Artificial intelligence for the working mathematician

Lecture 2: LLMs, and what it looks like in practice

Mini-course on emerging applications of AI in theoretical mathematics
ICMAT, Madrid, Spain

Damien Galant

Department of Mathematics
Brown University

Tuesday 15 September 2026

The class of neural networks

Fix a depth n, widths $d = d_0, d_1, \ldots, d_n = m$, and a function $\sigma : \mathbb{R} \to \mathbb{R}$ applied componentwise (for example $\sigma(t) = \max(t, 0)$). For parameters

$$\theta = (W_1, b_1, \ldots, W_n, b_n) \in \mathbb{R}^N, \qquad W_i \in \mathbb{R}^{d_i \times d_{i-1}}, \quad b_i \in \mathbb{R}^{d_i},$$

write $A_i x := W_i x + b_i$ and

$$f_\theta := A_n \circ \sigma \circ A_{n-1} \circ \cdots \circ \sigma \circ A_1 \ : \ \mathbb{R}^d \to \mathbb{R}^m.$$

The *class* is $\mathcal{N} := \{ f_\theta \mid \theta \in \mathbb{R}^N \}$.

The class of neural networks

Fix a depth n, widths $d = d_0, d_1, \ldots, d_n = m$, and a function $\sigma : \mathbb{R} \to \mathbb{R}$ applied componentwise (for example $\sigma(t) = \max(t, 0)$). For parameters

$$\theta = (W_1, b_1, \ldots, W_n, b_n) \in \mathbb{R}^N, \qquad W_i \in \mathbb{R}^{d_i \times d_{i-1}}, \quad b_i \in \mathbb{R}^{d_i},$$

write $A_i x := W_i x + b_i$ and

$$f_\theta := A_n \circ \sigma \circ A_{n-1} \circ \cdots \circ \sigma \circ A_1 \ : \ \mathbb{R}^d \to \mathbb{R}^m.$$

The *class* is $\mathcal{N} := \{ f_\theta \mid \theta \in \mathbb{R}^N \}$.

The one structural remark

$\mathcal{N}$ is cut out by finitely many parameters, but it is *not a linear subspace* of $C(\mathbb{R}^d, \mathbb{R}^m)$ — unlike polynomials of degree $\leq k$. Much of the difficulty comes from that.

What the class can represent

Universal approximation (Leshno–Pinkus et al., 1993)

For σ continuous and not a polynomial and $K \subset \mathbb{R}^d$ compact, the *union over all widths* k,

$$\left\{ x \mapsto \sum_{1 \le j \le k} c_j \, \sigma(w_j \cdot x + b_j) \;\middle|\; k \ge 1 \right\}$$

is dense in $C(K)$. *One hidden layer suffices* — but the width is unbounded, and nothing here gives an algorithm for finding the coefficients.

What the class can represent

Universal approximation (Leshno–Pinkus et al., 1993)

For σ continuous and not a polynomial and $K \subset \mathbb{R}^d$ compact, the *union over all widths* k,

$$\left\{ x \mapsto \sum_{1 \le j \le k} c_j \, \sigma(w_j \cdot x + b_j) \;\middle|\; k \ge 1 \right\}$$

is dense in $C(K)$. *One hidden layer suffices* — but the width is unbounded, and nothing here gives an algorithm for finding the coefficients.

Do not be impressed

So are polynomials, by Weierstrass. *Density is not the point, and never was*: what separates these families is not what they can approximate, but what happens when you *fit* them.

Training is a regression problem

Given a *sample* $(x_i, y_i)_{1 \leq i \leq M}$ and a *loss* $\ell : \mathbb{R}^m \times \mathbb{R}^m \to [0, \infty[$ — say $\ell(u, v) = |u - v|^2$ — minimise the empirical risk

$$\widehat{R}(\theta) := \frac{1}{M} \sum_{1 \leq i \leq M} \ell(f_\theta(x_i), y_i)$$

over $\theta \in \mathbb{R}^N$. Nobody solves this: one *approximately* optimises it by stochastic gradient methods, the gradient $\nabla_\theta \widehat{R}$ being computed by the chain rule — *backpropagation*.

Training is a regression problem

Given a *sample* $(x_i, y_i)_{1 \leq i \leq M}$ and a *loss* $\ell : \mathbb{R}^m \times \mathbb{R}^m \to [0, \infty[$ — say $\ell(u, v) = |u - v|^2$ — minimise the empirical risk

$$\widehat{R}(\theta) := \frac{1}{M} \sum_{1 \leq i \leq M} \ell(f_\theta(x_i), y_i)$$

over $\theta \in \mathbb{R}^N$. Nobody solves this: one *approximately* optimises it by stochastic gradient methods, the gradient $\nabla_\theta \widehat{R}$ being computed by the chain rule — *backpropagation*.

That is genuinely the whole of it

Nonconvex optimisation over a parametrised family. With $\ell(u, v) = |u - v|^2$ this is nonlinear least squares; a language model instead uses cross-entropy, i.e. maximum likelihood.

Training is a regression problem

Given a *sample* $(x_i, y_i)_{1 \leq i \leq M}$ and a *loss* $\ell : \mathbb{R}^m \times \mathbb{R}^m \to [0, \infty[$ — say $\ell(u, v) = |u - v|^2$ — minimise the empirical risk

$$\widehat{R}(\theta) := \frac{1}{M} \sum_{1 \leq i \leq M} \ell(f_\theta(x_i), y_i)$$

over $\theta \in \mathbb{R}^N$. Nobody solves this: one *approximately* optimises it by stochastic gradient methods, the gradient $\nabla_\theta \widehat{R}$ being computed by the chain rule — *backpropagation*.

That is genuinely the whole of it

Nonconvex optimisation over a parametrised family. With $\ell(u, v) = |u - v|^2$ this is nonlinear least squares; a language model instead uses cross-entropy, i.e. maximum likelihood.

So why does that work?

The questions that are actually open

- Why does *stochastic gradient descent*, on a wildly nonconvex problem with 10^{11}–10^{13} parameters, find anything good at all?

The questions that are actually open

- Why does *stochastic gradient descent*, on a wildly nonconvex problem with 10^{11}–10^{13} parameters, find anything good at all?
- Why does the result *generalise*, in a regime where the model has enough capacity to memorise its training data outright?

The questions that are actually open

- Why does *stochastic gradient descent*, on a wildly nonconvex problem with 10^{11}–10^{13} parameters, find anything good at all?
- Why does the result *generalise*, in a regime where the model has enough capacity to memorise its training data outright?
- Which of the apparent *abrupt* jumps in capability are real, and which are artefacts of the metric?

The questions that are actually open

- Why does *stochastic gradient descent*, on a wildly nonconvex problem with 10^{11}–10^{13} parameters, find anything good at all?
- Why does the result *generalise*, in a regime where the model has enough capacity to memorise its training data outright?
- Which of the apparent *abrupt* jumps in capability are real, and which are artefacts of the metric?

Honest answer

Nobody knows. There is real mathematics being done here — and if that is what attracts you, it is wide open — but the practice ran far ahead of the theory.

The questions that are actually open

- Why does *stochastic gradient descent*, on a wildly nonconvex problem with 10^{11}–10^{13} parameters, find anything good at all?
- Why does the result *generalise*, in a regime where the model has enough capacity to memorise its training data outright?
- Which of the apparent *abrupt* jumps in capability are real, and which are artefacts of the metric?

Honest answer

Nobody knows. There is real mathematics being done here — and if that is what attracts you, it is wide open — but the practice ran far ahead of the theory.

We are in the situation of thermodynamics before statistical mechanics: it works, it is used, and the explanation is missing.

Stage one: what a language model is

Fix a vocabulary of *tokens* (roughly, word fragments). A language model is a map

$$(t_1, \ldots, t_n) \longmapsto p_\theta(\,\cdot \mid t_1, \ldots, t_n),$$

a distribution over *the next token*; text is generated by sampling, appending and repeating. Pre-training fits θ by maximum likelihood on an enormous corpus, *at a scale no university can afford*.

Stage one: what a language model is

Fix a vocabulary of *tokens* (roughly, word fragments). A language model is a map

$$(t_1, \ldots, t_n) \longmapsto p_\theta(\,\cdot \mid t_1, \ldots, t_n),$$

a distribution over *the next token*; text is generated by sampling, appending and repeating. Pre-training fits θ by maximum likelihood on an enormous corpus, *at a scale no university can afford*.

Still an f_θ, fitted as before — but the architecture is a *transformer* (Vaswani et al., 2017), built on *attention*: each token's representation is rebuilt as a weighted average of the *earlier* ones, with weights computed from the text.

Stage one: what a language model is

Fix a vocabulary of *tokens* (roughly, word fragments). A language model is a map

$$(t_1, \ldots, t_n) \longmapsto p_\theta(\,\cdot \mid t_1, \ldots, t_n),$$

a distribution over *the next token*; text is generated by sampling, appending and repeating. Pre-training fits θ by maximum likelihood on an enormous corpus, *at a scale no university can afford*.
Still an f_θ, fitted as before — but the architecture is a *transformer* (Vaswani et al., 2017), built on *attention*: each token's representation is rebuilt as a weighted average of the *earlier* ones, with weights computed from the text.

