Machine Learning and Neural AI

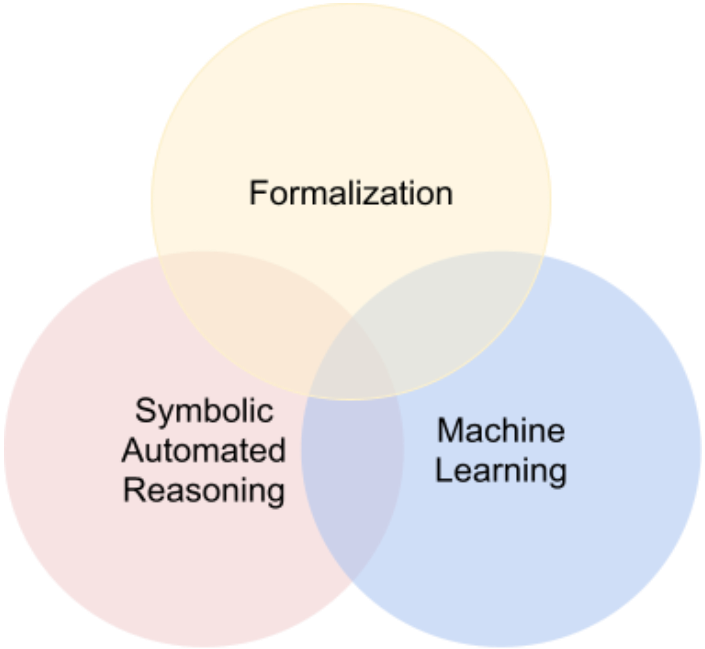
Jeremy Avigad
Department of Philosophy
Department of Mathematical Sciences
Carnegie Mellon University

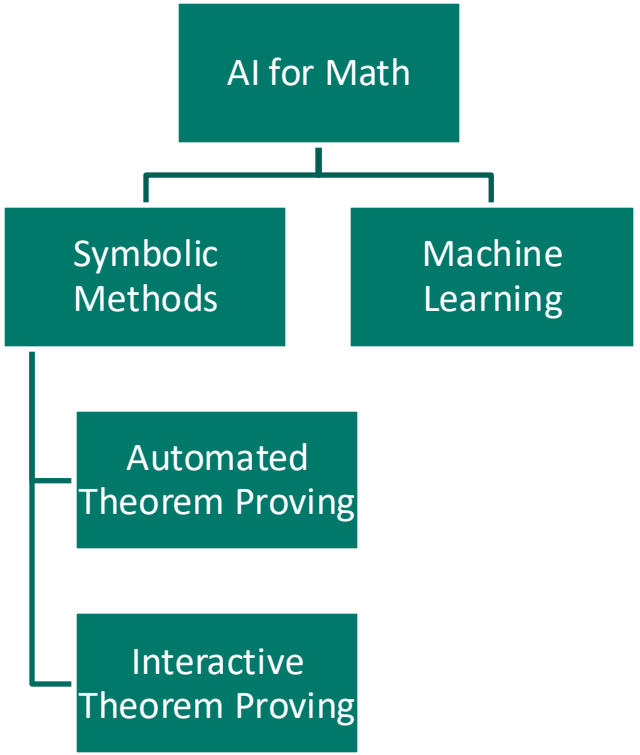
Institute for Computer-Aided Reasoning in Mathematics

October 16, 2025

Overview

AI for mathematics





Applications to mathematics

Some applications I know of:

- finding patterns in computational data
- finding combinatorial objects
- learning about PDEs
- finding expressions
- finding formal proofs
- finding informal proofs

These rely on mixtures of supervised, unsupervised, and reinforcement learning, and more.

Machine learning

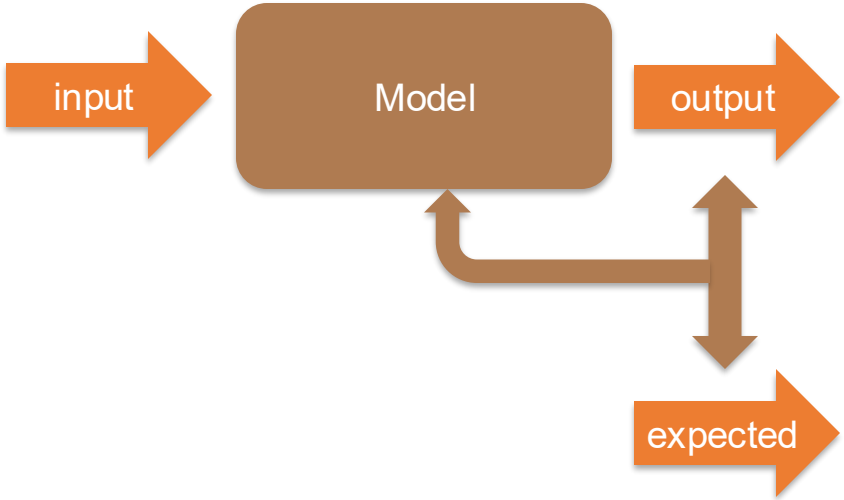
Key approaches:

- **Supervised learning:** the system is presented with (input, output) pairs and learns a rule connecting them.
- **Unsupervised learning:** the system is presented with data and learns some sort of structure.
- **Reinforcement learning:** the system acts in a space and is rewarded accordingly; it learns to maximize rewards.

Models can be very simple (linear regression, decision trees) to very complex (neural networks).

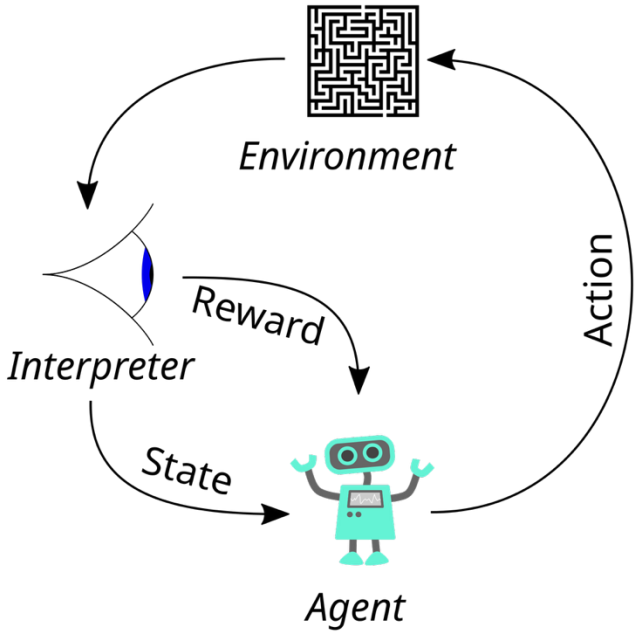
Supervised learning with a neural network

- A *model* has and *architecture* and lots of *parameters.*
- It computes a nonlinear function of the *inputs*.
- The *outputs* are compared to expected values.
- Discrepancies are evaluated by a *loss function*.
- The system learns to minimize the loss.



