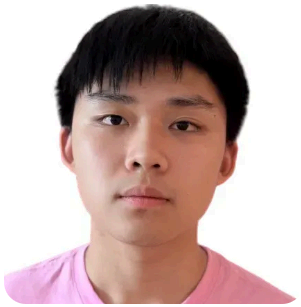
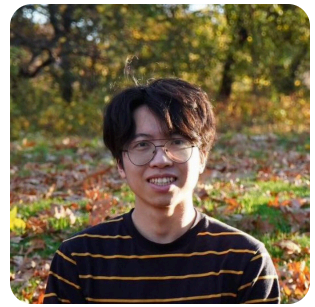
Hands-on experience using LLMs for TCS (and math)

Binghui Peng
University of Maryland

Runzhou Tao
University of Maryland



Steven Wang
University of Maryland



Hantao Yu
Columbia University

Outline of the talk

- **Part 1. Proof discovery** Solving open questions with a simple automated pipeline (GPT-5.5 Pro + Claude Opus 4.8)
- **Part 2. Formalization** Towards scalable verification using LEAN

The LLM mathematician

A wave of recent results: frontier models contributing to genuinely new mathematics

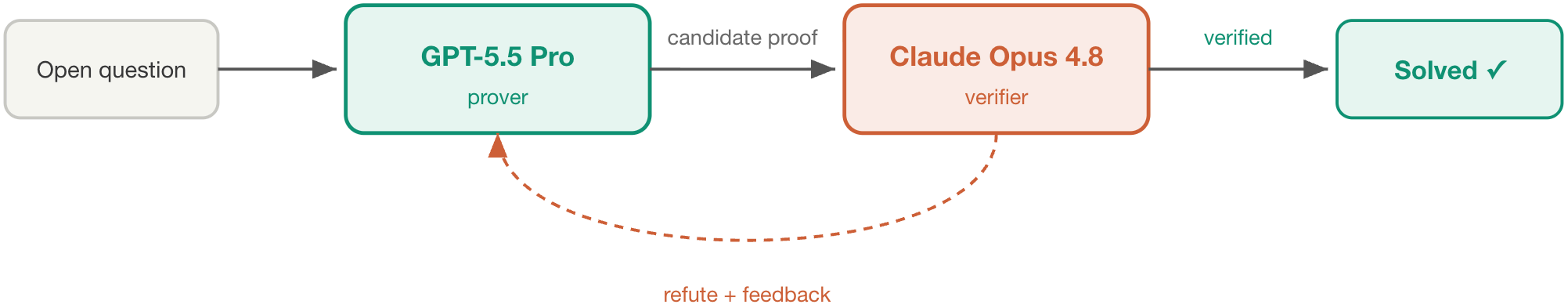
- **OpenAI**
  - *Early science acceleration experiments with GPT-5* — 4 new math results [OpenAI '25]
  - A few Erdős problems via GPT-5.x Pro [OpenAI '25–'26]
  - Disproof of the unit-distance conjecture [OpenAI '26]
- **Google DeepMind — Gemini**
  - *Semi-autonomous mathematics discovery with Gemini* — first proof paper [DeepMind '26]
  - AlphaProof Nexus — 9 Erdős problems (Gemini + Lean) [DeepMind '26]
- Peking Univ.'s Rethlas + Archon resolves Anderson's 2014 commutative-algebra conjecture; AxiomProver publishes 5 peer-reviewed math papers, … [Peking '26; Axiom '26]

What I want to present in this talk

- A first hands-on experience of using LLMs: how good (or bad) they are, and how to use them for research, opinions are my own
  - Not from Twitter; no complicated agentic pipeline — just two weeks of real research from the first principle

What I did (in the past two weeks)