The uncomfortable observation

Guess the next word has no business producing anything interesting, yet past a certain scale it does. *The best way to predict the next word is to understand the text.*

Scaling laws

The reason anyone kept building larger models is an *empirical* regularity.

Scaling laws

The reason anyone kept building larger models is an *empirical* regularity.

Kaplan et al., 2020

The cross-entropy loss falls as a *power law* in the number of parameters, the size of the dataset and the compute spent — with some trends holding over *more than seven orders of magnitude*. Architectural details such as width and depth matter remarkably little across a wide range.

Scaling laws

The reason anyone kept building larger models is an *empirical* regularity.

Kaplan et al., 2020

The cross-entropy loss falls as a *power law* in the number of parameters, the size of the dataset and the compute spent — with some trends holding over *more than seven orders of magnitude*. Architectural details such as width and depth matter remarkably little across a wide range.

So the loss became *predictable* before the run was launched — which turned training into an engineering programme rather than a gamble.

Scaling laws

The reason anyone kept building larger models is an *empirical* regularity.

Kaplan et al., 2020

The cross-entropy loss falls as a *power law* in the number of parameters, the size of the dataset and the compute spent — with some trends holding over *more than seven orders of magnitude*. Architectural details such as width and depth matter remarkably little across a wide range.

So the loss became *predictable* before the run was launched — which turned training into an engineering programme rather than a gamble.

And nobody derived it

These are fitted curves. *There is no argument from first principles saying the exponent should be what it is.*

Stage two: it is not trained the way you use it

Pre-training yields a *document continuer*, not an assistant: ask the raw model a question, and a likely continuation is *a second question*.

Stage two: it is not trained the way you use it

Pre-training yields a *document continuer*, not an assistant: ask the raw model a question, and a likely continuation is *a second question*.

Making it useful is a separate stage — *RLHF*, reinforcement learning from human feedback (Christiano et al., 2017; Ouyang et al., 2022):

(i) humans *write* good answers; fine-tune on them;

(ii) humans *rank* candidate answers; fit a *reward model* r_ϕ to the comparisons;

(iii) optimise the model *against* r_ϕ.

Stage three: replace the human by a checker

In *RLVR* — reinforcement learning from verifiable rewards — the reward is not a model of anyone's taste. It is *did the answer check out*: a proof assistant, a test suite, a numerical certificate.

Stage three: replace the human by a checker

In *RLVR* — reinforcement learning from verifiable rewards — the reward is not a model of anyone's taste. It is *did the answer check out*: a proof assistant, a test suite, a numerical certificate.

Why this step is decisive for us

It works exactly where correctness is cheap to verify: *mathematics and code*. Mathematics is not a niche application — it is one of the two things these systems are trained hardest on.

Stage three: replace the human by a checker

In *RLVR* — reinforcement learning from verifiable rewards — the reward is not a model of anyone's taste. It is *did the answer check out*: a proof assistant, a test suite, a numerical certificate.

Why this step is decisive for us

It works exactly where correctness is cheap to verify: *mathematics and code*. Mathematics is not a niche application — it is one of the two things these systems are trained hardest on.

What the model is rewarded for is the *whole working*, not the final line. So it learns to produce a long *chain of thought* — and to spend more test-time compute when the problem is harder.

Stage three: replace the human by a checker

In *RLVR* — reinforcement learning from verifiable rewards — the reward is not a model of anyone's taste. It is *did the answer check out*: a proof assistant, a test suite, a numerical certificate.

Why this step is decisive for us

It works exactly where correctness is cheap to verify: *mathematics and code*. Mathematics is not a niche application — it is one of the two things these systems are trained hardest on.

What the model is rewarded for is the *whole working*, not the final line. So it learns to produce a long *chain of thought* — and to spend more test-time compute when the problem is harder.

The chain, in one line: predict the next token $\longrightarrow$ satisfy a human judge $\longrightarrow$ satisfy a checker.

Chain of thought: three caveats

It is no longer a prompt. Chain-of-thought prompting (Wei et al., 2022) and "let us think step by step" (Kojima et al., 2022) were *tricks* you applied to a model that would otherwise answer immediately.
On current models it is trained in: *you cannot switch it off, and you no longer ask for it*.

Chain of thought: three caveats

It is no longer a prompt. Chain-of-thought prompting (Wei et al., 2022) and "let us think step by step" (Kojima et al., 2022) were *tricks* you applied to a model that would otherwise answer immediately.
On current models it is trained in: *you cannot switch it off, and you no longer ask for it*.

It is not one chain. Frontier systems spend the budget on *search* — many attempts, tool calls, self-checks — and show only a *summary* of the trace.

Chain of thought: three caveats

It is no longer a prompt. Chain-of-thought prompting (Wei et al., 2022) and "let us think step by step" (Kojima et al., 2022) were *tricks* you applied to a model that would otherwise answer immediately.
On current models it is trained in: *you cannot switch it off, and you no longer ask for it*.

It is not one chain. Frontier systems spend the budget on *search* — many attempts, tool calls, self-checks — and show only a *summary* of the trace.

Not a proof — and not even the computation

A model can reach the right answer for reasons its stated reasoning does not contain (Turpin et al., 2023). *Read the chain as a draft to be checked, never as evidence.*

Chain of thought: three caveats

It is no longer a prompt. Chain-of-thought prompting (Wei et al., 2022) and "let us think step by step" (Kojima et al., 2022) were *tricks* you applied to a model that would otherwise answer immediately.
On current models it is trained in: *you cannot switch it off, and you no longer ask for it*.

It is not one chain. Frontier systems spend the budget on *search* — many attempts, tool calls, self-checks — and show only a *summary* of the trace.

Not a proof — and not even the computation

A model can reach the right answer for reasons its stated reasoning does not contain (Turpin et al., 2023). *Read the chain as a draft to be checked, never as evidence.*

Likewise for the reward: optimise a proxy hard enough and you get what the proxy measures — sycophancy from a human judge, a passed test suite from a checker.

Which machine is which

They are not all "AI", and they do not do the same job.

	Learns	**Searches**	**Certifies**
Computer algebra	no	no	a little
Interval arithmetic	no	no	yes
SAT solver	no	yes	yes
Proof assistant	no	a little	yes
Domain-specific network	yes	yes	no
Language model	yes	no	no
Agent (model + tools)	yes	yes	no

Which machine is which

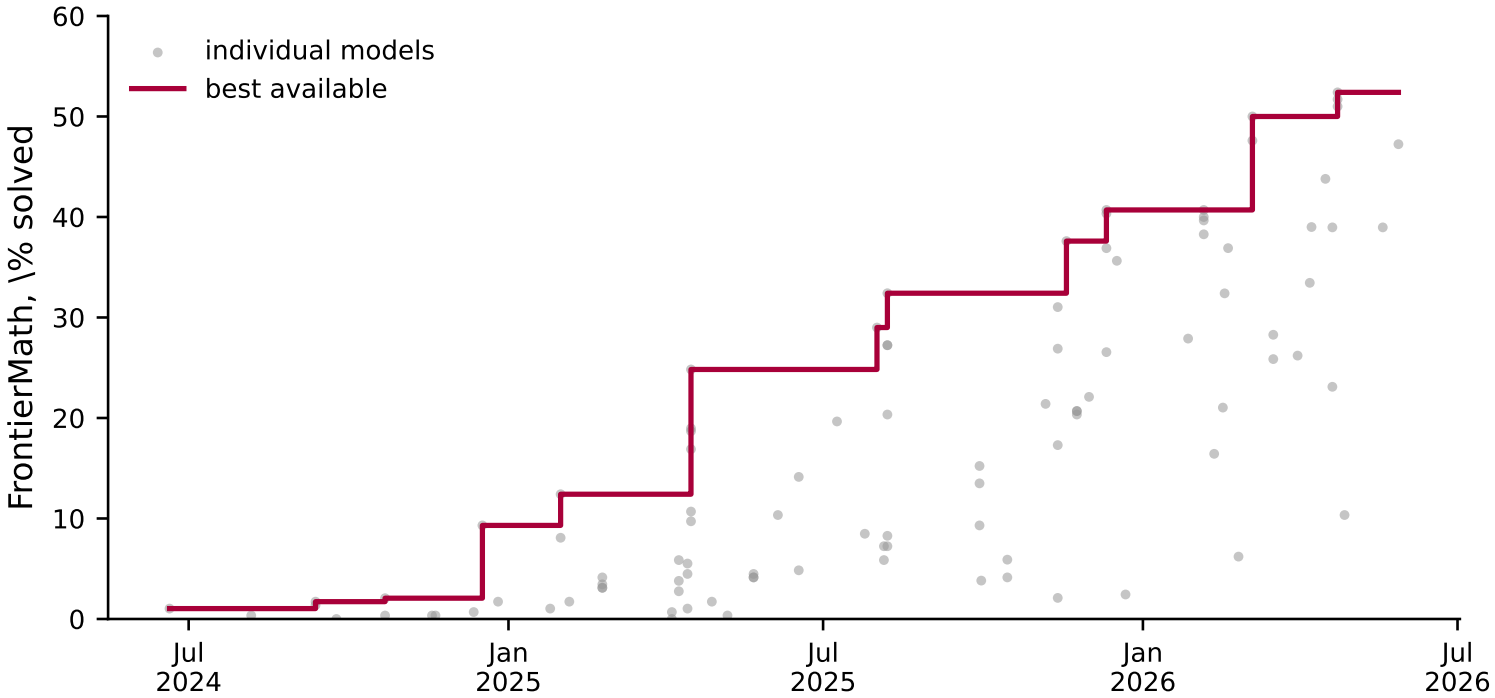
They are not all "AI", and they do not do the same job.