Reinforcement learning

- The interaction is between an *agent* and an *environment*.
- The agent gets information from the environment.
- The agent chooses an *action* from an *action space*.
- The environment changes; ultimately the agent gets a *reward*.
- The agent learns to maximize the reward
  - possibly by learning a value function.
  - possibly by learning a policy.



Large language models

Train a model on a lot of data.

Given a sequence of tokens, what is likely to come next?

It was a dark and stormy _____

Generative AI samples from the distribution.

Can models learn to generate proofs this way?

Where does the data come from?

Supervised learning:

- the internet
- computation
- databases

Reinforcement learning:

- games
- the real world
- computation

History

A brief history

- In his 1950 paper, “Computing machinery and intelligence,” Alan Turing discusses both symbolic and machine-learning approaches.
- Machine learning approaches were presented at the Dartmouth Summer Research Project on Artificial Intelligence in 1956.
- Research was ongoing in the latter half of the twentieth century.
- 1990s: rise of big data
- 2012: AlexNet, a neural network, wins the ImageNet Large Scale Visual Recognition Challenge
- 2016: AlphaGo beats Lee Sedol in a five-game match
- 2022: ChatGPT released

Discovering Patterns in Computational Data

Discovering patterns in knot invariants

Knot theorists assign algebraic, geometric, and numeric *invariants*.

Working with Google DeepMind, András Juhász and Marc Lackenby studied:

- hyperbolic invariants
- algebraic invariants

A supervised learning model was able to learn to predict an algebraic invariant, $\sigma(k)$, from a collection of hyperbolic invariants.

Sensitivity analysis reduced the dependency to three quantities, $Re(\mu)$, $Im(\mu)$, and λ.

Discovering patterns in knot invariants

Based on the data, and further experimentation, they conjectured and then proved a theorem:

Theorem. There exists a constant c such that, for any hyperbolic knot K, $|2\sigma(K) - slope(K)| \leq c\ vol(K)\ inj(K)^{-3}$.

Article

Advancing mathematics by guiding human intuition with AI

https://doi.org/10.1038/s41586-021-04086-x

Received: 10 July 2021

Accepted: 30 September 2021

Published online: 1 December 2021

Open access

Check for updates

Alex Davies[1]✉, Petar Veličković[1], Lars Buesing[1], Sam Blackwell[1], Daniel Zheng[1], Nenad Tomašev[1], Richard Tanburn[1], Peter Battaglia[1], Charles Blundell[1], András Juhász[2], Marc Lackenby[2], Geordie Williamson[3], Demis Hassabis[1] & Pushmeet Kohli[1]✉

The practice of mathematics involves discovering patterns and using these to formulate and prove conjectures, resulting in theorems. Since the 1960s, mathematicians have used computers to assist in the discovery of patterns and formulation of conjectures[1], most famously in the Birch and Swinnerton-Dyer conjecture[2], a Millennium Prize Problem[3]. Here we provide examples of new fundamental results in pure mathematics that have been discovered with the assistance of machine learning—demonstrating a method by which machine learning can aid mathematicians in discovering new conjectures and theorems. We propose a process of using machine learning to discover potential patterns and relations between mathematical objects, understanding them with attribution techniques and using these observations to guide intuition and propose conjectures. We outline this machine-learning-guided framework and demonstrate its successful application to current research questions in distinct areas of pure mathematics, in each case showing how it led to meaningful mathematical contributions on important open problems: a new connection between the algebraic and geometric structure of knots, and a candidate algorithm predicted by the combinatorial invariance conjecture for symmetric groups[4]. Our work may serve as a model for collaboration between the fields of mathematics and artificial intelligence (AI) that can achieve surprising results by leveraging the respective strengths of mathematicians and machine learning.

Discovering patterns in number theory

In 2020, Yang-Hui He, Kyu-Hwan Lee, and Thomas Oliver used machine learning algorithms on data associated with elliptic curves over the finite fields with p elements, with p prime.

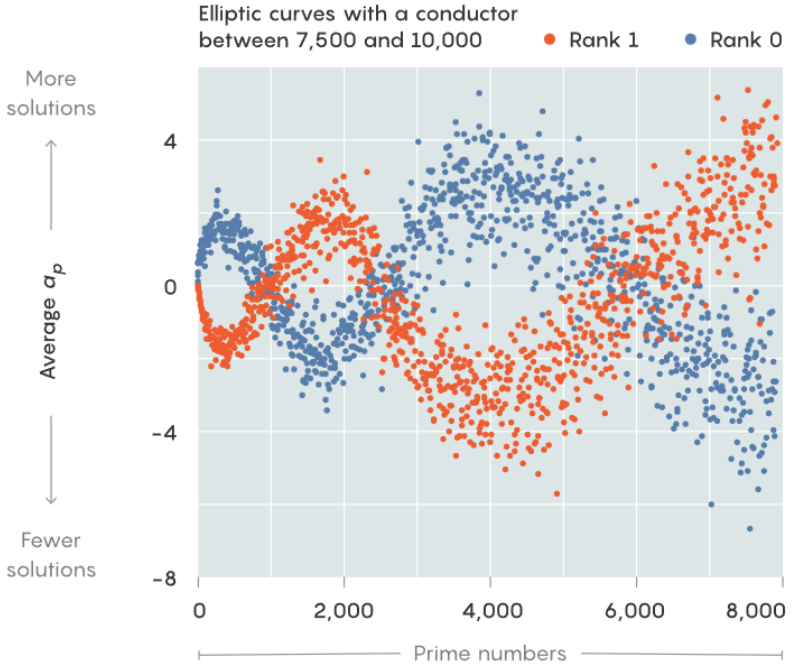
They determined that a simple logistical regression model could predict the rank of r_E from the Frobenius traces $a_p(E) = p + 1 - N_p$, where N_p is the number of rational points on the curve.

The prediction matches the Birch-Swinnerton-Dyer conjecture.

Discovering patterns in number theory

In 2022, Lee asked an undergraduate student, Alexey Pozdnyakov, to study the data.

Pozdnyakov used elementary data analysis methods and discovered a pattern separating rank 0 and rank 1 curves as a function of p.



Discovering patterns in number theory

The pattern was dubbed a “murmuration” phenomenon.

Drew Sutherland at MIT used the LMFDB to confirm the pattern with a lot more data.

In 2023, Nina Zubrilina, a student of Peter Sarnak, derived a formula that explains the phenomenon.