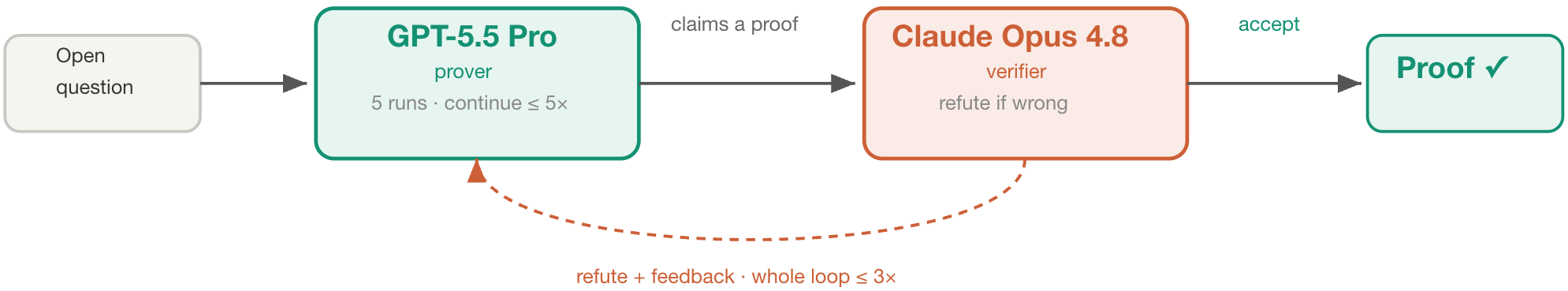
A simple, automated pipeline: GPT-5.5 Pro proves, Claude Opus 4.8 verifies



- Run on a curated list of open problems: open questions from my past paper · COLT · commutative algebra

Workflow

- **Rule of thumb:** as simple as plausible
- Prompt = a pointer to the open question



Benchmarks

- **My research questions.**
- **COLT open questions.** Posed every year (~5–10/yr); after removing solved ones: ~20 fully open + ~50 partially open. I test fully-open ones
- **Commutative algebra.** From the 2014 open-problems list [Cahen et al. '14] (same list as in the Peking group's paper); I asked GPT/Claude to pick 5 most promising ones.
  - They are promising, approachable questions
  - Test whether complicated agentic framework is still necessary
  - Test our LEAN formalization framework
  - Can I do "zero-knowledge" math — research outside my own area?

Results

Source	Solved
COLT open questions	? / 20

- We tested all 20 fully-open COLT open problems — how many did the pipeline solve?
- Take a guess.

Results

Source	Solved
COLT open questions	4 / 20
Commutative algebra	4 / 4
My FOCS'22 paper	1 / 1

- Rest of this part: (1) are these questions interesting — and is the proof novel? (2) when GPT fails; (3) some simple ablation study

Online leverage-score sampling vs. an adaptive adversary

κ = condition number
ϵ = spectral approximation error

- **Leverage-score sampling.** For $A = \sum_t a_t a_t^\top$, sample $\tilde{O}(d/\epsilon^2)$ rows $\propto \tau_t = a_t^\top A^{-1} a_t$ to get $(1-\epsilon)A \preceq \tilde{A} \preceq (1+\epsilon)A$ [Spielman–Srivastava '11]
  - A core primitive in numerical linear algebra: regression, ℓ_2 solvers, …
- **Online (oblivious adversary).** Rows a_t arrive one by one; still keep $\tilde{O}(d/\epsilon^2)$ [Cohen–Musco–Pachocki '16]
- **Open question** (my FOCS'22 paper). Against an adaptive adversary (next row depends on past coins), the known bounds carry an extra factor d:
  - $\tilde{O}(d^2\kappa^2)$ [Braverman et al. '21] · $\tilde{O}(d^2 \operatorname{polylog} \kappa)$ [Jiang–Peng–Weinstein '22]
  - Can we match the oblivious $\tilde{O}(d \operatorname{polylog} \kappa)$ — remove the extra d?

The technical barrier

- The oblivious proof normalizes by the **fixed terminal** Gram matrix A_T (matrix Chernoff–Freedman [Tropp '12])
- Against an adaptive adversary A depends on a_t, so that normalization is not predictable
- What GPT does: a trace-exponential argument — beyond Tropp's matrix Chernoff
  - Lieb concavity (already used by Tropp) [Lieb '73]
  - new: a cleverly designed potential $\Phi(Z_t),\ Z_t = A_t^{-1/2}(\tilde{A}_t - A_t)A_t^{-1/2}$
  - new: the Brown–Kosaki trace inequality $\operatorname{tr} e^{QHQ} \le \operatorname{tr}(Q^2 e^H) + \operatorname{tr}(I - Q^2)$ — the "matrix Jensen" step [Brown–Kosaki '90]