	Learns	**Searches**	**Certifies**
Computer algebra	no	no	a little
Interval arithmetic	no	no	yes
SAT solver	no	yes	yes
Proof assistant	no	a little	yes
Domain-specific network	yes	yes	no
Language model	yes	no	no
Agent (model + tools)	yes	yes	no

The line that matters

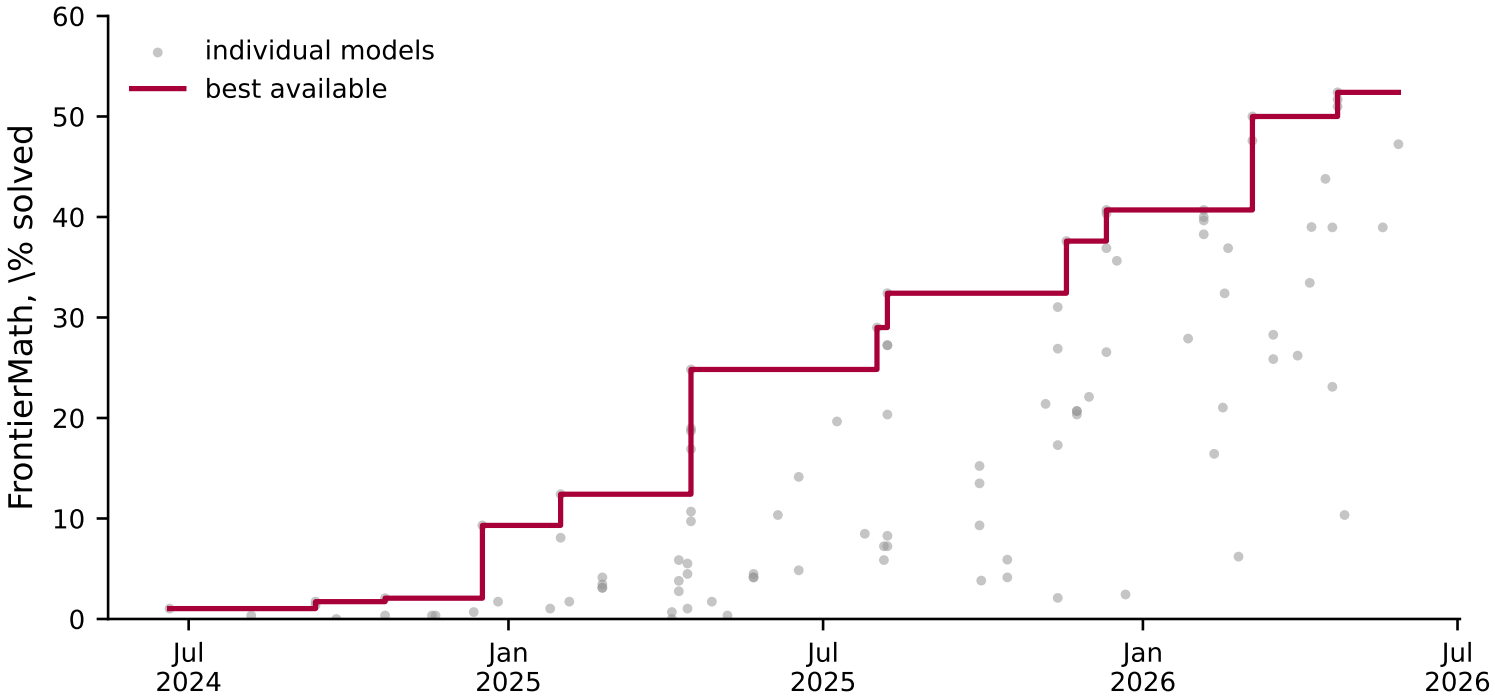
Nothing that learns also certifies. Every certificate in this lecture came from a classical row — which is exactly why an agent's job is to *call* those rows.

One benchmark, over two years



FrontierMath tiers 1–3 (v2): problems written by mathematicians and kept private, solved in a Python-enabled harness. Advanced undergraduate up to early research level — the distinctly research-level problems are tier 4. Data: Epoch AI, *AI Benchmarking Hub* (CC BY 4.0).

One benchmark, over two years



FrontierMath tiers 1–3 (v2): problems written by mathematicians and kept private, solved in a Python-enabled harness. Advanced undergraduate up to early research level — the distinctly research-level problems are tier 4. Data: Epoch AI, *AI Benchmarking Hub* (CC BY 4.0).

From 1% in June 2024 to 52% in April 2026. Horizontal axis: release date; each dot is one model–configuration pair.

The IMO as a thermometer

Year	System	Result
2024	AlphaProof + AlphaGeometry 2 (formal input, days of compute)	28/42, silver
2025	OpenAI, Google DeepMind (natural language, in the time limit)	*35/42, gold*
2026†	seven models, independent runs	*42/42*: one first pass

† Not an official entry: published problems, autonomous runs, graded by AI agents. One model scored 42/42 first pass; two reached it after being shown grader reports.
In 2025, only Google's scripts were graded by IMO coordinators.

The IMO as a thermometer

Year	System	Result
2024	AlphaProof + AlphaGeometry 2 (formal input, days of compute)	28/42, silver
2025	OpenAI, Google DeepMind (natural language, in the time limit)	*35/42, gold*
2026†	seven models, independent runs	*42/42*: one first pass

† Not an official entry: published problems, autonomous runs, graded by AI agents. One model scored 42/42 first pass; two reached it after being shown grader reports.
In 2025, only Google's scripts were graded by IMO coordinators.

What to read into this

Not "machines are better than us": IMO problems are short and known to have a solution. What matters is *the speed of progress* — and that the test conditions have grown steadily more agentic.

The IMO as a thermometer

Year	System	Result
2024	AlphaProof + AlphaGeometry 2 (formal input, days of compute)	28/42, silver
2025	OpenAI, Google DeepMind (natural language, in the time limit)	*35/42, gold*
2026†	seven models, independent runs	*42/42*: one first pass

† Not an official entry: published problems, autonomous runs, graded by AI agents. One model scored 42/42 first pass; two reached it after being shown grader reports.
In 2025, only Google's scripts were graded by IMO coordinators.

What to read into this

Not "machines are better than us": IMO problems are short and known to have a solution. What matters is *the speed of progress* — and that the test conditions have grown steadily more agentic.

An opinion formed eighteen months ago is about a different technology.

Where this stands today

- **Competition problems**: as we just saw, largely within reach. Benchmarks now have to be research-level to be informative.

Where this stands today

- **Competition problems**: as we just saw, largely within reach. Benchmarks now have to be research-level to be informative.
- **Small research contributions** — a constant improved, a case closed, a counterexample supplied — happen *routinely enough that they are no longer news*. *And occasionally far more than that: next slide.*

Where this stands today

- **Competition problems**: as we just saw, largely within reach. Benchmarks now have to be research-level to be informative.
- **Small research contributions** — a constant improved, a case closed, a counterexample supplied — happen *routinely enough that they are no longer news*. *And occasionally far more than that: next slide.*
- **The Erdős problems episode** (2025): several problems on `erdosproblems.com` were announced as "solved by AI" — and it turned out the model had *found solutions already in the literature*.

  Overstated, and corrected. But note what it actually did: *it read the literature.*

Where this stands today

- **Competition problems**: as we just saw, largely within reach. Benchmarks now have to be research-level to be informative.
- **Small research contributions** — a constant improved, a case closed, a counterexample supplied — happen *routinely enough that they are no longer news*. *And occasionally far more than that: next slide.*
- **The Erdős problems episode** (2025): several problems on `erdosproblems.com` were announced as "solved by AI" — and it turned out the model had *found solutions already in the literature*.

  Overstated, and corrected. But note what it actually did: *it read the literature.*

Both the hype and the correction are part of the picture. I will come back to the hype on Wednesday.

The unit distance conjecture (Erdős, 1946) is false

The question: among n points in the plane, how many pairs can be at distance exactly 1? Write $U(n)$ for the maximum. Erdős proved $U(n) \geq n^{1+\Omega(1/\log\log n)}$ and conjectured

$$U(n) \leq n^{1+o(1)}.$$

The unit distance conjecture (Erdős, 1946) is false

The question: among n points in the plane, how many pairs can be at distance exactly 1? Write $U(n)$ for the maximum. Erdős proved $U(n) \geq n^{1+\Omega(1/\log\log n)}$ and conjectured

$$U(n) \leq n^{1+o(1)}.$$

In **May 2026** an internal OpenAI model produced, *in one shot*, a counterexample: for some $\varepsilon > 0$ there are point sets $\mathcal{P}$ with $|\mathcal{P}| \to \infty$ and at least $|\mathcal{P}|^{1+\varepsilon}$ unit distances.