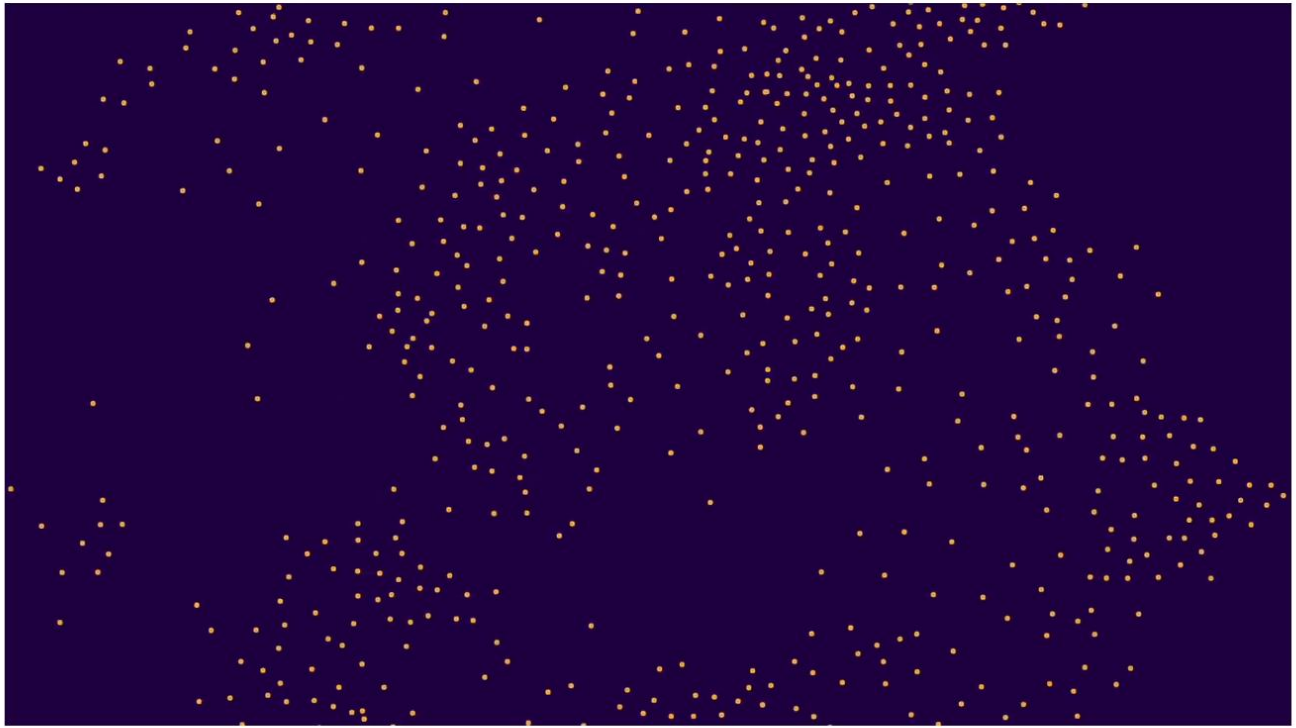
Pozdnyakov has provided Python notebooks online that reproduce the analysis. You can easily run them in Colab.

NUMBER THEORY

Elliptic Curve 'Murmurations' Found With AI Take Flight

6 |

Mathematicians are working to fully explain unusual behaviors uncovered using artificial intelligence.



When viewed the right way, elliptic curves can flock like birds.

Paul Chaikin for *Quanta Magazine*

scikit-learn

Machine Learning in Python

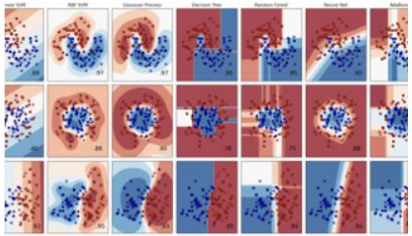
Getting Started | Release Highlights for 1.7

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

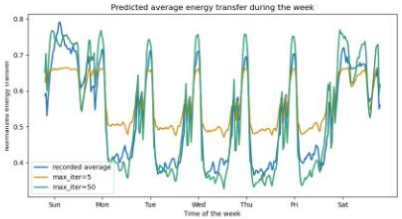


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

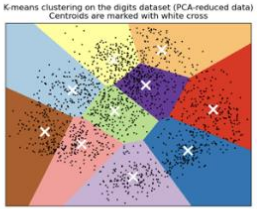


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, grouping experiment outcomes.
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...



Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, increased efficiency.
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning.
Algorithms: Grid search, cross validation, metrics, and more...

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: Preprocessing, feature extraction, and more...

Computing complex data

Physics-informed neural networks

Neural networks learn from data.

Physical data is often constrained to satisfy partial differential equations, solutions to which are hard to compute.

PINNs build-in PDE constraints. They can then learn from and predict data subject to physical constraints.

This is an important new field in scientific computation.

Studying PDEs

PINNs can be used to compute approximate numeric solutions to PDEs.

They have been used to find detect singularities that are then confirmed by conventional methods.

Google DeepMind is going after a Clay Millenium Prize and hoping to find unstable singularities in the Navier-Stokes equations.

MATHEMATICAL PHYSICS

Deep Learning Poised to 'Blow Up' Famed Fluid Equations

6 |

For centuries, mathematicians have tried to prove that Euler's fluid equations can produce nonsensical answers. A new approach to machine learning has researchers betting that "blowup" is near.

Mathematics > Analysis of PDEs

[Submitted on 17 Sep 2025]

Discovery of Unstable Singularities

Yongji Wang, Mehdi Bennani, James Martens, Sébastien Racanière, Sam Blackwell, Alex Matthews, Stanislav Nikolov, Gonzalo Cao-Labora, Daniel S. Park, Martin Arjovsky, Daniel Worrall, Chongli Qin, Ferran Alet, Borislav Kozlovskii, Nenad Tomašev, Alex Davies, Pushmeet Kohli, Tristan Buckmaster, Bogdan Georgiev, Javier Gómez-Serrano, Ray Jiang, Ching-Yao Lai

Whether singularities can form in fluids remains a foundational unanswered question in mathematics. This phenomenon occurs when solutions to governing equations, such as the 3D Euler equations, develop infinite gradients from smooth initial conditions. Historically, numerical approaches have primarily identified stable singularities. However, these are not expected to exist for key open problems, such as the boundary-free Euler and Navier-Stokes cases, where unstable singularities are hypothesized to play a crucial role. Here, we present the first systematic discovery of new families of unstable singularities. A stable singularity is a robust outcome, forming even if the initial state is slightly perturbed. In contrast, unstable singularities are exceptionally elusive; they require initial conditions tuned with infinite precision, being in a state of instability whereby infinitesimal perturbations immediately divert the solution from its blow-up trajectory. In particular, we present multiple new, unstable self-similar solutions for the incompressible porous media equation and the 3D Euler equation with boundary, revealing a simple empirical asymptotic formula relating the blow-up rate to the order of instability. Our approach combines curated machine learning architectures and training schemes with a high-precision Gauss-Newton optimizer, achieving accuracies that significantly surpass previous work across all discovered solutions. For specific solutions, we reach near double-float machine precision, attaining a level of accuracy constrained only by the round-off errors of the GPU hardware. This level of precision meets the requirements for rigorous mathematical validation via computer-assisted proofs. This work provides a new playbook for exploring the complex landscape of nonlinear partial differential equations (PDEs) and tackling long-standing challenges in mathematical physics.

