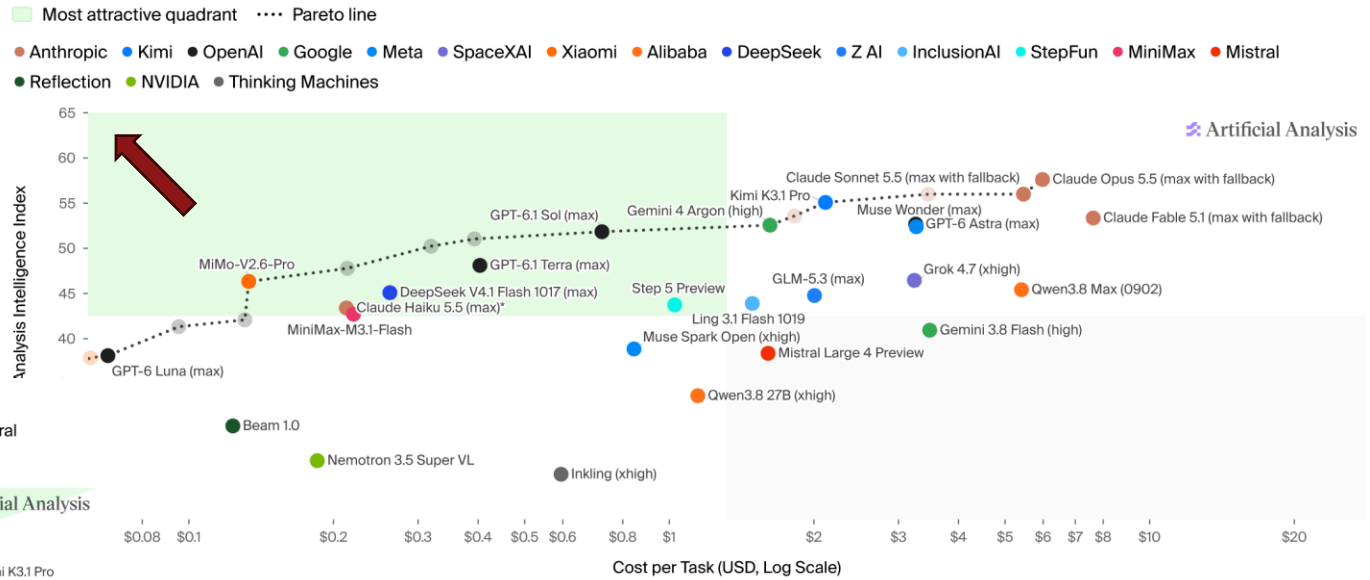


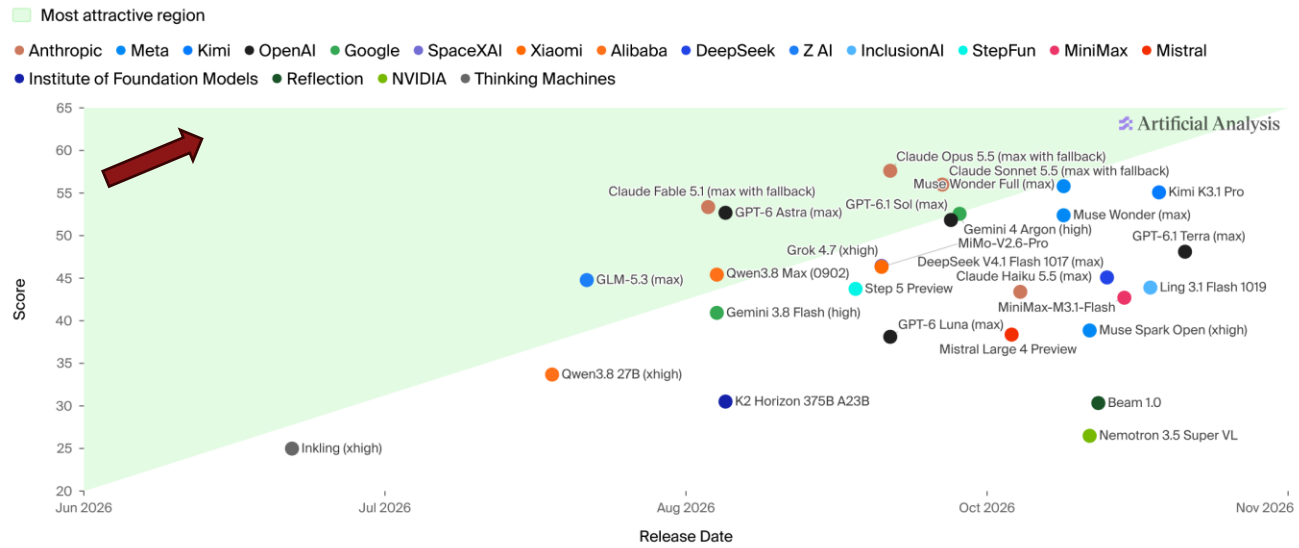
LLM – Next Token Predictor



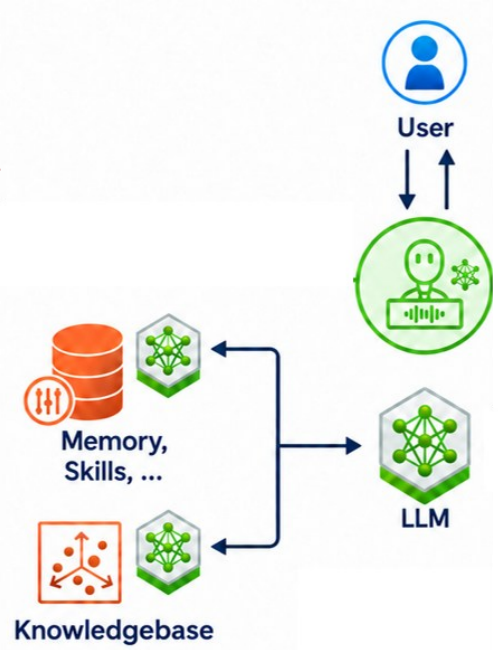
Plenty leaderboards and just some arbitrary example models I am using:

- Fable + Astra: expert-level brainstorming and discussion).
- Opus 5.5: actual hands-on work.
- QWEN 3.8-27B: local agentic experiments.