The unit distance conjecture (Erdős, 1946) is false

The question: among n points in the plane, how many pairs can be at distance exactly 1? Write $U(n)$ for the maximum. Erdős proved $U(n) \geq n^{1+\Omega(1/\log\log n)}$ and conjectured

$$U(n) \leq n^{1+o(1)}.$$

In **May 2026** an internal OpenAI model produced, *in one shot*, a counterexample: for some $\varepsilon > 0$ there are point sets $\mathcal{P}$ with $|\mathcal{P}| \to \infty$ and at least $|\mathcal{P}|^{1+\varepsilon}$ unit distances.

What it used, and who checked it

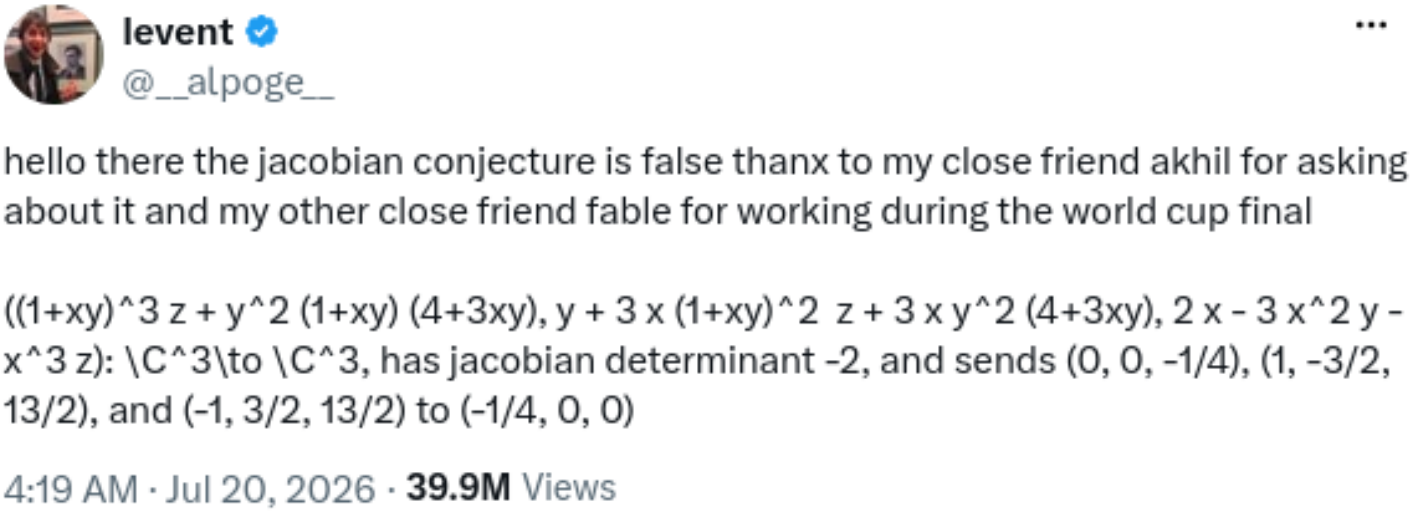
Infinite class field towers of Golod–Shafarevich type: *algebraic number theory, for a question about the plane*. Nine mathematicians — Alon, Bloom, Gowers, Litt, Sawin, Shankar, Tsimerman, Wang, Wood — then published *"a short, digested, human-verified version"*.

The Jacobian conjecture (Keller, 1939) is false

The conjecture: every *polynomial* map $F : \mathbb{C}^n \to \mathbb{C}^n$ whose Jacobian determinant is a nonzero constant has a polynomial inverse. In **July 2026** Alpöge announced a polynomial counterexample in dimension three, found with AI assistance.

The Jacobian conjecture (Keller, 1939) is false

The conjecture: every *polynomial* map $F : \mathbb{C}^n \to \mathbb{C}^n$ whose Jacobian determinant is a nonzero constant has a polynomial inverse. In **July 2026** Alpöge announced a polynomial counterexample in dimension three, found with AI assistance.



levent
@__alpoge__

hello there the jacobian conjecture is false thanx to my close friend akhil for asking about it and my other close friend fable for working during the world cup final

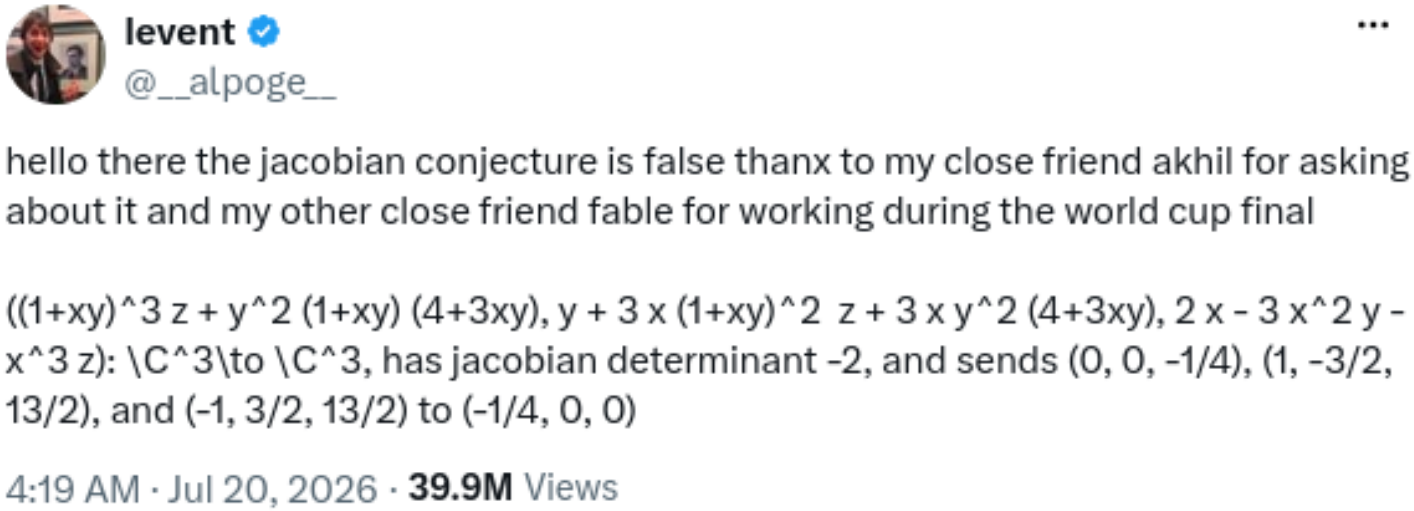
((1+xy)^3 z + y^2 (1+xy) (4+3xy), y + 3 x (1+xy)^2 z + 3 x y^2 (4+3xy), 2 x - 3 x^2 y - x^3 z): \C^3\to \C^3, has jacobian determinant -2, and sends (0, 0, -1/4), (1, -3/2, 13/2), and (-1, 3/2, 13/2) to (-1/4, 0, 0)

4:19 AM · Jul 20, 2026 · **39.9M** Views

Three points of $\mathbb{C}^3$, one image.

The Jacobian conjecture (Keller, 1939) is false

The conjecture: every *polynomial* map $F : \mathbb{C}^n \to \mathbb{C}^n$ whose Jacobian determinant is a nonzero constant has a polynomial inverse. In **July 2026** Alpöge announced a polynomial counterexample in dimension three, found with AI assistance.



levent
@__alpoge__

hello there the jacobian conjecture is false thanx to my close friend akhil for asking about it and my other close friend fable for working during the world cup final

((1+xy)^3 z + y^2 (1+xy) (4+3xy), y + 3 x (1+xy)^2 z + 3 x y^2 (4+3xy), 2 x - 3 x^2 y - x^3 z): \C^3\to \C^3, has jacobian determinant -2, and sends (0, 0, -1/4), (1, -3/2, 13/2), and (-1, 3/2, 13/2) to (-1/4, 0, 0)

4:19 AM · Jul 20, 2026 · **39.9M** Views

Three points of $\mathbb{C}^3$, one image.

An infinite family followed the next day (Gallagher), then a geometric explanation (Speyer). False for $n \geq 3$; *$n = 2$ remains open.*

“It is just a stochastic parrot”

The phrase is from a 2021 paper — which also raised cost, bias and concentration of resources. Here I mean only the narrow reading: *these systems merely recombine what they memorised*.

“It is just a stochastic parrot”

The phrase is from a 2021 paper — which also raised cost, bias and concentration of resources. Here I mean only the narrow reading: *these systems merely recombine what they memorised*.

Why I no longer accept it

Memorisation is a testable hypothesis, and it fails:

- they solve problems written after their training data;
- they produce *correct* arguments for statements that appear nowhere, as verified by us;
- they transfer between fields in ways a lookup table cannot.

"It is just a stochastic parrot"

The phrase is from a 2021 paper — which also raised cost, bias and concentration of resources. Here I mean only the narrow reading: *these systems merely recombine what they memorised*.