Single-shuffle SGD: a spectral-norm conjecture

- Single-shuffle vs random-reshuffle vs full-batch GD — how do they order? [Yun–Sra–Jadbabaie '21, COLT]
- Symmetric $A_1, \dots, A_n$ with $(1 - \eta_{n,K})I \preceq A_i \preceq I$; over K epochs:

$$\underbrace{W_{\mathrm{SS}} = \frac{1}{n!}\sum_{\sigma}\left(\prod_i A_{\sigma(i)}\right)^K}_{\text{one fixed shuffle}} \quad \underbrace{W_{\mathrm{RS}} = \left(\frac{1}{n!}\sum_{\sigma}\prod_i A_{\sigma(i)}\right)^K}_{\text{fresh shuffle / epoch}} \quad \underbrace{W_{\mathrm{GD}} = \left(\frac{1}{n}\sum_i A_i\right)^{nK}}_{\text{full batch}}$$

- **Conjecture 1:** $\|W_{\mathrm{SS}}\| \le \|W_{\mathrm{RS}}\| \le \|W_{\mathrm{GD}}\|$ with a dimension/matrix-free $\eta_{n,K}$

Resolution of the conjecture

✗ $\|W_{\mathrm{SS}}\| \le \|W_{\mathrm{RS}}\|$ **is false**

- Refuted already at $n = 3, K = 2, d = 4$, arbitrarily close to I
- Counterexample: rank-one D_3 projectors P_1, P_2, P_3; $B_i = qI + (1-q)P_i$; $A_i = B_i \otimes B_i$ (eigs $1, q, q, q^2$)
- $\|W_{\mathrm{SS}}\| - \|W_{\mathrm{RS}}\| \sim \frac{(1-q)^6}{16} > 0$ as $q \to 1$ ⇒ no dim/matrix-free $\eta_{n,K}$ exists

✓ $\|W_{\mathrm{RS}}\| \le \|W_{\mathrm{GD}}\|$ **holds for well-conditioned matrices**

- false in general [Lai–Lim '20] (without well-conditioness)
- AI proves it near the identity: $(1 - \frac{1}{4n^2+1})I \preceq A_i \preceq I \;\Rightarrow\; \|W_{\mathrm{RS}}\| \le \|W_{\mathrm{GD}}\|$

Some other COLT open questions

- **Langford's question.** An oracle reduction from k-class conditional-probability estimation $P(y \mid x)$, $y \in \{1, \ldots, k\}$ to binary regression $B(z \mid x)$, $z \in [0, 1]$ [Langford et al. '10]
  - Key trick: use matching-vector code instead of Hadamard code [Dvir–Gopalan–Yekhanin '11]
- **Quantum circuits.** Learning an unknown quantum circuit when you may intervene on a single gate (inject a value at its input); prove an exponential query lower bound — partially solved [Kun–Reyzin '15]
- **Dataset selection.** Regression: from a large dataset of size N, pick $n \ll N$ points so that minimizing the loss on the n also minimizes it on all N [Hanneke et al. '25]
  - Weighted selection is solved; unweighted was open → solved by reweighting the hard instance

Commutative algebra questions

- Drawn from the 2014 open-problems list [Cahen et al. '14]
- Overall impression: somewhat **niche** questions (the big open problems there were not the ones we selected)
- The **Peking Univ. group** showed natural-language proof + formal reasoning via an agentic pipeline; later work uses it to solve several more [Peking '26]
- We demonstrate prompting GPT 5.5 pro is enough

When does AI fail?

- The problem is too hard, or the problem definition is not clear enough — especially an issue for COLT open questions
- Concretely:
  - In some cases, I can't quite understand what the question is asking
  - Some questions ask to "characterize …" — a vague goal for an AI

Ablation study: Could we use GPT/Codex instead?

Same two problems, solved with cheaper / open pipelines — each attempt scored **0–2** (2 = fully correct)

Pipeline	COLT question	Leverage-score question
pass@128	68 got 1/2 · 60 got 0/2 (of 128)	128 got 0/2 (of 128)
Sequential revision round 4 final	2 got 2/2 · 1 got 1/2 · 13 got 0/2 (of 16)	16 got 0/2
Sequential revision round 8 final	1 got 2/2 · 15 got 0/2 (of 16)	16 got 0/2
Huang–Yang pipeline	4 got 2/2 · 8 got 1/2 · 4 got 0/2 (of 16)	11 got 2/2 · 5 got 0/2 (of 16)

The agentic Huang–Yang pipeline works pretty good [Huang & Yang '25].

Digest of Part 1

- **Observation 1.** LLMs solve real open questions at a measurable rate
  - I tested 20 fully-open COLT questions (~20% solved); ~50 remain — does this rate generalize?
  - I'd love a **community TCS benchmark** to measure AI progress
- **Observation 2.** You don't need a big tech lab — any grad student can do this now

Part 2. Towards scalable verification using LEAN

A guardrail for AI-generated proofs.