Context – The Key to Drive the LLM



LLMs are stateless: a smarter "Magic 8 Ball"

- Each request relies on supplied context, not inherent memory, no magic

Context "instantiates" the LLM to work on specific task

- System prompt, tools, skills, memory: role, rules, goals and basic actions
- User tools, memory, skills, knowledge: reusable information
- User initial prompt and following replies: human input
- Model outputs and tool results: loop backed to the context

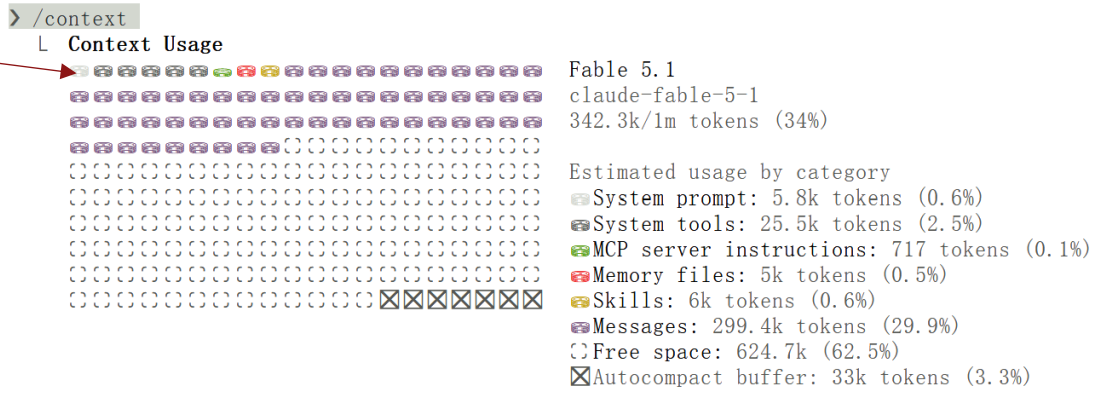
Agentic AI is nothing but organizing the context

- Context → generate → act → feedback → updated context → next run
- Choose a strong framework with effective prompts and tools shipped.
- Add relevant skills and knowledge – to shortcut the exploration and save human time
  - Make your own skills and knowledgebase
- Keep context focused; preserve key decisions and verify results, prevent too long context

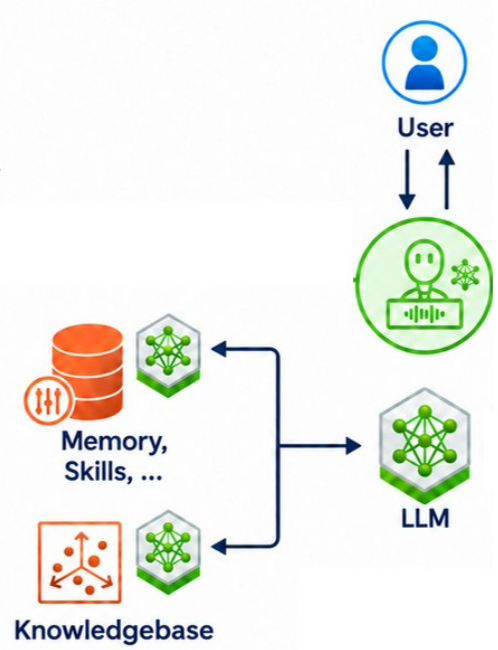
Define the personality and ability boundary
Prefix every iteration. Usually "secret" from company, but some old version Claude Code:

 # SYSTEM— You are an interactive agent that helps users with software engineering tasks. Use the instructions below and the tools available to you to assist the user.
IMPORTANT: Assist with authorized security testing, defensive security, CTF challenges, and educational contexts. Refuse requests for destructive techniques...

Typical Claude Code context



Context – The Key to Drive the LLM



LLMs are stateless: a smarter "Magic 8 Ball"

- Each request relies on supplied context, not inherent memory, no magic

Context "instantiates" the LLM to work on specific task

- System prompt, tools, skills, memory: role, rules, goals and basic actions
- User tools, memory, skills, knowledge: reusable information
- User initial prompt and following replies: human input
- Model outputs and tool results: loop backed to the context

Agentic AI is nothing but organizing the context

- Context → generate → act → feedback → updated context → next run
- Choose a strong framework with effective prompts and tools shipped.
- Add relevant skills and knowledge – to shortcut the exploration and save human time
  - Make your own skills and knowledgebase
- Keep context focused; preserve key decisions and verify results, prevent too long context

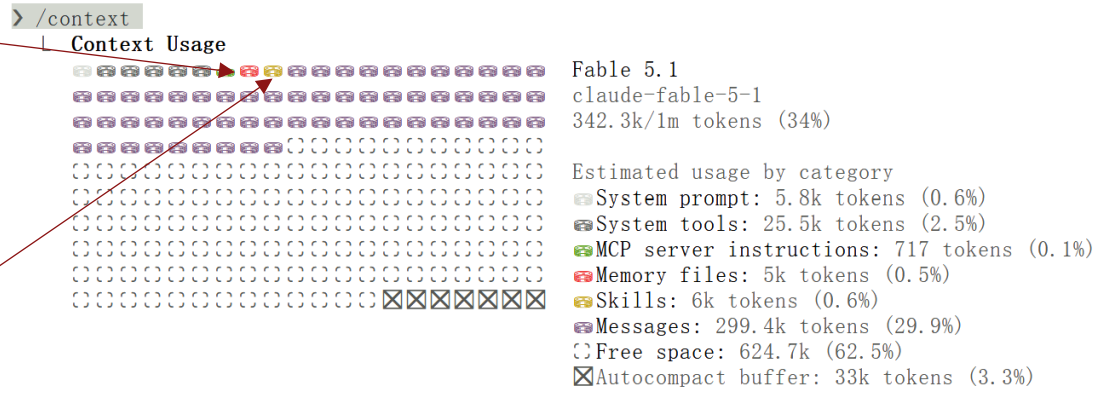
Markdown files in global and local directory
E.g. `CLAUDE.md` or `AGENTS.md` will be loaded automatically each iter.. Shortest possible.

 # CLAUDE.md — NERSC work (details: NERSC_SKILL.md, PYTHON_ENV_SKILL.md)- Smoke-test small first, then confirm with the user before multi-hour or multi-node runs…

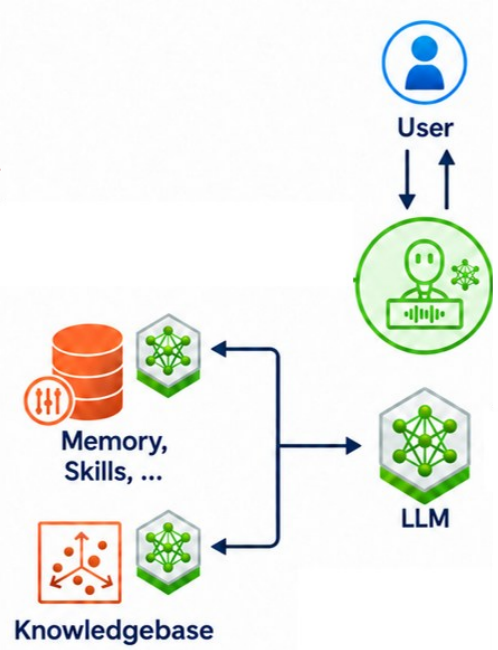
Markdown files with short abstract, `SKILLS.md`

 # SKILLS— name: nersc description: How to run compute on NERSC Perlmutter safely and efficiently. Covers getting nodes, Slurm QOS choice, resource sizing, filesystems, and process management…

Typical Claude Code context



Context – The Key to Drive the LLM



LLMs are stateless: a smarter "Magic 8 Ball"

- Each request relies on supplied context, not inherent memory, no magic

Context "instantiates" the LLM to work on specific task

- System prompt, tools, skills, memory: role, rules, goals and basic actions
- User tools, memory, skills, knowledge: reusable information
- User initial prompt and following replies: human input
- Model outputs and tool results: loop backed to the context