Why I no longer accept it

Memorisation is a testable hypothesis, and it fails:

- they solve problems written after their training data;
- they produce *correct* arguments for statements that appear nowhere, as verified by us;
- they transfer between fields in ways a lookup table cannot.

As a complete account of what these systems do, memorisation is *empirically inadequate*. The paper's other claims are untouched by that.

Mathematics is an unusually favourable target

- **Right and wrong are well defined.** Often we can check cheaply — exact answers, tests, certificates, formal proofs — though not a forty-page informal argument.

Mathematics is an unusually favourable target

- **Right and wrong are well defined.** Often we can check cheaply — exact answers, tests, certificates, formal proofs — though not a forty-page informal argument.
- **The data is abundant and machine-readable.** Decades of LaTeX written by careful people — though licensing and quality are uneven.

Mathematics is an unusually favourable target

- **Right and wrong are well defined.** Often we can check cheaply — exact answers, tests, certificates, formal proofs — though not a forty-page informal argument.
- **The data is abundant and machine-readable.** Decades of LaTeX written by careful people — though licensing and quality are uneven.
- **We can manufacture more.** Synthetic problems with known answers, in unlimited quantity — this is exactly how AlphaGeometry was trained.

Mathematics is an unusually favourable target

- **Right and wrong are well defined.** Often we can check cheaply — exact answers, tests, certificates, formal proofs — though not a forty-page informal argument.
- **The data is abundant and machine-readable.** Decades of LaTeX written by careful people — though licensing and quality are uneven.
- **We can manufacture more.** Synthetic problems with known answers, in unlimited quantity — this is exactly how AlphaGeometry was trained.
- **There is a formal target.** Lean, Coq, Isabelle: a compiler that says yes or no. An unambiguous reward signal, which is a rare luxury.

Mathematics is an unusually favourable target

- **Right and wrong are well defined.** Often we can check cheaply — exact answers, tests, certificates, formal proofs — though not a forty-page informal argument.
- **The data is abundant and machine-readable.** Decades of LaTeX written by careful people — though licensing and quality are uneven.
- **We can manufacture more.** Synthetic problems with known answers, in unlimited quantity — this is exactly how AlphaGeometry was trained.
- **There is a formal target.** Lean, Coq, Isabelle: a compiler that says yes or no. An unambiguous reward signal, which is a rare luxury.

Consequence, whether we like it or not

We are not bystanders in this. *Mathematics is one of the main proving grounds*, and the tools are being shaped around what we do.

Six reasons, from my own last twelve months

1. Experiments became cheap.
2. Finding things in the literature became easy.
3. The boring part of the work shrank.
4. You are never stuck alone at 2 a.m.
5. Computation-heavy mathematics became accessible.
6. You can build a tool for *your* problem.

Six reasons, from my own last twelve months

1. Experiments became cheap.
2. Finding things in the literature became easy.
3. The boring part of the work shrank.
4. You are never stuck alone at 2 a.m.
5. Computation-heavy mathematics became accessible.
6. You can build a tool for *your* problem.

Not one of these requires you to know anything about machine learning, or to change what you work on.

1–2. Experiments and literature

Experiments

"I wonder what happens to the second eigenvalue if I stretch this edge" used to mean two days of writing throw-away code. *It now means fifteen minutes.*

The consequence is not that you compute more: it is that you *ask more questions*, because asking got cheap.

1–2. Experiments and literature

Experiments

"I wonder what happens to the second eigenvalue if I stretch this edge" used to mean two days of writing throw-away code. *It now means fifteen minutes.*

The consequence is not that you compute more: it is that you *ask more questions*, because asking got cheap.

Literature

"Surely someone has proved this — but I do not know the word for it, so I cannot search for it."

The most reliable win, in my experience. *Google needs the keyword; the model needs only the idea.*

1–2. Experiments and literature

Experiments

"I wonder what happens to the second eigenvalue if I stretch this edge" used to mean two days of writing throw-away code. *It now means fifteen minutes.*

The consequence is not that you compute more: it is that you *ask more questions*, because asking got cheap.

Literature

"Surely someone has proved this — but I do not know the word for it, so I cannot search for it."

The most reliable win, in my experience. *Google needs the keyword; the model needs only the idea.*

Caveat: it also invents references. Always check.

3–4. The boring part, and the 2 a.m. problem

The boring part

Tedious but routine lemmas. Turning a computation into a figure. Rewriting the introduction for the fourth referee.

More time for the real thing. This is not a small effect: measure how much of your week it actually is.

3–4. The boring part, and the 2 a.m. problem

The boring part

Tedious but routine lemmas. Turning a computation into a figure. Rewriting the introduction for the fourth referee.

More time for the real thing. This is not a small effect: measure how much of your week it actually is.

You will never walk alone

There is now always someone to ask — at midnight, on a Sunday, about a stupid question, without embarrassment.

5–6. Computational mathematics, and tools of your own

The barrier dropped

A rigorous computer-assisted proof used to require a second craft: interval arithmetic, C, parallelism, error analysis.

That barrier is now *much lower*. The mathematics is unchanged — you still have to design the reduction — but the implementation is no longer the reason you abandon the project.

5–6. Computational mathematics, and tools of your own

The barrier dropped

A rigorous computer-assisted proof used to require a second craft: interval arithmetic, C, parallelism, error analysis.

That barrier is now *much lower*. The mathematics is unchanged — you still have to design the reduction — but the implementation is no longer the reason you abandon the project.

Problem-specific tools

In my own current project I have a *website* that monitors the state of a large computer-assisted proof: which claims are proved, which are open, which certificates are stale.

I would never have written that website. *It took an afternoon.*

5–6. Computational mathematics, and tools of your own

The barrier dropped

A rigorous computer-assisted proof used to require a second craft: interval arithmetic, C, parallelism, error analysis.

That barrier is now *much lower*. The mathematics is unchanged — you still have to design the reduction — but the implementation is no longer the reason you abandon the project.

Problem-specific tools

In my own current project I have a *website* that monitors the state of a large computer-assisted proof: which claims are proved, which are open, which certificates are stale.

I would never have written that website. *It took an afternoon.*

Do we not all secretly want to turn our research into a video game?

The three shapes of tool

1. **The chat window.** You ask, it answers. Good for: understanding, literature, reformulating, checking your own reasoning.

The three shapes of tool

1. **The chat window.** You ask, it answers. Good for: understanding, literature, reformulating, checking your own reasoning.
2. **The agent.** It has a terminal, a file system, and a loop. Good for: anything involving code, data, or many steps. *This is the one most mathematicians have never tried, and it is the one that changed my work.*

The three shapes of tool

1. **The chat window.** You ask, it answers. Good for: understanding, literature, reformulating, checking your own reasoning.
2. **The agent.** It has a terminal, a file system, and a loop. Good for: anything involving code, data, or many steps. *This is the one most mathematicians have never tried, and it is the one that changed my work.*
3. **Search at scale.** Machine learning inside a search loop, generating and testing thousands of candidates. Good for: explicit extremal problems, bounds, constructions.

The three shapes of tool

1. **The chat window.** You ask, it answers. Good for: understanding, literature, reformulating, checking your own reasoning.
2. **The agent.** It has a terminal, a file system, and a loop. Good for: anything involving code, data, or many steps. *This is the one most mathematicians have never tried, and it is the one that changed my work.*
3. **Search at scale.** Machine learning inside a search loop, generating and testing thousands of candidates. Good for: explicit extremal problems, bounds, constructions.

Rough rule

Chat for *thinking*. Agent for *doing*. Search for *finding an object you cannot guess*.

How to ask: what actually matters

Forget "prompt engineering". Three things matter, and they are the same three that matter when asking a colleague.

1. **Give the context.** Paste the definitions and the paper; say what you tried and why it failed. *The most common mistake is under-specifying.*

How to ask: what actually matters

Forget "prompt engineering". Three things matter, and they are the same three that matter when asking a colleague.

1. **Give the context.** Paste the definitions and the paper; say what you tried and why it failed. *The most common mistake is under-specifying.*
2. **Ask for the argument, not the answer.** "Prove or disprove" invites a confident guess. "Give me the argument, and tell me which step you are least sure of" does not.

How to ask: what actually matters

Forget "prompt engineering". Three things matter, and they are the same three that matter when asking a colleague.

1. **Give the context.** Paste the definitions and the paper; say what you tried and why it failed. *The most common mistake is under-specifying.*
2. **Ask for the argument, not the answer.** "Prove or disprove" invites a confident guess. "Give me the argument, and tell me which step you are least sure of" does not.
3. **Make it adversarial.** "Now attack that proof. Where is it weakest?" *Models are much better at criticising than at being right first time.*

How to ask: what actually matters

Forget "prompt engineering". Three things matter, and they are the same three that matter when asking a colleague.