A short introduction to LEAN

- **LEAN** is an interactive theorem prover: you state a theorem, then give a proof its kernel mechanically checks — if it compiles, it is correct [de Moura et al. '15]

State it — without proving it

```
theorem sqrt2_irrational :
    Irrational (Real.sqrt 2) := by
  sorry
```

`sorry` = a placeholder; Lean accepts the file, but the theorem is not proved

State it + prove it

```
theorem add_comm (a b : ℕ) :
    a + b = b + a := by
  omega
```

a real proof — kernel-checked ✓

Why we need formal verification

- **Why?** Correctness of the proof; a better, structural understanding of the math
- **Why we don't use LEAN** (a faculty member asked me this in my job interview two years ago) — it is painful to write
- AI tools give huge help and ease the pain — but do they destroy our understanding of the math?
- **My opinion:** formalization is a **guardrail** for AI-generated proofs

Two facets of LEAN formalization

Part 1. Formalize the statement

- You must state the problem correctly — else the AI can cheat, e.g. hide what you wanted to prove inside a definition:

```
-- Goal: prove  ∀ x, P x
def P (x : ℝ) : Prop := True   -- hard part hardcoded away
theorem main : ∀ x, P x :=
  fun _ => trivial             -- compiles ✓, proves nothing
```

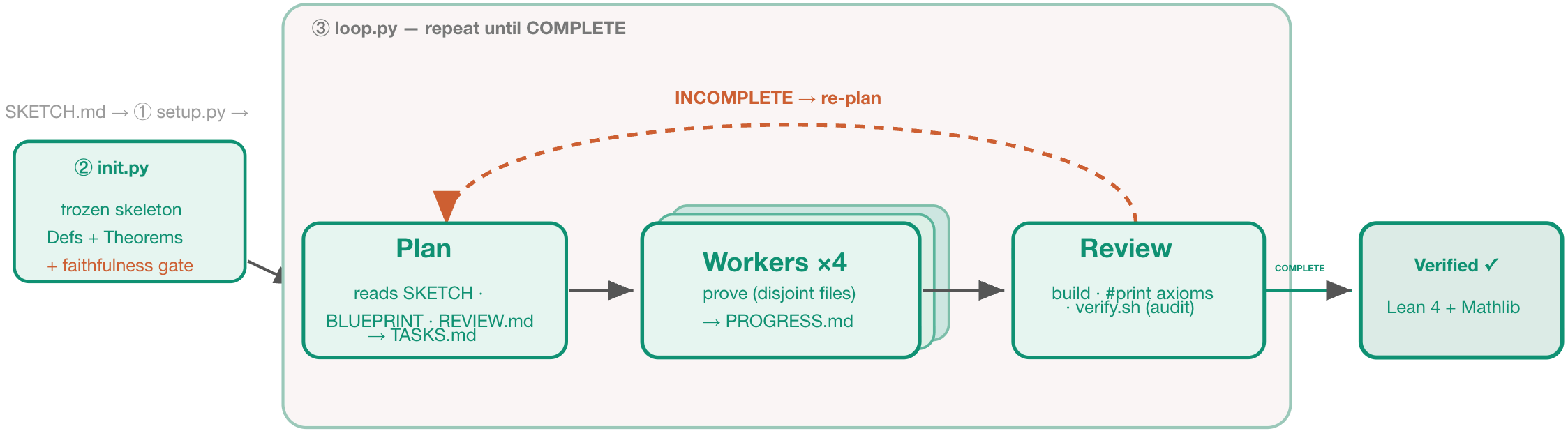
- You can ask GPT/Claude to check the statement, but no guarantee short of manual review — **a general open problem!**

Part 2. Formalize the proof

- Turn a natural-language proof into LEAN
- Huge help from AI here — this is where current tools shine

Our pipeline for proof formalization

Rule of thumb: keep it simple, let the AI run long (I use /goal with Codex). Steven built a more autonomous one:



every box = a headless `claude -p` agent · TASKS / PROGRESS / REVIEW logs are append-only

Results: LEAN formalization

- We formalize the **statements** of all 4 commutative-algebra questions [Cahen et al. '14]
- … and the **first proof** — problem P6

Problem ID	Lines of LEAN
4(b)	~4,200
20	~2,800
27(b)	~1,300
30(c)	~2,800
P6 (first proof)	~2,500

Thanks!

Paper: github.com/Pengbinghui/pipeline-math