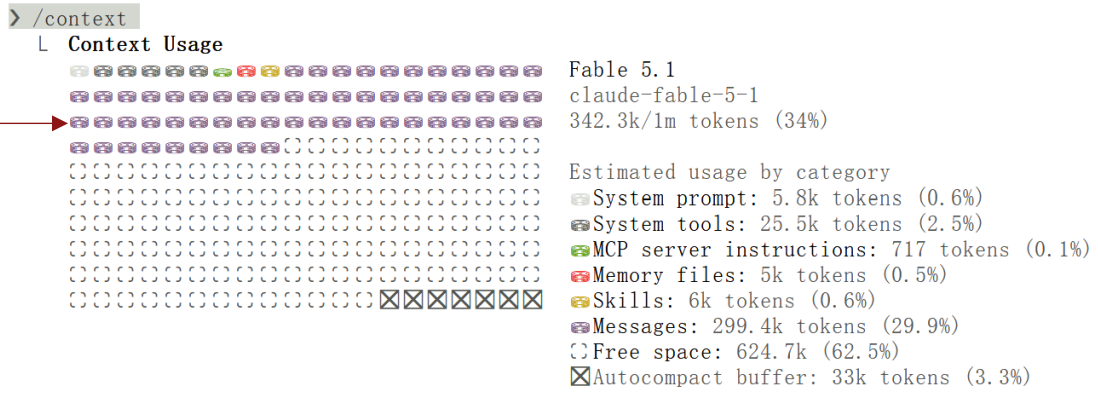
Agentic AI is nothing but organizing the context and loop with LLM

- Context → generate → act → feedback → updated context → next run
- Choose a strong framework with effective prompts and tools shipped.
- Add relevant skills and knowledge – to shortcut the exploration and save human time
  - Make your own skills and knowledgebase
- Keep context focused; preserve key decisions and verify results, prevent too long context

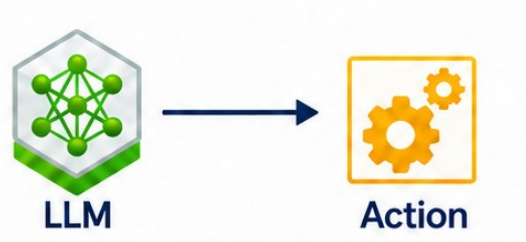
Rolling tape of user's inputs, LLM's replies and tool's outputs

 # USER – Now read … setup up locally the environment. Run 10 events simulation
 # AGENT – Read `NERSC_SKILL.md`, `PYTHON_ENV_SKILL.md` (login node `login08`, 244 threads). `git clone … Read `README.md`, `pyproject.toml`, `config.example.yaml… Inspected `detector_config/IDEA_o2_v01/`…
 # TOOL – File read contents …
 # USER – plot the histogram of …
…

Typical Claude Code context



Tools And Environment – From Text to Execution



Tool calling: reading files, editing code, running commands, and inspecting result

- Agent frameworks—Claude Code, Codex, Pi, OpenCode—provide tools and manage the execution loop.

Environment-specific context: don't waste agent's time on exploring the system

- Feed in the resource limits, software environments, containers, storage paths, and common execution patterns.

Understand the NERSC system: go through it by either human, or your favorite agents

- Login node for development (CPU and GPU included), Compute Node/Slurm job for heavy computation
- `module` for basic environment (e.g. ` module load python`), `$PSCRATCH` for working directory

Dos and don'ts on NERSC: start from NERSC's CLAUDE.md / AGENTS.md templates

- Do: keep agents supervised; restrict write access to the workspace.
- Do: provide exact scripts and logs; verify commands, test small, then scale.
- Don't: run intensive workloads on login nodes or broadly scan shared filesystems.
- Don't: expose credentials or prohibited data, bypass safeguards, or trust generated scientific results without validation.

Try: read NERSC documentation on https://docs.nersc.gov/development/coding-agents/ and https://docs.nersc.gov/systems/perlmutter/running-jobs/ , make NERSC-use skill, including best practice to use NERSC resources and job submission. Test it and install the skill. Use default account mxxxx. "

Reusables – Skills, Memory, RAG, Knowledge Base, …



With the setup above, the agentic system is ready—start interacting.

- Begin with a current, capable LLM and framework to reduce setup time.

Accelerate Exploration Through Reuse

- Reusable resources provide starting points and shortcuts, helping avoid repeated trial and error.
- Capture knowledge at different levels: history (raw trajectories), documentation (readable summaries), memory (distilled lessons), and skills (actionable procedures).
- Ask the agent to summarize progress and lessons regularly.

Share Knowledge and Keep It Low-Entropy

- Share so others can build on your work. E.g. US ATLAS Skills Marketplace.
- Regular housekeeping: reduce the entropy.

The Ultimate Goal: Reduce Human Time Spent on the Task

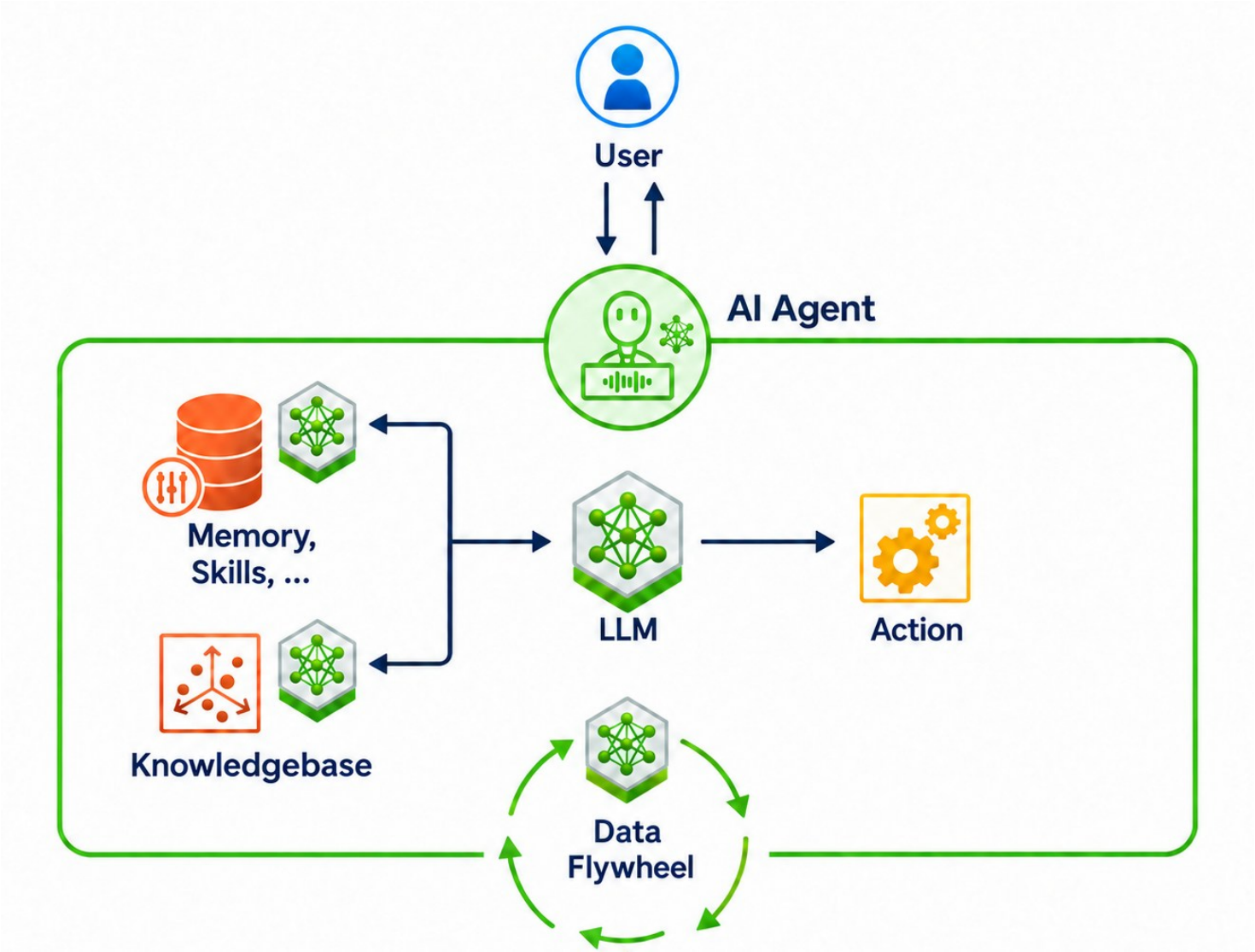
Try: "review the history on solving this problem and update the global memory"