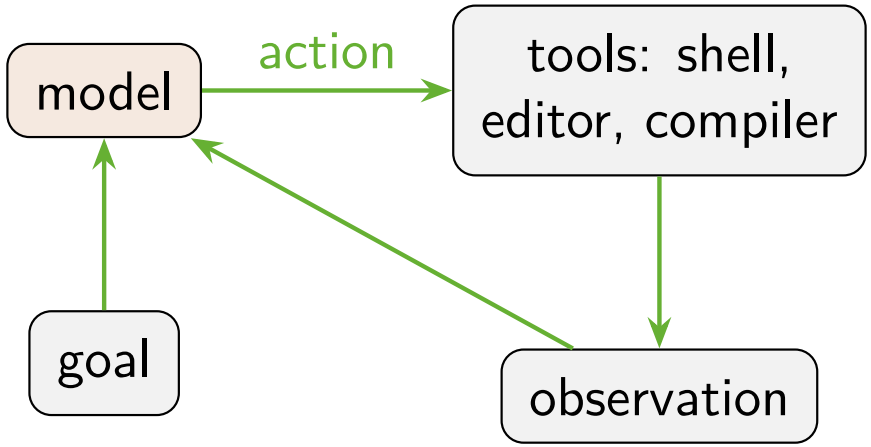
1. **Give the context.** Paste the definitions and the paper; say what you tried and why it failed. *The most common mistake is under-specifying.*
2. **Ask for the argument, not the answer.** "Prove or disprove" invites a confident guess. "Give me the argument, and tell me which step you are least sure of" does not.
3. **Make it adversarial.** "Now attack that proof. Where is it weakest?" *Models are much better at criticising than at being right first time.*

And one rule

Never accept a reference you have not opened. This is where these systems still lie most fluently.

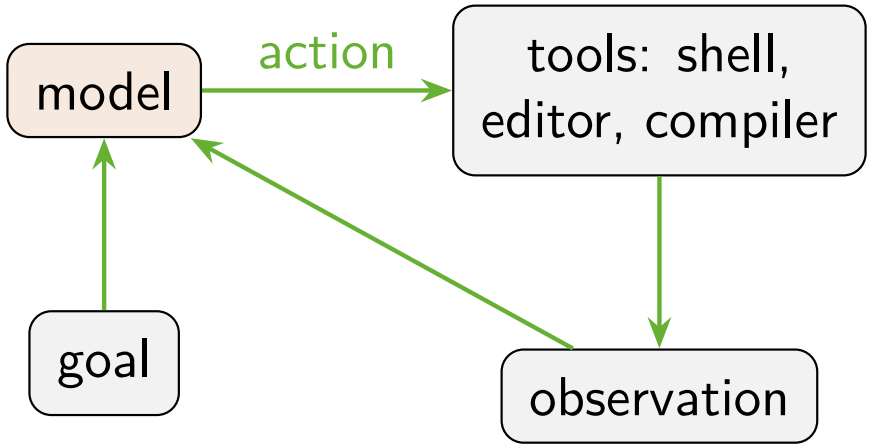
Agents: models with hands

A model alone can only emit text. Give it a loop and some tools:



Agents: models with hands

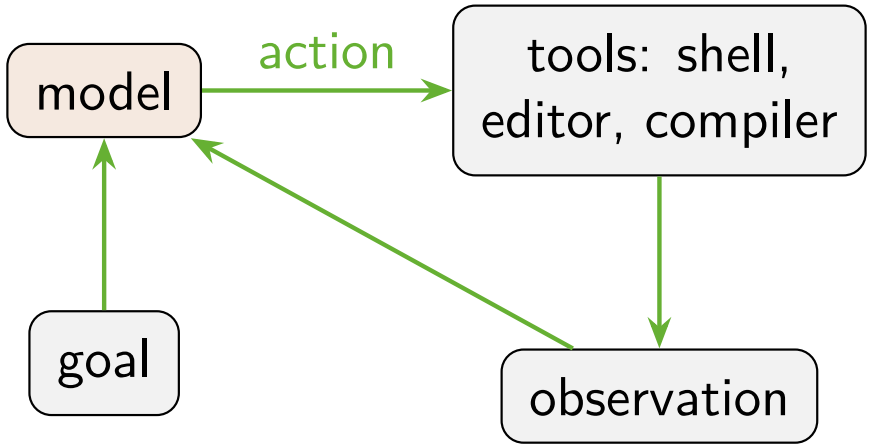
A model alone can only emit text. Give it a loop and some tools:



It writes a program, runs it, reads the error, fixes it — and can run *for hours without you*.

Agents: models with hands

A model alone can only emit text. Give it a loop and some tools:



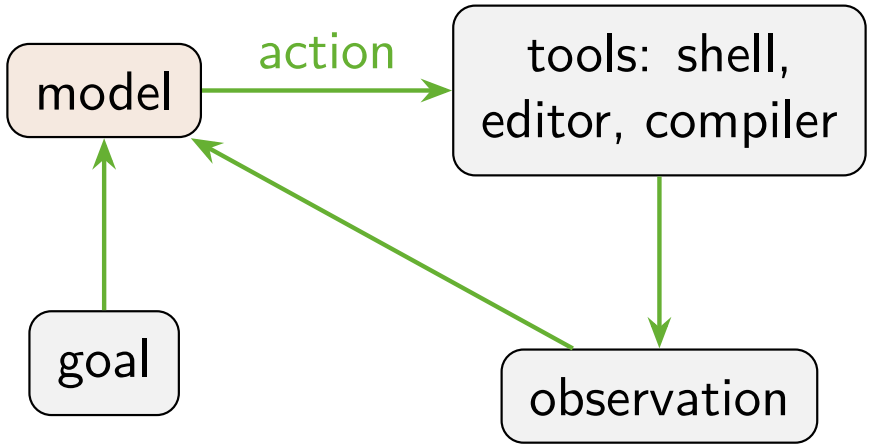
It writes a program, runs it, reads the error, fixes it — and can run *for hours without you*.

A frequent misunderstanding

"It cannot even multiply two large numbers." Correct, and irrelevant: *neither can you*. It calls a computer algebra system, exactly as you would.

Agents: models with hands

A model alone can only emit text. Give it a loop and some tools:



It writes a program, runs it, reads the error, fixes it — and can run *for hours without you*.

A frequent misunderstanding

"It cannot even multiply two large numbers." Correct, and irrelevant: *neither can you*. It calls a computer algebra system, exactly as you would.

The rest of today is almost entirely about this.

What an agent changes for us

The loop — write a file, run it, read the error, repeat — runs for minutes or hours. *You are not in it.* You come back to a result.

What an agent changes for us

The loop — write a file, run it, read the error, repeat — runs for minutes or hours. *You are not in it.* You come back to a result.

Why this matters for us specifically

Technical skill is less of a barrier than it has ever been. What is scarce is knowing *what to compute* and *whether the answer is plausible* — which is exactly what a mathematician has and a programmer does not.

Live demonstration: a tool for one problem

The project

Alvaro Carballeira, Damien Galant, Javier Gómez-Serrano, *The energy of fractional Allen–Cahn layers in dimension one*, arXiv:2608.06226.

Strict decrease of the energy is proved *with computer assistance* — finitely many inequalities, checked in *interval arithmetic*.

Live demonstration: a tool for one problem

The project

Alvaro Carballeira, Damien Galant, Javier Gómez-Serrano, *The energy of fractional Allen–Cahn layers in dimension one*, arXiv:2608.06226.

Strict decrease of the energy is proved *with computer assistance* — finitely many inequalities, checked in *interval arithmetic*.

Such a proof has many moving parts: claims, certificates, parameter ranges, dependencies, and what is stale after a change.

Live demonstration: a tool for one problem

The project

Alvaro Carballeira, Damien Galant, Javier Gómez-Serrano, *The energy of fractional Allen–Cahn layers in dimension one*, arXiv:2608.06226.

Strict decrease of the energy is proved *with computer assistance* — finitely many inequalities, checked in *interval arithmetic*.

Such a proof has many moving parts: claims, certificates, parameter ranges, dependencies, and what is stale after a change.

The tool

A dashboard: every claim, its status, its dependencies, and what breaks if a constant changes.

Nobody would fund this. Nobody would write this.

The discipline that makes this safe

Everything above is worthless without the following habits.

1. **Nothing enters a paper unchecked.** Not a reference, not a constant, not a lemma.

The discipline that makes this safe

Everything above is worthless without the following habits.

1. **Nothing enters a paper unchecked.** Not a reference, not a constant, not a lemma.
2. **Make the machine check itself.** Ask for tests. Ask for the computation to be redone by a second, different method. *Ask for it in interval arithmetic* — then the output is a theorem, not a number.

The discipline that makes this safe

Everything above is worthless without the following habits.

1. **Nothing enters a paper unchecked.** Not a reference, not a constant, not a lemma.
2. **Make the machine check itself.** Ask for tests. Ask for the computation to be redone by a second, different method. *Ask for it in interval arithmetic* — then the output is a theorem, not a number.
3. **Use a second model as a referee.** Different systems fail differently, and an adversarial review catches a surprising amount.

The discipline that makes this safe

Everything above is worthless without the following habits.