Finding mathematical objects

Finding mathematical objects

One can use supervised learning with synthetic data to find expressions with verifiable properties.

- antiderivatives
- Lyapunov functions

The challenge is to generate data with a representative distribution.

Computer Science > Symbolic Computation

[Submitted on 2 Dec 2019]

Deep Learning for Symbolic Mathematics

Guillaume Lample, François Charton

Neural networks have a reputation for being better at solving statistical or approximate problems than at performing calculations or working with symbolic data. In this paper, we show that they can be surprisingly good at more elaborated tasks in mathematics, such as symbolic integration and solving differential equations. We propose a syntax for representing mathematical problems, and methods for generating large datasets that can be used to train sequence-to-sequence models. We achieve results that outperform commercial Computer Algebra Systems such as Matlab or Mathematica.

Computer Science > Machine Learning

[Submitted on 10 Oct 2024]

Global Lyapunov functions: a long-standing open problem in mathematics, with symbolic transformers

Alberto Alfarano, François Charton, Amaury Hayat

Despite their spectacular progress, language models still struggle on complex reasoning tasks, such as advanced mathematics. We consider a long-standing open problem in mathematics: discovering a Lyapunov function that ensures the global stability of a dynamical system. This problem has no known general solution, and algorithmic solvers only exist for some small polynomial systems. We propose a new method for generating synthetic training samples from random solutions, and show that sequence-to-sequence transformers trained on such datasets perform better than algorithmic solvers and humans on polynomial systems, and can discover new Lyapunov functions for non-polynomial systems.

Finding mathematical objects

Methods of finding combinatorial objects:

- use reinforcement learning
- use transformers + local search
- evolve code
  - AlphaEvolve
  - OpenEvolve
  - Shinka-Evolve

Constructions in combinatorics via neural networks

Adam Zsolt Wagner*

Abstract

We demonstrate how by using a reinforcement learning algorithm, the deep cross-entropy method, one can find explicit constructions and counterexamples to several open conjectures in extremal combinatorics and graph theory. Amongst the conjectures we refute are a question of Brualdi and Cao about maximizing permanents of pattern avoiding matrices, and several problems related to the adjacency and distance eigenvalues of graphs.

1 Introduction

Computer-assisted proofs have a long history in mathematics, including breakthrough results such as the proof of the four color theorem in 1976 by Appel and Haken [7], and the proof of the Kepler conjecture in 1998 by Hales [29]. Recently, significant progress has been made in the area of machine learning algorithms, and they have have quickly become some of the most exciting tools in a scientist's toolbox. In particular, recent advances in the field of reinforcement learning have led computers to

PatternBoost: Constructions in Mathematics with a Little Help from AI

François Charton* Jordan Ellenberg† Adam Zsolt Wagner‡
Geordie Williamson§

November 4, 2024

Abstract

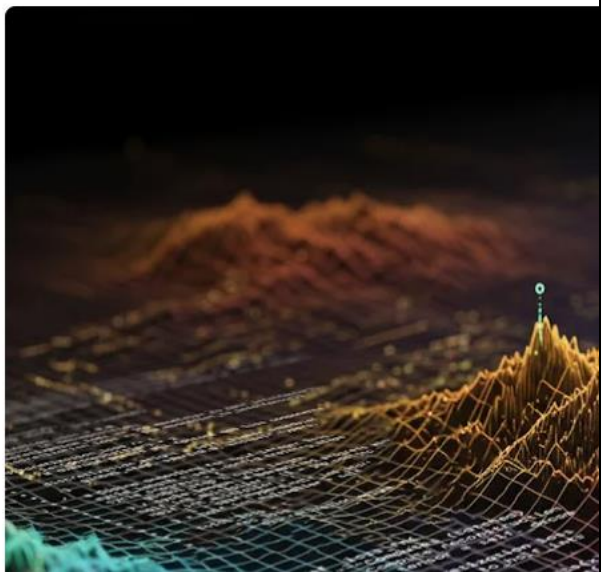
We introduce PatternBoost, a flexible method for finding interesting constructions in mathematics. Our algorithm alternates between two phases. In the first "local" phase, a classical search algorithm is used to produce many desirable constructions. In the second "global" phase, a transformer neural network is trained on the best such constructions. Samples from the trained transformer are then used as seeds for the first phase, and the process is repeated. We give a detailed introduction to this technique, and discuss the results of its application to several problems in extremal combinatorics. The performance of PatternBoost varies across different problems, but there are many situations where its performance is quite impressive. Using our technique, we find the best known solutions to several long-standing problems, including the construction of a counterexample to a conjecture that had remained open for 30 years.