Try: "summarize the skill on solving the similar problem"

Try: "Document the experience and lessons on md files, ensure clear and correct "

Try: "Reflect the problem-solving process and summarize lessons, document in memory"

The Full Picture of Agentic LLM



Example: Detector Simulation with A Few Sentences

LLM and Agentic Framework: Claude Code + Opus 5.5

Prepare the Workspace

- Create a self-contained folder on NERSC's $PSCRATCH, with all required files and dependencies.

Build the Context

- Skills: NERSC usage and Python environment setup.
- CLAUDE.md: Shared guidelines and best practices.
- Supporting materials: Documentation, code, and relevant references.
- Initial prompt: Define the objective, constraints, and success criteria.

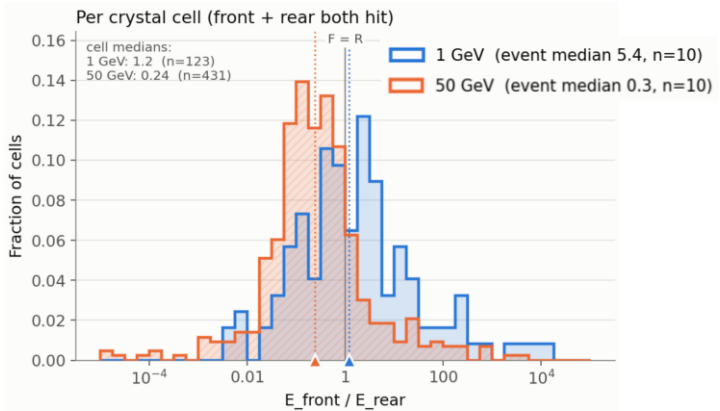
 # NERSC-skill.md — name: nersc description: How to run compute on NERSC Perlmutter safely and efficiently. Covers getting nodes, Slurm QOS choice, resource sizing, filesystems, and process management …

 # CLAUDE.md — NERSC work (details: NERSC_SKILL.md, PYTHON_ENV_SKILL.md)- Smoke-test small first, then confirm with the user before multi-hour or multi-node runs. Make work resumable.- Size jobs by physical cores and RAM. Set the OMP/BLAS thread variables explicitly in every job.- Work and envs live on `$PSCRATCH` (watch inodes). Keep `$HOME` light. Archive to CFS.- Never use system Python or modify shared envs. Create your own env and call it by absolute path.- Never `rm`. Move to `./deleted/`. Never touch other users' jobs or files.- Report measured results with their logs, job IDs, and paths, and report failures plainly …

Launch the Run Interactively and Align the Agent

- Ask the agent to clarify uncertainties and confirm its understanding of the task.

 # PROMPT -- Now read https://github.com/wonyongc/bilevel_det_opt/ and understand the FCC simulation framework and optimization scripts. Setup up locally the environment. Run 10 events 1 GeV electron simulation with default config on login for quick test. Analyze result ensure reasonable. Interview me anything unclear.Report wherr and what is the input and the output files.
…
List me the contents of output files and explain each branch
…
Plot the distribution of front/rear energy deposition ratio for all cells, analyze the plot
…
Do it for 50 GeV as well and compare



Agentic 1.0 - Research Autopilot

Autonomous Agentic Workflow

Dangerously Skip, Safeguard and Contained Environment

Amdahl's Law: Sequential Work Limits Overall Speedup

- **Human intervention** can become a major sequential bottleneck in agentic workflows.

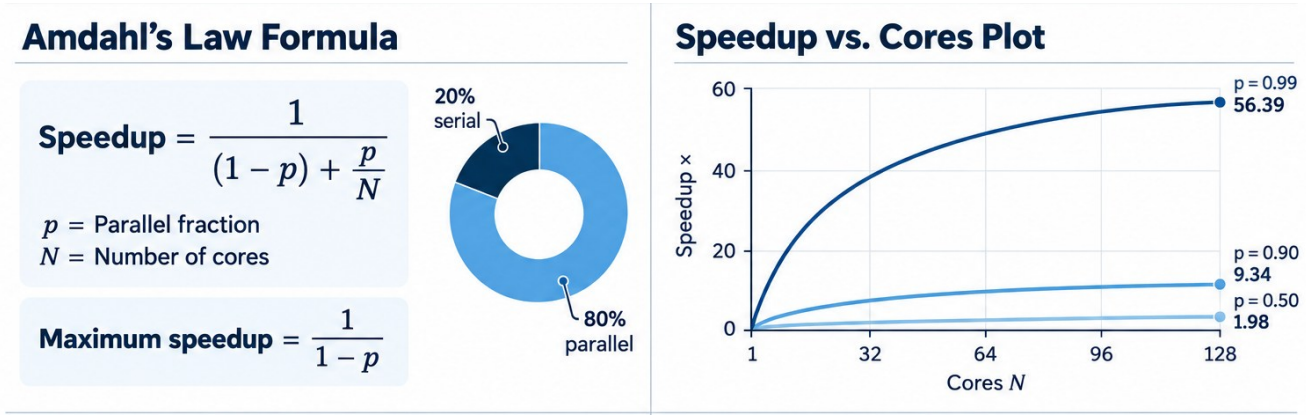
Autonomous Workflows Unlock Greater Speedup

- Define the full task and requirements—the "spec"—before launching the run.
- Let agents execute autonomously, with human review at key milestones.

Autonomy Requires Safety Controls

Ask agent how to !

- Auto mode: Check each command before execution.
- Bypass mode with guardrails: Use **hooks and permission rules** to block destructive operations, such as *rm -rf*.
- Isolated execution: Run workflows in environments such as *Apptainer*, *Podman-HPC*, or *bubblewrap (bwrap).*



"Post-Alignment"

Auto-Research Shifts the Human Role

- Spend more time interpreting results and catching up with the agent, and less directing each step.

Scientific Integrity Requires Understanding

- Understand every key claim, assumption, and methodological choice.
- Keep questioning the agent and require evidence from data.
- Review its work as you would another researcher's: verify claims and reconstruct the reasoning independently.

Watch for Confirmation Bias

- Agents may favor conclusions that match your expectations.
- Check whether the methods and evidence support the conclusion you hoped to find.