1. **Nothing enters a paper unchecked.** Not a reference, not a constant, not a lemma.
2. **Make the machine check itself.** Ask for tests. Ask for the computation to be redone by a second, different method. *Ask for it in interval arithmetic* — then the output is a theorem, not a number.
3. **Use a second model as a referee.** Different systems fail differently, and an adversarial review catches a surprising amount.
4. **Keep the reduction human.** You decide what is proved and how it reduces to a finite computation. That part is not delegable.

The discipline that makes this safe

Everything above is worthless without the following habits.

1. **Nothing enters a paper unchecked.** Not a reference, not a constant, not a lemma.
2. **Make the machine check itself.** Ask for tests. Ask for the computation to be redone by a second, different method. *Ask for it in interval arithmetic* — then the output is a theorem, not a number.
3. **Use a second model as a referee.** Different systems fail differently, and an adversarial review catches a surprising amount.
4. **Keep the reduction human.** You decide what is proved and how it reduces to a finite computation. That part is not delegable.

This is exactly the culture the interval arithmetic community built over thirty years. *We should steal it wholesale.*

Finding the conjecture, not the proof

You suspect that two families of invariants of the same objects are related.
You cannot see how.

Finding the conjecture, not the proof

You suspect that two families of invariants of the same objects are related. You cannot see how.

The method (Davies et al., Nature 2021)

Train a network to predict one from the other. If it succeeds, *a relation exists*. Then *saliency* tells you which few variables carry it — and you go and prove a statement about those.

Finding the conjecture, not the proof

You suspect that two families of invariants of the same objects are related. You cannot see how.

The method (Davies et al., Nature 2021)

Train a network to predict one from the other. If it succeeds, *a relation exists*. Then *saliency* tells you which few variables carry it — and you go and prove a statement about those.

Two theorems came out this way:

- the signature of a hyperbolic knot, from its cusp geometry (with Juhász and Lackenby);
- structure in Bruhat intervals, towards combinatorial invariance for Kazhdan–Lusztig polynomials (with Williamson).

Finding the conjecture, not the proof

You suspect that two families of invariants of the same objects are related. You cannot see how.

The method (Davies et al., Nature 2021)

Train a network to predict one from the other. If it succeeds, *a relation exists*. Then *saliency* tells you which few variables carry it — and you go and prove a statement about those.

Two theorems came out this way:

- the signature of a hyperbolic knot, from its cusp geometry (with Juhász and Lackenby);
- structure in Bruhat intervals, towards combinatorial invariance for Kazhdan–Lusztig polynomials (with Williamson).

The network proved nothing. It said which variables mattered.

Learning what a good construction looks like

Many combinatorial problems are: *find the best object in a huge discrete space*. Local search finds good ones, then gets stuck.

Learning what a good construction looks like

Many combinatorial problems are: *find the best object in a huge discrete space*. Local search finds good ones, then gets stuck.

PatternBoost (Charton, Ellenberg, Wagner, Williamson, 2024)

Alternate two phases: a classical local search produces many good constructions, then *a transformer is trained on the best of them* and its samples seed the next search. Repeat.

Learning what a good construction looks like

Many combinatorial problems are: *find the best object in a huge discrete space*. Local search finds good ones, then gets stuck.

PatternBoost (Charton, Ellenberg, Wagner, Williamson, 2024)

Alternate two phases: a classical local search produces many good constructions, then *a transformer is trained on the best of them* and its samples seed the next search. Repeat.

Graham and Harary: how many edges can you delete from the d-cube *without increasing its diameter*? The known construction leaves $2^d + \binom{d}{\lfloor d/2 \rfloor} - 2$ edges, and Graham conjectured it optimal. For $d = 6$ that is 82.

Learning what a good construction looks like

Many combinatorial problems are: *find the best object in a huge discrete space*. Local search finds good ones, then gets stuck.

PatternBoost (Charton, Ellenberg, Wagner, Williamson, 2024)

Alternate two phases: a classical local search produces many good constructions, then *a transformer is trained on the best of them* and its samples seed the next search. Repeat.

Graham and Harary: how many edges can you delete from the d-cube *without increasing its diameter*? The known construction leaves $2^d + \binom{d}{\lfloor d/2 \rfloor} - 2$ edges, and Graham conjectured it optimal. For $d = 6$ that is 82.

PatternBoost found 81. One edge — and the conjecture is false. First progress on it in thirty years.

Evolution over programs

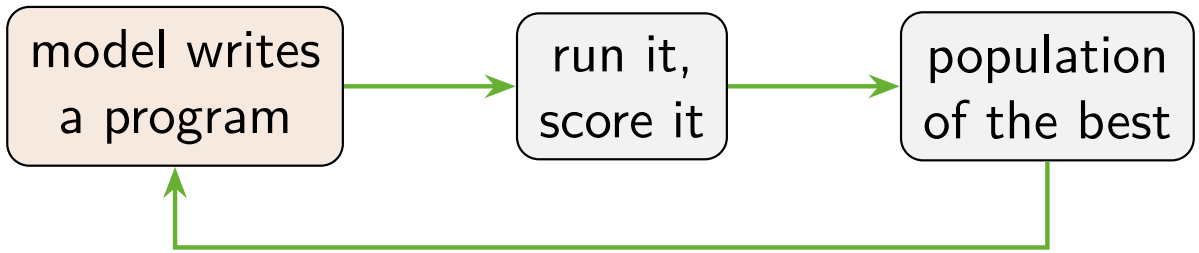
The idea (FunSearch, 2023; AlphaEvolve, 2025)

You do not ask the model for the answer. You ask it for a *program* that constructs a candidate, you *score* the candidate, and you keep the best ones as inspiration for the next generation.

Evolution over programs

The idea (FunSearch, 2023; AlphaEvolve, 2025)

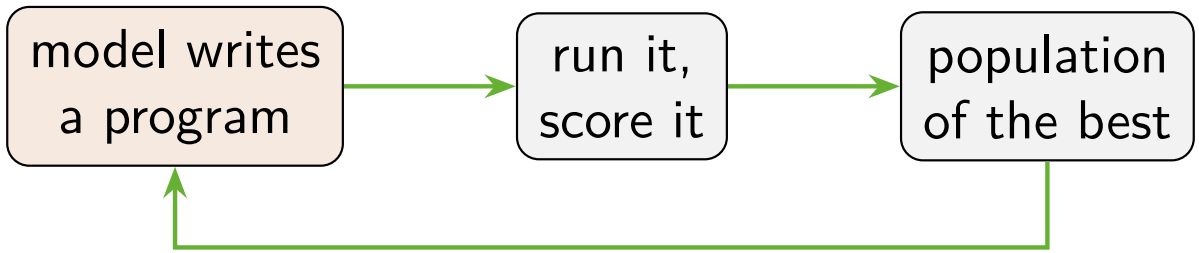
You do not ask the model for the answer. You ask it for a *program* that constructs a candidate, you *score* the candidate, and you keep the best ones as inspiration for the next generation.



Evolution over programs

The idea (FunSearch, 2023; AlphaEvolve, 2025)

You do not ask the model for the answer. You ask it for a *program* that constructs a candidate, you *score* the candidate, and you keep the best ones as inspiration for the next generation.



Why this is safe

The output is a *verifiable object* — a construction, an algorithm, a certificate. You check it by hand or by machine. *The model's reasoning is never trusted; only its output is used.*

What it has actually found

- **FunSearch** (Nature, 2024): a new lower bound for the *cap set problem*, and better heuristics for online bin packing. Co-authored with Jordan Ellenberg.

What it has actually found

- **FunSearch** (Nature, 2024): a new lower bound for the *cap set problem*, and better heuristics for online bin packing. Co-authored with Jordan Ellenberg.
- **AlphaEvolve** (2025): multiplying two 4×4 complex matrices with *48* scalar multiplications — the first improvement on Strassen's 49 in 56 years.

  Also: a better lower bound for the kissing number in dimension 11.

What it has actually found

- **FunSearch** (Nature, 2024): a new lower bound for the *cap set problem*, and better heuristics for online bin packing. Co-authored with Jordan Ellenberg.
- **AlphaEvolve** (2025): multiplying two 4×4 complex matrices with *48* scalar multiplications — the first improvement on Strassen's 49 in 56 years.

  Also: a better lower bound for the kissing number in dimension 11.
- Turned loose on 67 *open problems* in analysis, combinatorics, geometry and number theory, it rediscovered the best known solution in most cases and *improved* on several.

What it has actually found

- **FunSearch** (Nature, 2024): a new lower bound for the *cap set problem*, and better heuristics for online bin packing. Co-authored with Jordan Ellenberg.
- **AlphaEvolve** (2025): multiplying two 4×4 complex matrices with *48* scalar multiplications — the first improvement on Strassen's 49 in 56 years.

  Also: a better lower bound for the kissing number in dimension 11.
- Turned loose on *67 open problems* in analysis, combinatorics, geometry and number theory, it rediscovered the best known solution in most cases and *improved* on several.

The paper to read

Georgiev, Gómez-Serrano, Tao and Wagner, *Mathematical exploration and discovery at scale*, arXiv:2511.02864 (2025).

Which of your problems this fits

It works when

the object is *finite and explicit* — a configuration, a construction, a set of coefficients; there is a *cheap score*, so you can rank two candidates in milliseconds; and the search space is too large to guess but structured enough that good candidates resemble each other.