SCIENCE

AlphaEvolve: A Gemini-powered coding agent for designing advanced algorithms

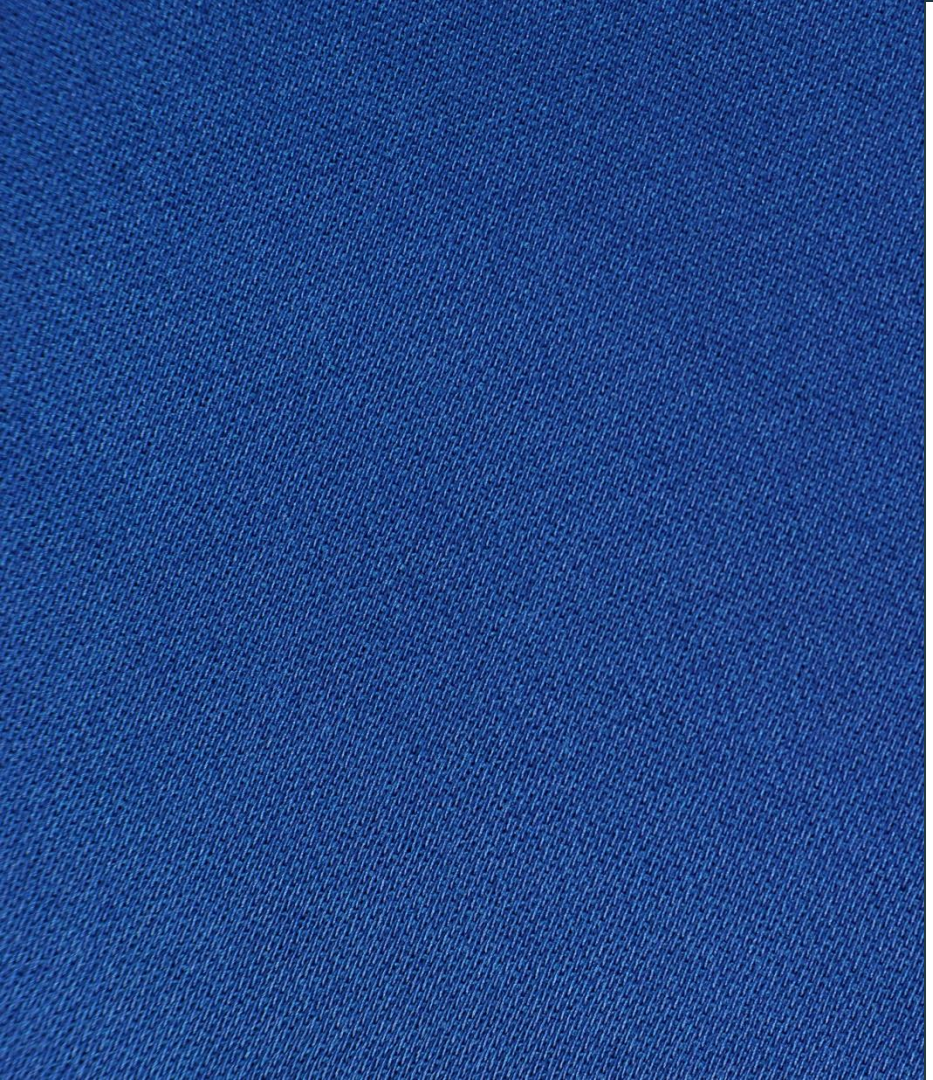
14 MAY 2025

By AlphaEvolve team



To investigate AlphaEvolve's breadth, we applied the system to over 50 open problems in mathematical analysis, geometry, combinatorics and number theory. The system's flexibility enabled us to set up most experiments in a matter of hours. In roughly 75% of cases, it rediscovered state-of-the-art solutions, to the best of our knowledge.

And in 20% of cases, AlphaEvolve improved the previously best known solutions, making progress on the corresponding open problems. For example, it advanced the kissing number problem. This geometric challenge has fascinated mathematicians for over 300 years and concerns the maximum number of non-overlapping spheres that touch a common unit sphere. AlphaEvolve discovered a configuration of 593 outer spheres and established a new lower bound in 11 dimensions.



Interlude

Skepticism

It’s not surprising that machine learning can act on symbolic and numeric data and find and construct finite objects and expressions.

Can it do *real* mathematics?

Mathematics is encoded in expressions. We can compute with definitions, statements, and proofs.

At one extreme, machine learning can act on formal libraries and data.

Intermediate encodings and representations are likely to be useful.

Machine Learning and Formal Methods

Generalities

If you are trying to get a neural system to do math, there are some choices to make.

What should the system do?

- **Search:** find theorems and definitions
- **Autoformalize:** given an informal definition / proof, formalize it
- **Prove:** construct formal or informal proofs
- **Conjecture:** guess potentially useful statements and objects

These are all interrelated.

Generalities

Should you use formal or informal data?

- There is a lot more informal mathematics out there.
- Proof assistants provide a clear signal for correctness.

You can use both, e.g.:

- translate formal to informal, sketch a proof, use chain of thought, then formalize
- translate informal to formal, search with a proof assistant, translate back

Generalities

Distinguish between:

- **Training:** how to you train the system?
- **Task:** when it comes time to perform, what does the system do?

The is no fine line between them. Given an IMO problem, AlphaProver makes up similar, possibly simpler problems, tries to solve them, and learns from the results.

Generalities

How are you going to evaluate success?

- for training and tuning
- to justify publication

Some benchmarks:

- **MiniF2F:** formalized high school problems
- **ProofNet:** college textbook exercises
- **PutnamBench:** Putnam problems
- **FrontierMath:** challenging problems with numerical answers

Generalities

If a system is learning by exploring, you need a local measure of success:

- numeric score
- approval from a proof assistant
- evaluation approval from a model

Asking humans to judge and provide feedback is too expensive.

Generalities

Where will you get data?

- scrape it from the web
- scrape it from specific databases
- generate data yourself
  - use symbolic methods
  - use another model

Generating it manually is generally too expensive.

Generalities

Strategies for theorem proving:

- whole proof generation
- iteration on error messages
- line-by-line generation
- sketch / refine

Strategies for boosting performance:

- **Pass@k:** generate several answers, take the best one
- **Expert iteration:** use the best answers to retrain, iterate
- **Tree search:** generate multiple options at each step, take the best ones
- **Backtracking:** use successes and failures to revisit earlier choices
- **Chain of Thought:** prompt / train the model to generate informal reasoning steps
- **Brute force:** use more computation.

Making it science

Once you get anything to work, find optimal settings of parameters.

- Parameter sweep is often too expensive.
- Use local search as due diligence.

Perform *ablations*: turn off features and components one by one to determine their effects.

An ML paper needs:

- an idea (only one!)
- an experimental setup
- evaluation

Search and retrieval

Premise selection

Thomas Zhu, Joshua Clune, Albert Jiang, Sean Welleck, and I have developed a neural premise selector for Lean, using methods introduced in Mikuła et al., “Magnushammer: A Transformer-Based Approach to Premise Selection.”

Method:

- Collect positive and negative instances of (goal, premise) pairs.
- Use a transformer model and embed both into the same latent space.
- Nudge positive instances closer, negative instances further.

Include information like docstrings, namespaces.

Computer Science > Machine Learning

[Submitted on 9 Jun 2025]

Premise Selection for a Lean Hammer

Thomas Zhu, Joshua Clune, Jeremy Avigad, Albert Qiaochu Jiang, Sean Welleck

Neural methods are transforming automated reasoning for proof assistants, yet integrating these advances into practical verification workflows remains challenging. Hammers are tools that interface with external automatic theorem provers to automate tedious reasoning steps. They have dramatically improved productivity in proof assistants, but the Lean proof assistant still does not have a hammer despite its growing popularity. We present LeanHammer, the first end-to-end domain-general hammer for Lean, built on a novel neural premise selection system for a hammer in dependent type theory. Unlike existing Lean premise selectors, our approach dynamically adapts to user-specific contexts and combines with symbolic proof search and reconstruction to create a practical hammer. With comprehensive evaluations, we show that our premise selector enables LeanHammer to solve 21\% more goals relative to existing premise selectors, and generalize well to diverse domains. Our work bridges the gap between neural retrieval and symbolic reasoning, making formal verification more accessible to researchers and practitioners.