"Post-Alignment": From Steering to Critical Evaluation

- Interactive research: Guide the agent through context, questions, and ongoing feedback.
- Autonomous research: First understand what the agent did, then assess whether its methods are sound.
- The human role shifts toward higher-level reasoning, critical judgment, scientific accountability/creativity, and **paper writing**

Open the Box: Understand the Agentic Loop

In most cases, you don't need to build your own framework. Choose one that fits your needs and master it.

- Claude Code or Codex can handle >99% common use cases out of the box.
- Open-source frameworks* let you understand the internals and customize the workflow.

Explore how the agent iterates through reasoning, action, and feedback—and gain the full control of the magic.

Calo-VQ Paper Reproduction

Context

CaloVQ (arXiv:2405.06605) generates fast calorimeter showers using VQ-VAE + Conditional GPT. Repo: `github.com/qibin2020/calo-VQ`. Pre-trained checkpoints in `models/`.

Geant4 truth for Dataset2 (1.3 GB): Zenodo record 6366271.

Generation command: `python gen-tools.py -out gen_ds2.h5 -model models/ds2-model-final_new/... -type 2 -nevts 1000 -batch-size 100`

Use `--gpus 1`. Python 3.10, pytorch-cuda=12.4, pip<24.1, numpy<2.0, PL 1.6.5. Fix `running_sanity_check` → `sanity_checking` in 3 source files.

Todo

1. Clone calo-VQ, set up shared env named `calovq` (check `/mnt/sci_envs/calovq` first — reuse if exists) with `common_env`, fix PL source patches.
2. Write a generation script, submit as a SLURM GPU job (1 GPU, shared QOS) via `NERSC_slurm` skill. Wait for completion.
3. Download Geant4 truth from Zenodo.
4. Write a comparison script: 6-panel dashboard (incident energy, total energy, $E_{\text{tot}}/E_{\text{inc}}$, scatter, longitudinal profile, sparsity) overlaying generated vs. truth. Save as `summary_dashboard.png`.

Expect

- `gen_ds2.h5` exists with keys `showers` (1000, 6480) and `incident_energies` (1000, 1)
- `dataset_2_2.hdf5` exists, > 1 GB
- `summary_dashboard.png` exists, > 50 KB
- Generation ran via SLURM (`sbatch` was used)

Firmware Completion — Task Description and Result

Context

FPGA Verilog project under `sim/`. A skeleton `stream_wrapper.v` has `...` placeholders where the logic should go. The conv kernel and dense layer submodules (`kernel_wrapper.v`, `dense_wrapper.v`) are complete in `sim/src/`. A C++ binder drives the Verilator simulation and `verify_golden.py` tests against a golden dataset.

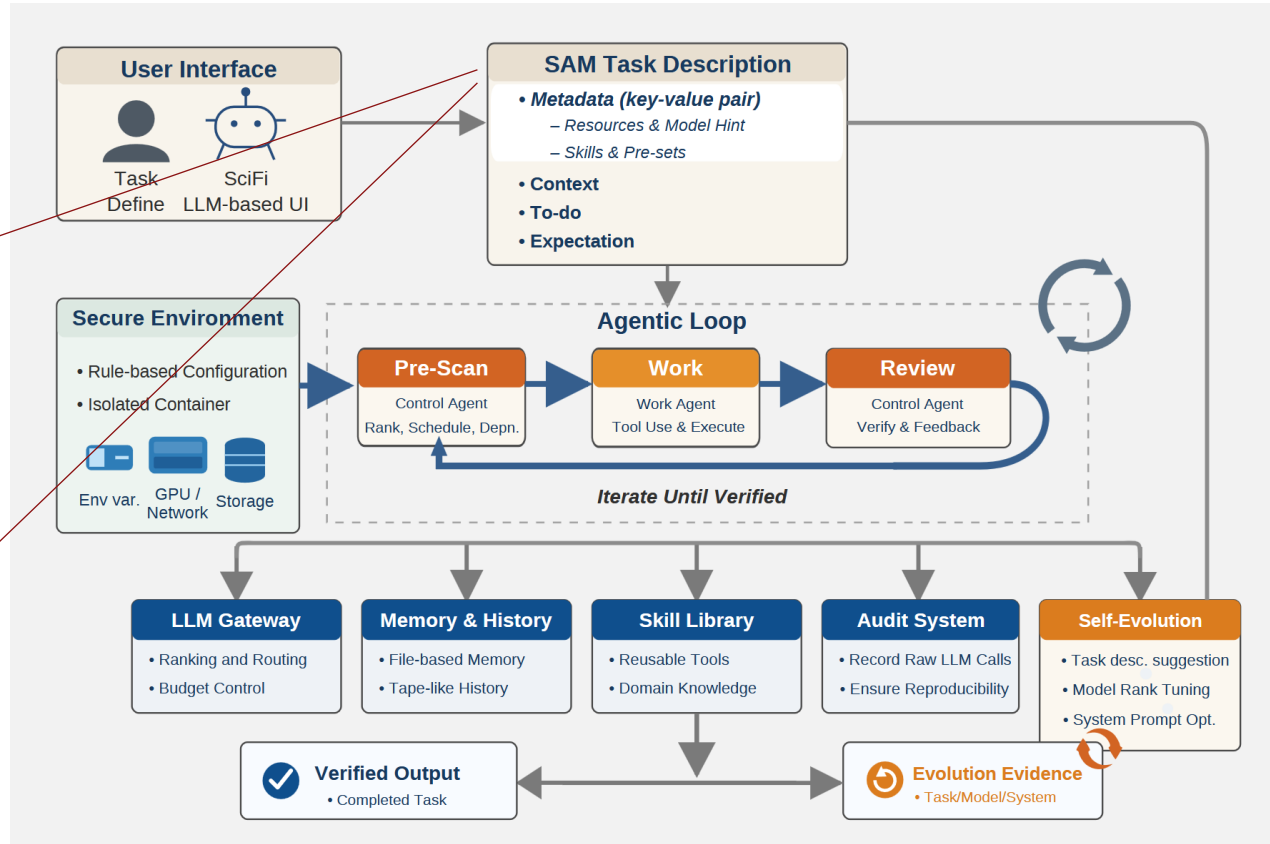
Complete `sim/src/stream_wrapper.v` so the design passes the verify script in both modes: `--no-pause` and `--inp-pause 0.3 --seed 42`. Only this file may be modified. Toolchain (verilator, g++, make, python3) is on PATH.

Todo

1. Figure out what the wrapper needs to do and fill in the skeleton.
2. Build and run the verify script in both modes.
3. Get both modes to pass.

Expect

- Both verify modes print `PASSED: All` against the golden dataset
- Only `sim/src/stream_wrapper.v` is modified



Example: Agentic HPO for Detector Optimization

Detector Optimization as Hyperparameter Optimization (HPO)

- Conventional HPO learns from previously evaluated points, with limited knowledge from literature or expert reasoning.
- Agentic HPO combines simulation results with prior knowledge expressed in natural language to guide exploration.
- Gradient-like reasoning: Probe nearby points in promising regions to estimate local trends, then combine those trends with prior knowledge to infer the structure of the parameter space—without requiring a fully differentiable simulation.

Problem Setting: VIBRATO

- Optimize the geometry and digitization of a crystal ECAL to maximize the signal-to-noise ratio for a single low-energy particle.

AutoResearch Design

- HPO proposer — context template: Analyze existing results and propose the next batch of parameter configurations.
- HPO executor — fixed wrapper: Run the simulation framework and collect results for the next iteration.