Which of your problems this fits

It works when

the object is *finite and explicit* — a configuration, a construction, a set of coefficients; there is a *cheap score*, so you can rank two candidates in milliseconds; and the search space is too large to guess but structured enough that good candidates resemble each other.

It does not work when

the answer is a *proof* rather than an object, scoring needs human judgement, or the object is infinite-dimensional with no faithful discretisation.

Which of your problems this fits

It works when

the object is *finite and explicit* — a configuration, a construction, a set of coefficients; there is a *cheap score*, so you can rank two candidates in milliseconds; and the search space is too large to guess but structured enough that good candidates resemble each other.

It does not work when

the answer is a *proof* rather than an object, scoring needs human judgement, or the object is infinite-dimensional with no faithful discretisation.

Try this at home

OpenEvolve and *ShinkaEvolve* are free and open source. A first run is an afternoon's work.

The common structure

In every one of these, the machine produces *a candidate that a human then understands, and that a proof then certifies*. This is the same template as the interval arithmetic story from yesterday.

The common structure

In every one of these, the machine produces *a candidate that a human then understands, and that a proof then certifies*. This is the same template as the interval arithmetic story from yesterday.

Three steps of one idea: the network *points at the variables* (2021), *proposes the objects* (2024), then *writes the program* that builds them (2025).

The common structure

In every one of these, the machine produces *a candidate that a human then understands, and that a proof then certifies*. This is the same template as the interval arithmetic story from yesterday.

Three steps of one idea: the network *points at the variables* (2021), *proposes the objects* (2024), then *writes the program* that builds them (2025).

What it learned was never mathematics — which variables matter, what a good example looks like, which programs score well. *The mathematics stayed with the mathematician.*

What to do this week

1. **Pick a real question** from your current work — not a test question. Something you actually want to know.

What to do this week

1. **Pick a real question** from your current work — not a test question. Something you actually want to know.
2. **Give it the full context.** The definitions, the paper, what you tried.

What to do this week

1. **Pick a real question** from your current work — not a test question. Something you actually want to know.
2. **Give it the full context.** The definitions, the paper, what you tried.
3. **Install an agent** and let it write one experiment for you. *This is the step people skip, and it is the one that matters.*

What to do this week

1. **Pick a real question** from your current work — not a test question. Something you actually want to know.
2. **Give it the full context.** The definitions, the paper, what you tried.
3. **Install an agent** and let it write one experiment for you. *This is the step people skip, and it is the one that matters.*
4. **Check everything it tells you**, and notice how often it is right, and how it is wrong when it is wrong.

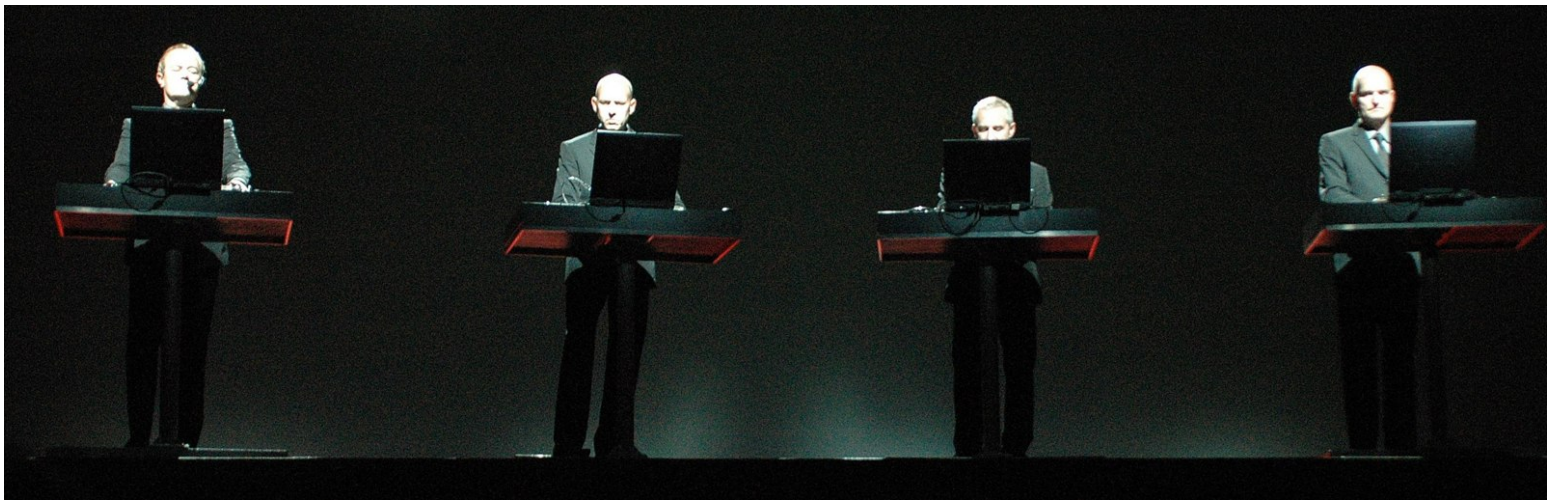
What to do this week

1. **Pick a real question** from your current work — not a test question. Something you actually want to know.
2. **Give it the full context.** The definitions, the paper, what you tried.
3. **Install an agent** and let it write one experiment for you. *This is the step people skip, and it is the one that matters.*
4. **Check everything it tells you**, and notice how often it is right, and how it is wrong when it is wrong.

Tomorrow

The uses that have nothing to do with code — and then the difficult conversation: what this does to our community, and what I am genuinely worried about.

¡Hasta mañana!



Kraftwerk, Ferrara, 2005.
Photo Daniele Dalledonne, CC BY-SA 2.0.

References

Search, evolution, discovery


Davies A. et al.
Advancing mathematics by guiding human intuition with AI. Nature **600** (2021), 70–74.


Romera-Paredes B. et al.
Mathematical discoveries from program search with large language models. Nature **625** (2024), 468–475.


Charton F., Ellenberg J. S., Wagner A. Z., Williamson G.
PatternBoost: constructions in mathematics with a little help from AI. arXiv:2411.00566 (2024).

References

Search, evolution, discovery (continued)


Carballeira A., Galant D., Gómez-Serrano J.
The energy of fractional Allen–Cahn layers in dimension one.
arXiv:2608.06226 (2026) — the project behind the dashboard.


Novikov A. et al.
AlphaEvolve: a coding agent for scientific and algorithmic discovery.
Google DeepMind (2025).


Georgiev B., Gómez-Serrano J., Tao T., Wagner A. Z.
Mathematical exploration and discovery at scale. arXiv:2511.02864 (2025).


Lange R. T., Imajuku Y., Cetin E.
ShinkaEvolve: towards open-ended and sample-efficient program evolution. arXiv:2509.19349 (2025).

References

Dilation of regular polygons


Mulzer W.
Minimum dilation triangulations for the regular n-gon. Master's thesis, Freie Universität Berlin (2004).


Dumitrescu A., Ghosh A.
Lower bounds on the dilation of plane spanners. In: Algorithms and Discrete Applied Mathematics, Springer (2016), 139–151.


Sattari S., Izadi M.
An improved upper bound on dilation of regular polygons. Computational Geometry **80** (2019), 53–68. doi:10.1016/j.comgeo.2019.01.009

References

Post-training, RL, and chains of thought


Christiano P., Leike J., Brown T. B., Martic M., Legg S., Amodei D. *Deep reinforcement learning from human preferences.* NeurIPS 2017. arXiv:1706.03741.


Ouyang L. et al. *Training language models to follow instructions with human feedback.* NeurIPS 2022. arXiv:2203.02155.


Wei J. et al. *Chain-of-thought prompting elicits reasoning in large language models.* NeurIPS 2022. arXiv:2201.11903.


Kojima T., Gu S. S., Reid M., Matsuo Y., Iwasawa Y. *Large language models are zero-shot reasoners.* NeurIPS 2022. arXiv:2205.11916.

References

Post-training, RL, and chains of thought (continued)


DeepSeek-AI
DeepSeek-R1: incentivizing reasoning capability in LLMs via reinforcement learning. Nature **645** (2025), 633–638. arXiv:2501.12948.


Turpin M., Michael J., Perez E., Bowman S. R.
Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. NeurIPS 2023. arXiv:2305.04388.

References

Two conjectures that fell in 2026


Alon N., Bloom T. F., Gowers W. T., Litt D., Sawin W., Shankar A., Tsimerman J., Wang V., Matchett Wood M.
Remarks on the disproof of the unit distance conjecture.
arXiv:2605.20695 (May 2026).


Gao S.
Counterexamples to the Jacobian conjecture in dimensions greater than two. arXiv:2608.00222 (2026) — recounts Alpöge's counterexample and its sequels.

Image credits

- **Kraftwerk, Ferrara, 2005.** Photo Daniele Dalledonne, CC BY-SA 2.0, via Wikimedia Commons. Resized; otherwise unaltered. https://commons.wikimedia.org/wiki/File:Kraftwerk_live_in_Ferrara_2005_(1).JPG