LeanSearch

LeanSearch similarly uses variations on existing retrieval methods to match informal queries with formal content.

LeanSearch

Find theorems in Mathlib4 using natural language query

Query Name or description of the theorem or definition you are looking for

the intermediate value theorem

Number of results 50

Clear | Query Augmentation | Search

Tip: **Query Augmentation** augments your query to increase the chance to find relevant results.

intermediate_value_Icc theorem

```
∀ {α : Type u} [inst : ConditionallyCompleteLinearOrder α] [inst_1 : TopologicalSpace α]
[inst_2 : OrderTopology α]  [inst_3 : DenselyOrdered α] {δ : Type u_1} [inst_4 : LinearOrder
δ] [inst_5 : TopologicalSpace δ]  [inst_6 : OrderClosedTopology δ] {a b : α},  a ≤ b → ∀ {f :
α → δ}, ContinuousOn f (Icc a b) → Icc (f a) (f b) ⊆ f '' Icc a b
```

▶ Intermediate Value Theorem for Continuous Functions on Closed Intervals

Go to doc

intermediate_value_Ioo theorem

```
∀ {α : Type u} [inst : ConditionallyCompleteLinearOrder α] [inst_1 : TopologicalSpace α]
[inst_2 : OrderTopology α]  [inst_3 : DenselyOrdered α] {δ : Type u_1} [inst_4 : LinearOrder
δ] [inst_5 : TopologicalSpace δ]  [inst_6 : OrderClosedTopology δ] {a b : α},  a ≤ b → ∀ {f :
α → δ}, ContinuousOn f (Icc a b) → Ioo (f a) (f b) ⊆ f '' Ioo a b
```

▶ Intermediate Value Theorem for Continuous Functions on Closed Intervals and Open Image Intervals

Go to doc

Computer Science > Information Retrieval

[Submitted on 20 Mar 2024 (v1), last revised 4 Feb 2025 (this version, v2)]

A Semantic Search Engine for Mathlib4

Guoxiong Gao, Haocheng Ju, Jiedong Jiang, Zihan Qin, Bin Dong

The interactive theorem prover Lean enables the verification of formal mathematical proofs and is backed by an expanding community. Central to this ecosystem is its mathematical library, mathlib4, which lays the groundwork for the formalization of an expanding range of mathematical theories. However, searching for theorems in mathlib4 can be challenging. To successfully search in mathlib4, users often need to be familiar with its naming conventions or documentation strings. Therefore, creating a semantic search engine that can be used easily by individuals with varying familiarity with mathlib4 is very important. In this paper, we present a semantic search engine (this https URL) for mathlib4 that accepts informal queries and finds the relevant theorems. We also establish a benchmark for assessing the performance of various search engines for mathlib4.

Proving

Neurosymbolic proving

New proving systems are announced regularly, with new benchmark results in the abstract and long lists of authors. (SOTA = “state of the art.”)

They come from industry and academia, and they implement various among the general strategies I described.

Some are closed source, some are open source, and some are open weight.

I don’t have time to survey the field, but a graph from the recent Seed-Prover paper describes the SOTA two months ago.

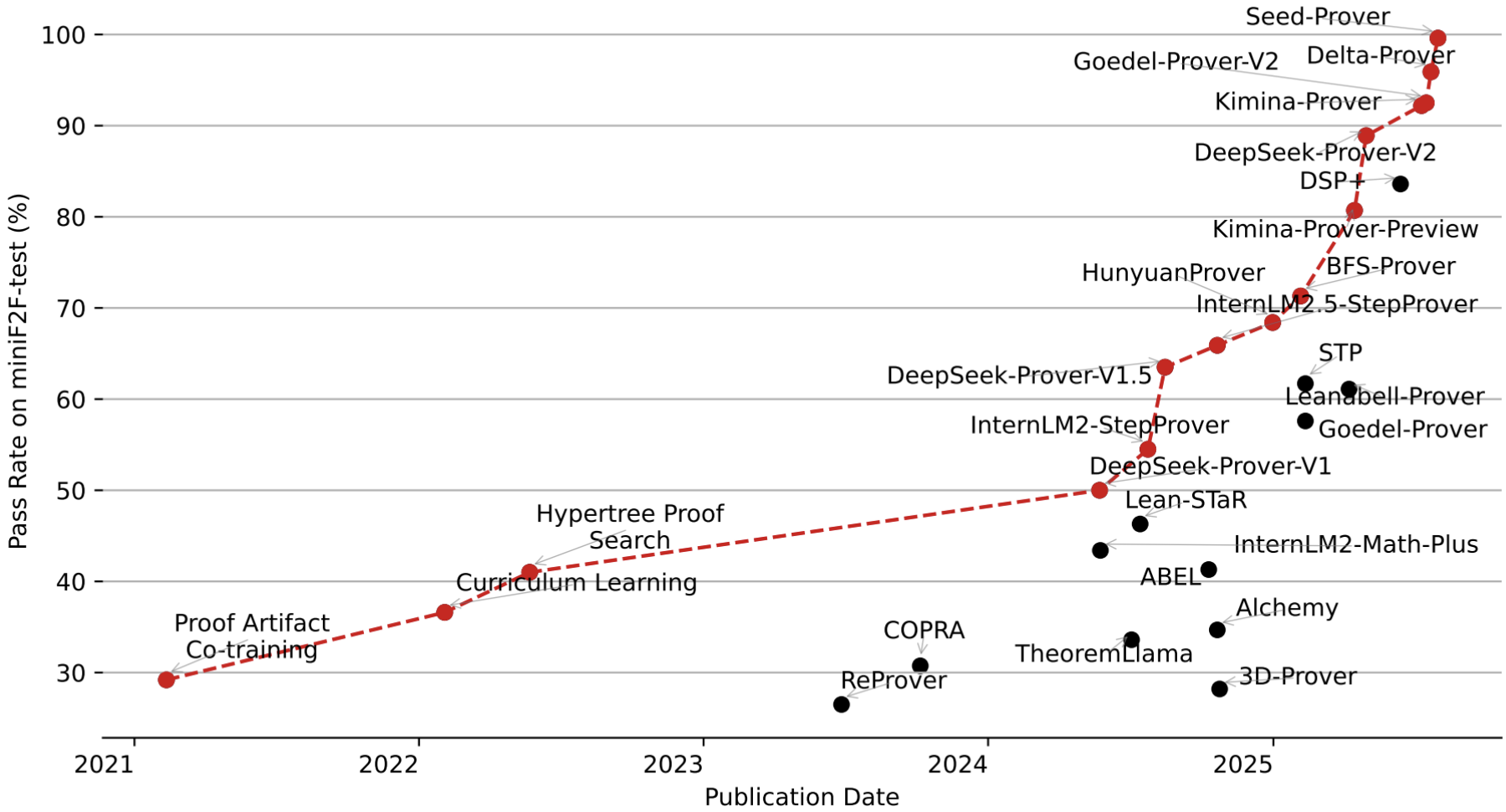
Computer Science > Artificial Intelligence