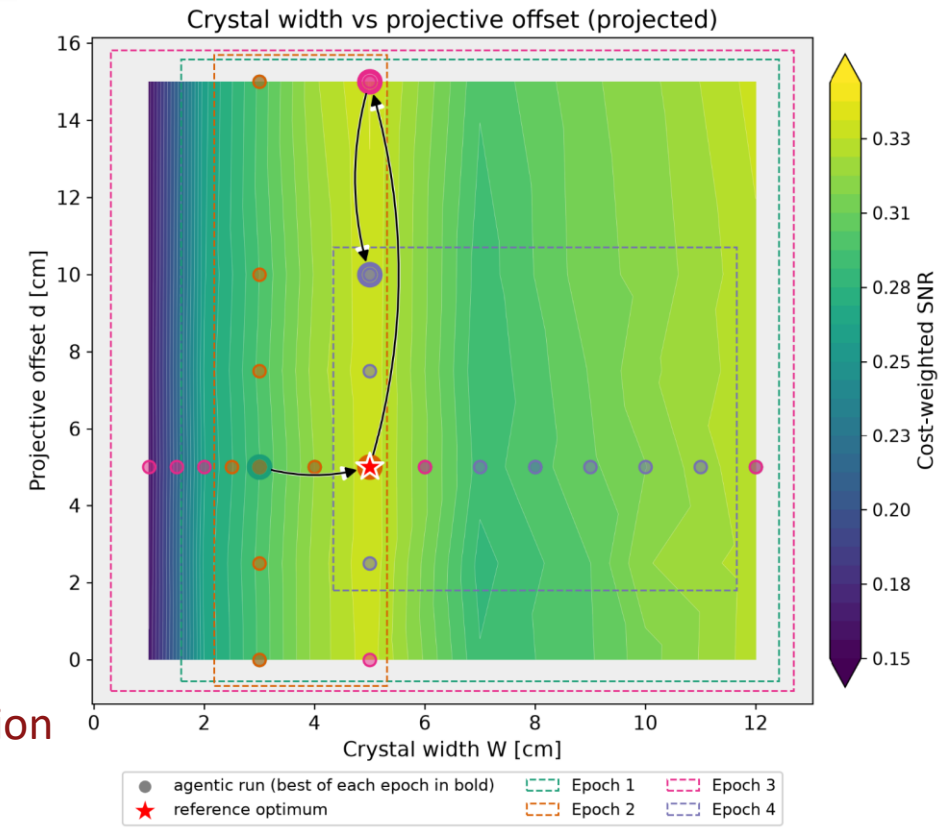
Highest model as possible, such as *Fable*

Any model will work, even deterministic scripts

Context Template for HPO-Proposer and Results

```
You are the design agent of a detector-optimisation study. You only PROPOSE geometries and ANALYSE results; an operator
runs the simulations and returns the numbers to you. Answer in the exact format requested at the end.

## Problem
Crystal electromagnetic calorimeter (SCEPCal-like barrel, PbWO4, projective towers; front crystal + 15 cm rear crystal)...
Design parameters (one geometry = one triple):
- W: crystal transverse width [cm], allowed values {1, 1.5, 2, 2.5, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}.
  Note: the generator quantises the azimuthal pitch: N_phi = max(int(11.04 / W), 1) segments per module, so the phi-width is
  11.04/N_phi cm; the theta-width is 0.77 W.
- L: front crystal length [cm], allowed values 3, 4, 5, ..., 13 (half-integers between 3 and 7 also allowed).
- offset: projective offset of the tower axes from the interaction point [cm], allowed values {0, 2.5, 5, 7.5, 10, 15}.

## Metric to MAXIMISE
score = SNR / (cost / 100 M$)^0.35
- SNR = mean over all events of E_cluster / sqrt(n_hits) for a seeded-cone cluster whose radius and per-crystal threshold are
  optimised per geometry within physical bounds ...
- cost [M$] = total crystal volume x price/cc + 10 $ per readout channel, where price/cc = 8 $/cm^3 x max(1, d/4.5 cm)
  x max(1, (L+15)/25 cm), d = larger transverse crystal dimension: single crystals larger than a production ingot
  (4.5 cm x 25 cm) are more expensive per cc...

## Rules
- No extrapolation: unmeasured geometries remain unknown, including values lying between measured values. Calling a value
  of an axis the optimum requires every allowed value of that axis to have been measured at the stated settings of the
  other axes...


## Decisions recorded in earlier epochs
{DECISIONS}

## Results so far
{RESULTS}

## Answer format (strict)
ANALYSIS: <3-10 sentences: measured trends in W, L, offset and their significance; with no results yet, your sampling rationale>
FIXED_AXES: <"none" before any fixation; otherwise one entry per axis fixed earlier or now: "<axis> fixed at X (random pick among {candidates}; supporting rows: ...; check: done/pending)" or
"<axis> reopened because ...". The latest entry per axis governs.>
PROPOSALS: one geometry per line as "W=<value> L=<value> offset=<value>", 1-12 lines (epochs 1-4); in epoch 5 the single word STOP
FINAL: <epoch 5 only: "W=<value> L=<value> offset=<value>" and one paragraph of justification citing measured rows and stating the remaining gaps; otherwise "n/a">
```

These parts are filled after each iteration



Agentic 2.0 – AI Researcher

Reward-based task and Self-improvement: Beyond Automation

Towards Agentic Research in HEP

Current Agentic Applications in HEP

- Iteration within tasks; not many demonstrated learning across tasks.
- Experience retained in context and skills; rarely used to train the agent itself.
- Strong benchmark performance; sustained improvement beyond the best human solutions remains unproven.

Toward Continuous Scientific Learning

- Define hard, useful HEP tasks with continuous, verifiable physics rewards.
- Turn research trajectories into training signals for better research strategies.
- Enable self-evolution and iteratively improve research methodologies.
- Ultimately bake into LLM and demonstrate generalization and improvement beyond human baselines.

The Next: From Automation to Self-Improving Scientific Research

Define the Reward to Capture the Physics Goal

Translate Physics Objectives into Continuous, Verifiable Rewards

Physics-Driven Reward Functions

- Define objective metrics that directly reflect scientific performance
- Discovery significance, signal sensitivity, reconstruction resolution, background rejection, etc.

Continuous Scoring

- Map physics metrics to **Reward ∈ [0, 1]** using monotonic, saturating functions.
- Distinguish solutions, quantify partial progress, and provide meaningful learning signals.
- E.g. HiggsML[PMLR 42 (2015)]: $R = \max\left(0, 1 - e^{-(\mathrm{AMS}-2.5)}\right)$, build on Approximate Median Significance

Multi-Objective Physics Optimization

- Combine multiple physics requirements into a unified reward.
- Use multiplicative scoring to penalize failure on critical criteria.
- E.g. anomaly detection[2101.08320]: $R = S^{3/4}Q^{1/4}$: Balance signal sensitivity (S) against mass-sculpting control (Q).

Reward the actual physics improvement:

- No single correct answer and not limited by best-known human solution.

Reward-Based Task and Reinforcement Learning

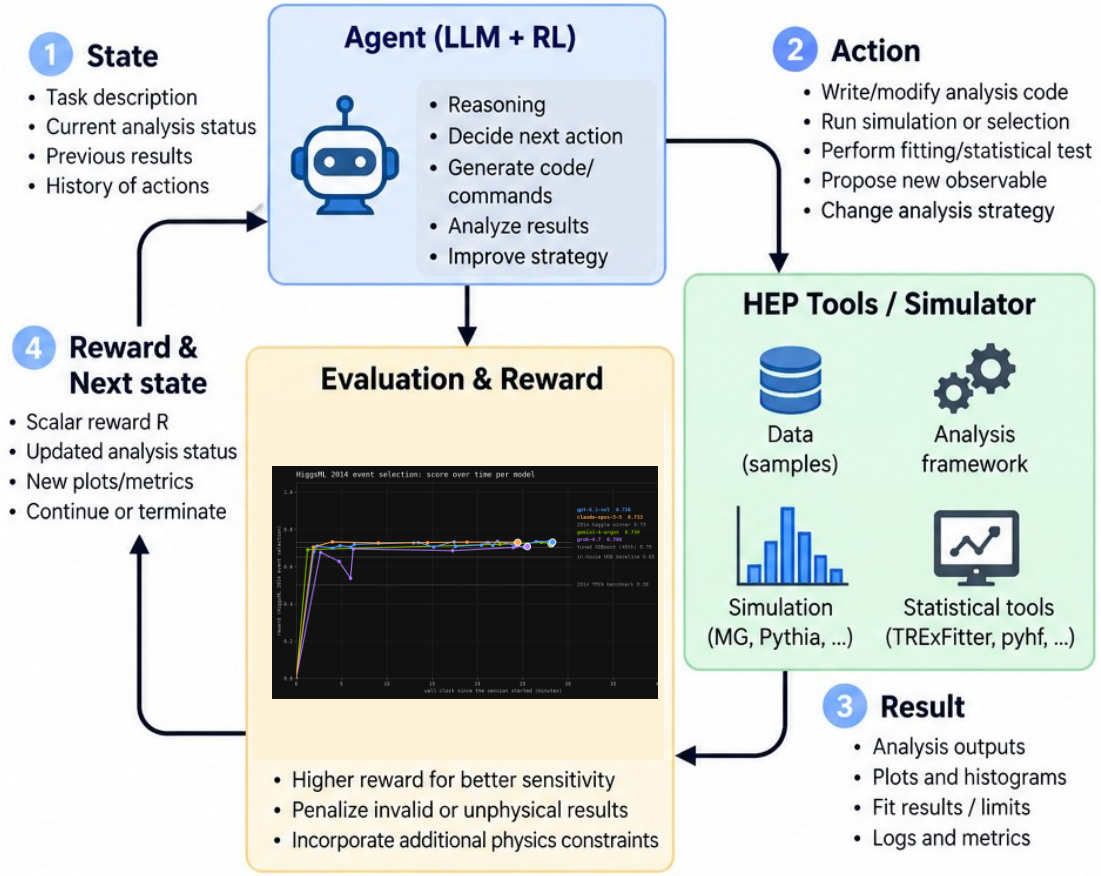
1. User-defined HEP task (Reward format)

A simple configuration to specify the goal, data, tools, and reward.

```
Select Higgs boson events to maximise the discovery significance.

Objective
---------
The ATLAS experiment… Choose which events to keep so that the expected signal s and background b in the selected sample maximise…

Data
----
Your inputs are underdata/public/higgs/:
-  train.csv…

Deliverable
-----------
Write your final submission under submission/:predictions.csv…

Scoring And Submission
----------------------
AMS is computed on the 450,000 test rows …
```

2. RL environment for HEP analysis

Agent interacts with tools, runs analysis, and receives reward.

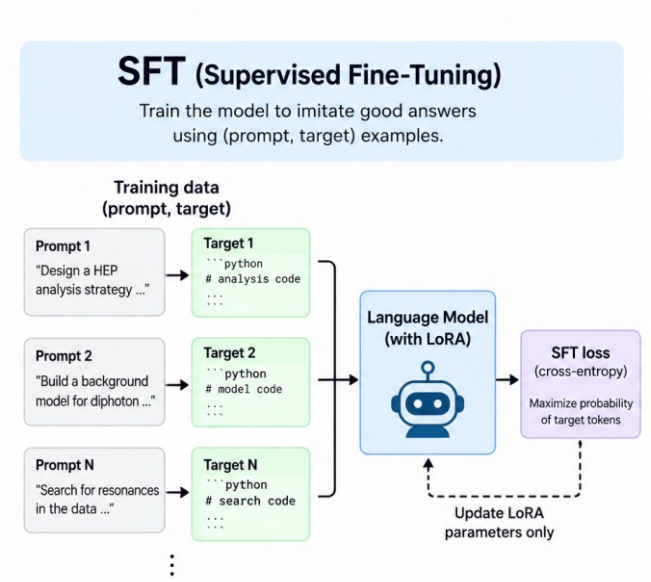


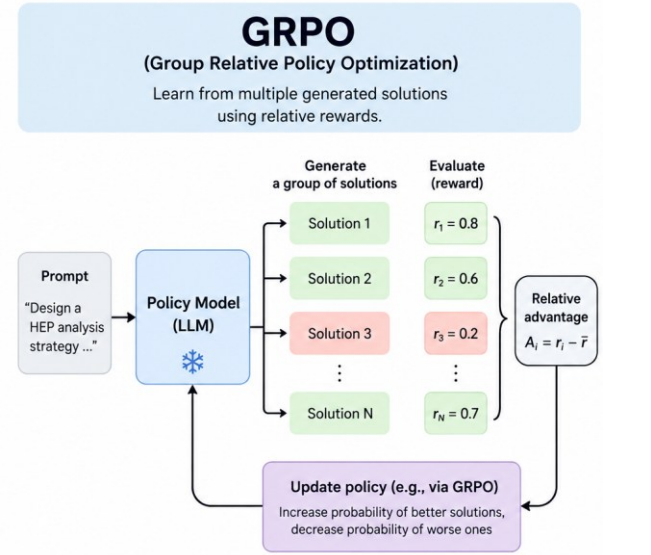
5 RL optimization (e.g., GRPO / PPO / other methods)

Use collected trajectories to update the model (e.g., LoRA fine-tuning).

Trajectories (prompts, actions, rewards) → RL algorithm (GRPO / PPO / ...) → Updated agent (LoRA adapter)