[Submitted on 31 Jul 2025 (v1), last revised 1 Aug 2025 (this version, v2)]

Seed-Prover: Deep and Broad Reasoning for Automated Theorem Proving

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu, Yuchen Wu, Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan, Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, Thomas Hanwen Zhu

LLMs have demonstrated strong mathematical reasoning abilities by leveraging reinforcement learning with long chain-of-thought, yet they continue to struggle with theorem proving due to the lack of clear supervision signals when solely using natural language. Dedicated domain-specific languages like Lean provide clear supervision via formal verification of proofs, enabling effective training through reinforcement learning. In this work, we propose \textbf{Seed-Prover}, a lemma-style whole-proof reasoning model. Seed-Prover can iteratively refine its proof based on Lean feedback, proved lemmas, and self-summarization. To solve IMO-level contest problems, we design three test-time inference strategies that enable both deep and broad reasoning. Seed-Prover proves 78.1% of formalized past IMO problems, saturates MiniF2F, and achieves over 50\% on PutnamBench, outperforming the previous state-of-the-art by a large margin. To address the lack of geometry support in Lean, we introduce a geometry reasoning engine \textbf{Seed-Geometry}, which outperforms previous formal geometry engines. We use these two systems to participate in IMO 2025 and fully prove 5 out of 6 problems. This work represents a significant advancement in automated mathematical reasoning, demonstrating the effectiveness of formal verification with long chain-of-thought reasoning.


Seed-Prover
Goedel-Prover-V2
Delta-Prover
Kimina-Prover
DeepSeek-Prover-V2
DSP+
Kimina-Prover-Preview
HunyuanProver
BFS-Prover
InternLM2.5-StepProver
STP
DeepSeek-Prover-V1.5
Leanabell-Prover
InternLM2-StepProver
Goedel-Prover
DeepSeek-Prover-V1
Lean-STaR
Hypertree Proof Search
InternLM2-Math-Plus
Curriculum Learning
ABEL
Alchemy
Proof Artifact Co-training
COPRA
TheoremLlama
3D-Prover
ReProver
100
90
80
70
60
50
40
30
2021
2022
2023
2024
2025
Pass Rate on miniF2F-test (%)
Publication Date

AI for mathematicians

AI for mathematicians

There is a tension between:

- doing ML research
- developing useful tools

See:

- Jason Rute, The Last Mile.
- A, AI for Mathematicians.

AI for mathematicians

Possible roles:

- **Expert colleague:** answers questions and makes suggestions
- **Copilot:** helps you write a proof
- **Grad student:** proves your lemmas for you
- **Collaborator:** proves some of the theorems in your next paper
- **Competitor:** proves theorems you wish you could prove

Variables:

- How long does it take to get an answer?
- How much compute is needed?
- How much does it cost?

AI for mathematicians

How long will it take before we can all use such tools?

Commercial foundation models already know about Lean. You can use Microsoft Copilot, Claude Sonnet, etc.

Several academic researchers are working on custom tools, provers, and copilots.

AI for mathematicians

Some corporate entities will soon have end-user tools:

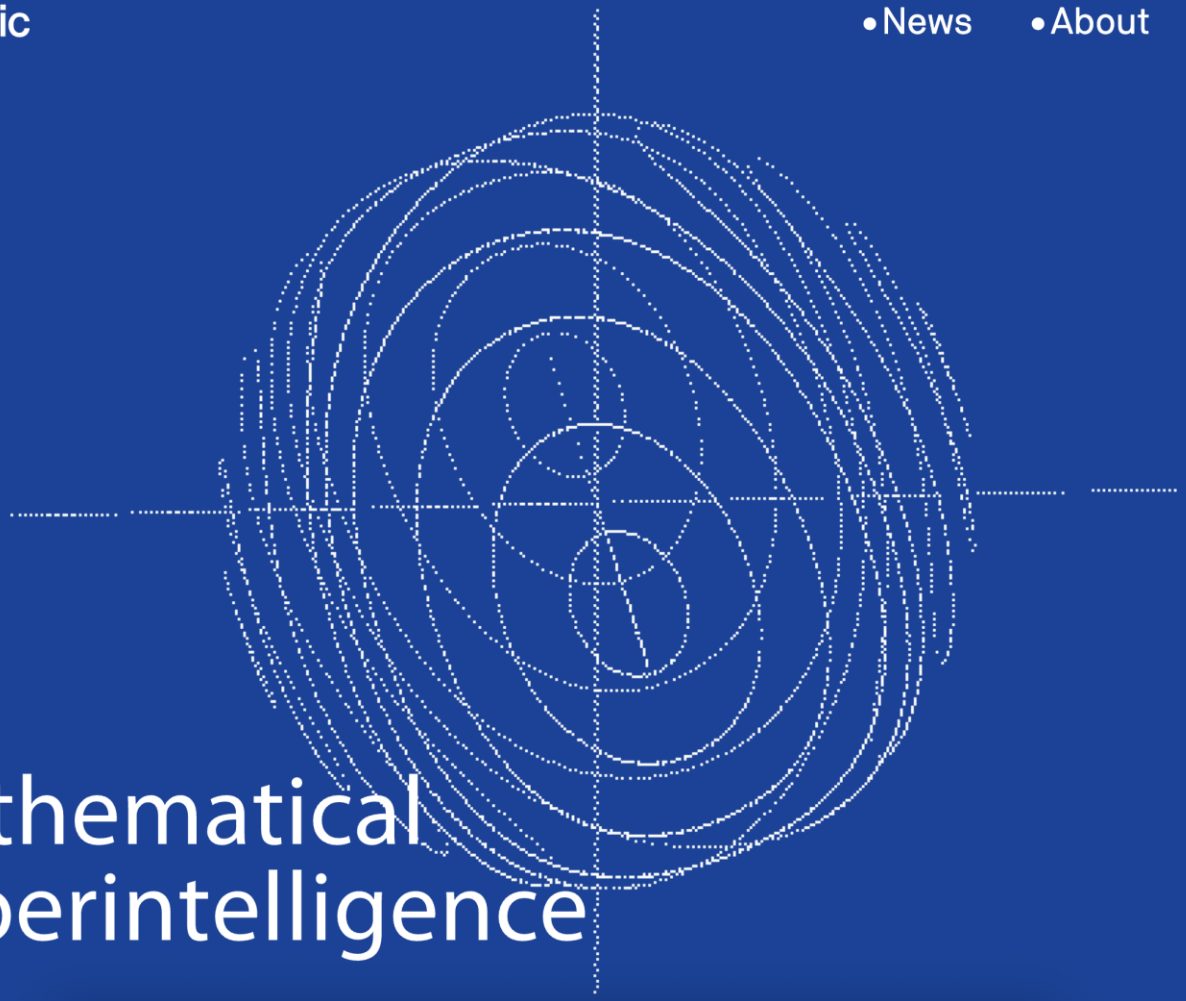
- **Harmonic:** has released an API for its Aristotle prover.
- **Math Inc.:** intends to make Gauss available.
- **Google DeepMind:** promised to make a version of AlphaProof available.

Things are moving quickly.

Harmonic's Aristotle achieves historic Gold-Medal Level Performance at the 2025 International Mathematical Olympiad - sign up for the Aristotle Lean API to try it yourself

Harmonic

• News • About • Careers



Mathematical Superintelligence

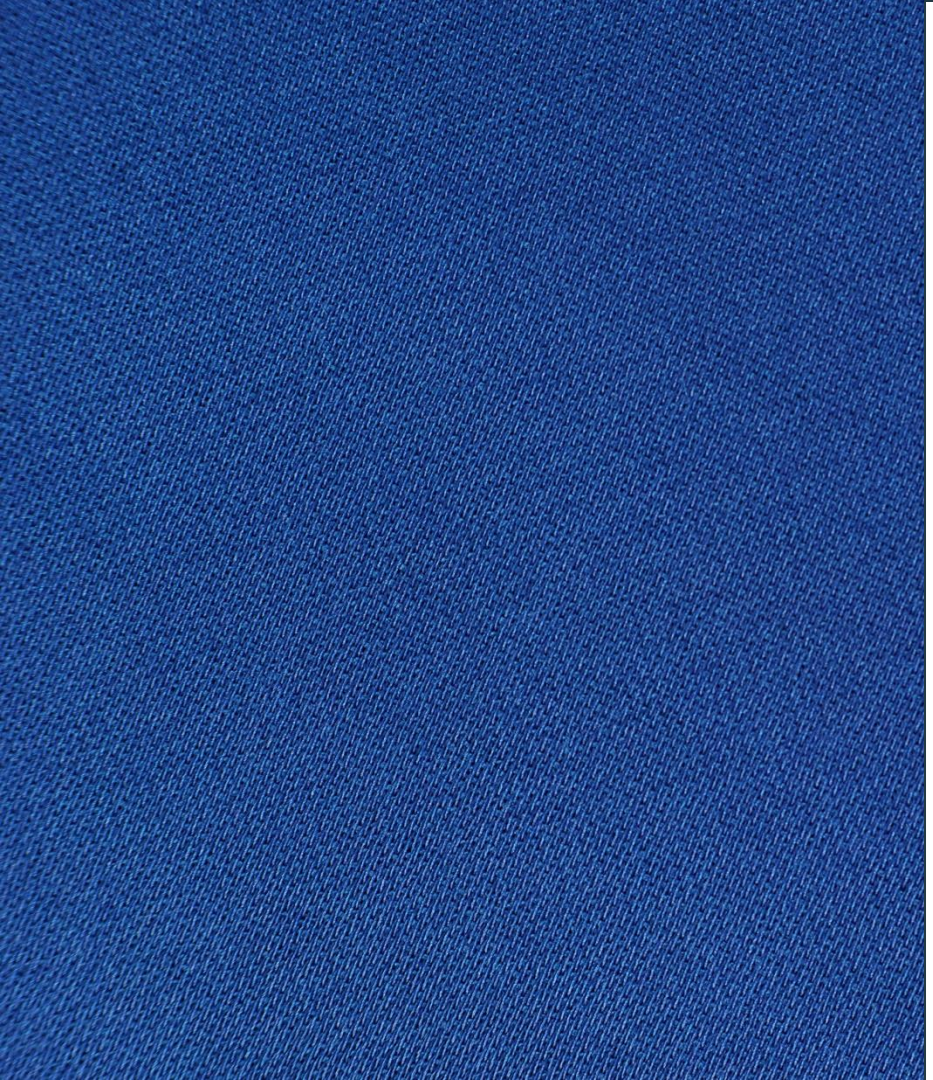
Math, Inc.

A new company dedicated to autoformalization and the creation of verified superintelligence.

Introducing Gauss, an agent for autoformalization →

Solve math, solve everything. →

Contact | Careers



Trying it Out

Trying it out

- You can run Andrew Pozdnyakov's remarkable tutorial in a Colab notebook.
- See Sean Welleck's transformer tutorials, based on similar tutorials by Adam Wagner.
- Open-source tools OpenEvolve and ShinkaEvolve are available online.
- Commercial foundation models and copilots know about Lean.
- You can find the projects, papers, and announcements discussed here online.

Conclusions

Conclusions

We have considered:

- formalization and proof assistants
- automated reasoning and symbolic AI
- machine learning and neural AI

The intersections, interactions, and synergies are especially interesting.

They are already beginning to play a role in mathematical discovery and verification.

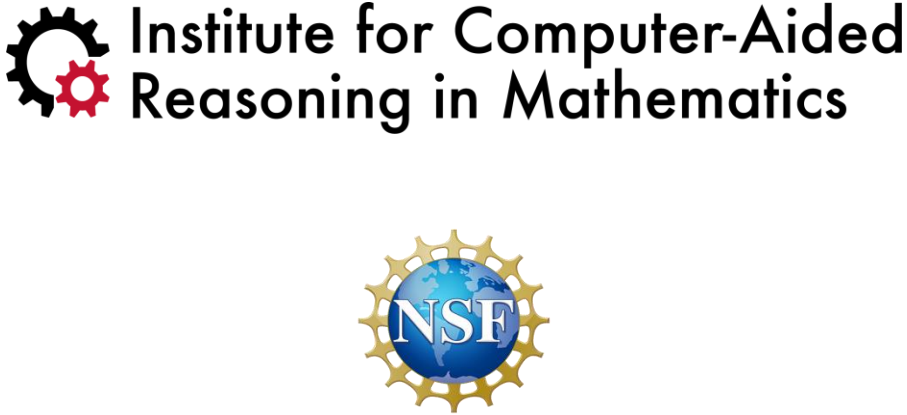
Conclusions

The technology is young, and we still have a lot to learn about how to use it well.

We need:

- exploration and experimentation
- collaborations between mathematicians and computer scientists.
- collaborations between generations.
- thoughtful discussion.

We hope to support all this with the new Institute for Computer-Aided Reasoning in Mathematics.



This is an exciting time for mathematics. Let’s make the most of it.