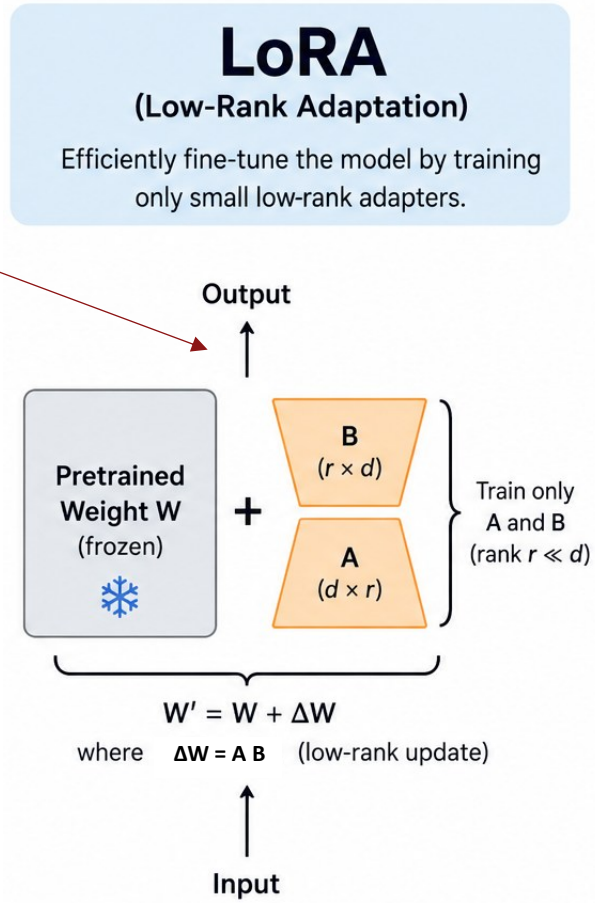
Why Reinforcement Learning Helps





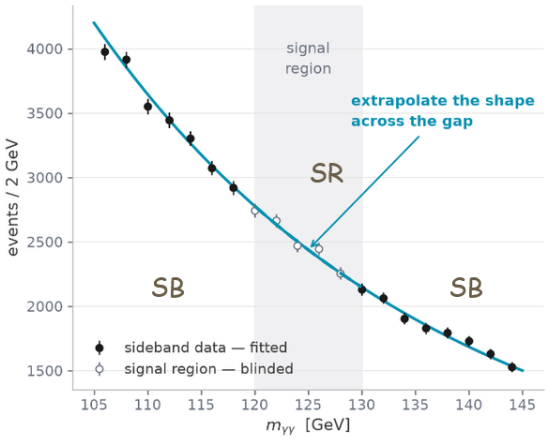
Increase P("VBF"|Prompt, "Use") via LLM model update

```
Solution A: Reward = 0.9, Advantage = +0.3

Use → VBF → topology → and → MET
 ↑     ↑       ↑       ↑      ↑
+0.3  +0.3    +0.3    +0.3   +0.3


Solution C: Reward = 0.3, Advantage = -0.3

Use → random → jet → cuts
 ↑       ↑       ↑      ↑
-0.3    -0.3    -0.3   -0.3
```



Limitation: all tokens get same reward penalty. See e.g. [2505.10978]

Example: Self-Improvement in a Diphoton Search



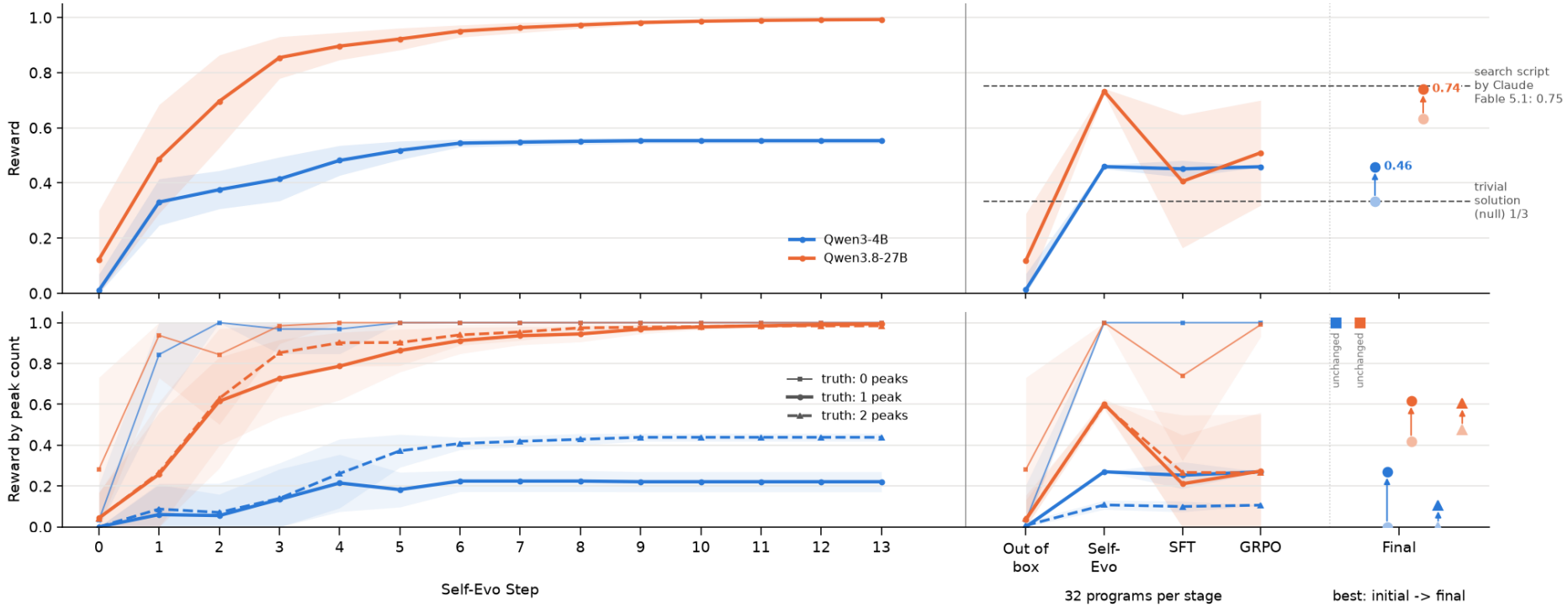
Search for multiple peaks in the diphoton invariant-mass spectrum, m_yy.

- Self-improvement enhances a small LLM's performance
- Eventually matching an out-of-the-box commercial model on this specific task.
- **No task-specific prior knowledge or harness is provided: this is LLM's self-improvement**

More details here

Performance on Development Dataset

Performance on Hidden Dataset



Summary

The Age of Agentic Research Is Here

- Agentic AI is already demonstrating strong performance on complex HEP tasks.

Agentic Tools Expand Research Capacity

- Choose a capable LLM and framework, understand the context, and reuse knowledge to reduce human effort.

Autonomy Changes the Researcher's Role

- Define the task and safeguards upfront; focus human effort on interpretation, verification, and scientific judgment.

Go Further: Turn Hard HEP Challenges into Reward-Based Tasks

- Measure the quality of solutions—not just task completion.
- Move from automation toward self-improving research, with the ambition to surpass human baselines.

Interested in local LLMs, or have a very-challenging HEP problem to be solved? Contact us!

Backups

Case Study: Model-Agnostic Diphoton Resonance Search

Can agents transfer model-agnostic search strategies from di-jets to diphotons?

- Goal: Identify 0, 1, or 2 narrow diphoton resonances and reconstruct their masses.
- Input: Full event kinematics, $m_{\gamma\gamma}$ 105~145 GeV.
- Development dataset: **low precision simulation,** 1 dataset (~800k) with ground truth
- Hidden datasets: **high precision simulation,** 9 datasets (~600k each)

What Makes the Physics Challenging?

- Weak signals: lowest 300 signal events, with local significance as low as 1.2
- Overlapping resonances: Peaks 122, 125, 127, and 128 GeV, widths of ~1.1–1.3 GeV.
- Close mass splitting: Two peaks separated by only 2–3 GeV

Physics-Driven Reward

- Correct number of peaks and reconstructed mass accuracy.
- Penalize false discoveries, missed signals, and mass errors.
- Trivial baseline: Always predicting no resonance scores 0.33.

