Some thoughts on AI and Mathematical Research

Sina Hazratpour, Milton Lin

April 2023

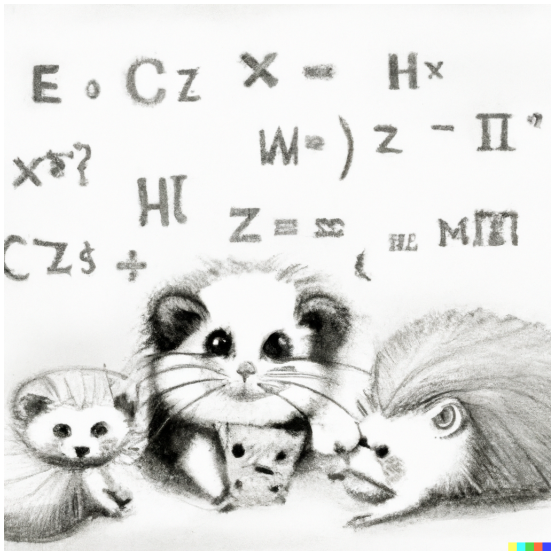
Unless stated otherwise, all art is AI-generated.

Contents

1 Introduction

The purpose of this article is to explore and report on the state-of-the-art in machine learning (ML) for *mathematical research*, serving as an extension to Williamson's survey [36]. Our main contribution is the impact of LLMs and recent advances in the theorem-proving community.

Mathematics, as a societal endeavor, encompasses teaching, learning, research, publishing, and outreach. There are complex dynamics between different stakeholders. Corporate interests often diverge from academic research. Academic research itself, is bounded by exhaustive publication protocols. The social nature of Mathematics often transcends beyond private thoughts; and its utility can raise crucial ethical questions, [5]. However, due to constraints, we curtail our discussion here. We refer to the text [12] for an introduction to these ideas.



The structure of the article follows that of a research mathematician, as described by Atyiah, [9]

> In mathematics, ideas and concepts come first, then come questions and problems. At this stage the search for solutions begins, one looks for a method or strategy...Before long you may realize, perhaps by finding counterexamples, that the problem was incorrectly formulated... Without proof the program remains incomplete, but without the imaginative input it never gets started.

This distribution of mental activity is partly reflected in the survey of an online collaborative project [16], see Fig. 1,

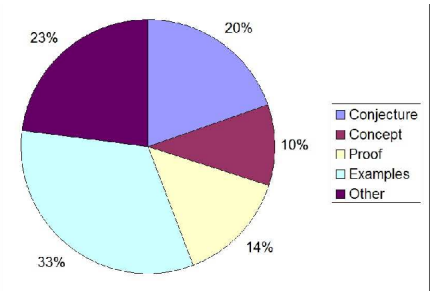


Figure 1: Distribution of comment contents

In section (2), we discuss progress towards the first stages of conjecturing. In section (3) we discuss progress towards AI-assisted proofs, or *formalization*. Lastly, in (4), we outline recent attempts in which the two are integrated. As observed by many, [37], AI-enabled mathematics not only helps mathematicians in their research but also serves as a litmus test for AGI itself: the ability to reason mathematically is a central unsolved problem in artificial intelligence.

1.1 Language models for mathematics

Recent advances and successes of large language models suggests exciting avenue. Modern LLMs are based on the decoder transformer architecture, [31]. With transfer learning, one can leverage the pre-trained model's knowledge to achieve better performance on the target task, even with limited labeled data.[1] This led to a range of pre-trained language models on mathematical data and fine-tuned on mathematical tasks. Language models satisfy the following two important properties:

- scaling laws. The ability of the model scales with both the number of parameters and the training corpus. The availability of good quality training corpus compared to natural language is scarce. In terms of training corpus, there are generally two types
  1. Curated datasets, such as Hendryck et al's [11], GitHuB, or arXiv datasets
  2. Synthetic datasets. Recent works, [37], suggests improvements from training on synthetic datasets.
- in-context learning (a.k.a prompting). This is a surprisingly simple way that steers LMs. This is the process of augmenting the context of instruction.

The extent to which mathematicians can leverage language models will be the focal point of this survey paper. [2] Tao gives a succinct summary of the current state of the art in using LLM-generated materials [28].

> Both humans and AI need to develop skills to analyze this new type of text. The stylistic signals that I traditionally rely on to "smell out" a hopelessly incorrect math argument are of little use with LLM-generated mathematics. Only line-by-line reading can discern if there is any substance. Strangely, even nonsensical LLM-generated math often references relevant concepts. With effort, human experts can modify ideas that do not work as presented into a correct and original argument.

[1] Note, however, that the inference of what a model makes is often dependent on its training corpus.

[2] We do not discuss on the fundamental problem of LLM's availability and their monetary restrictions. This, however, has immediate consequences for researchers. Smaller models empirically exhibit poorer estimation of tails of distributions, places higher probability mass on repetitions, etc, many of such empirically do not appear in larger models.

2 Learning from Lakotos: Conjectures and counterexamples

In the enlightening book, *Proofs and Refutations* of Lakotos [14] . Lakatos imagines an imaginary classroom, where the teacher makes a conjecture of the Euler-Poincaré formula for polyhedra: given a polyhedron where V, E and F are the number of vertices, edges and faces, respectively, then

$$V - E + F = 2 \tag{1}$$

The teacher provides an attempted proof where students respond by providing counterexamples. The dialogue continues back and forth, and the process can be summarized as

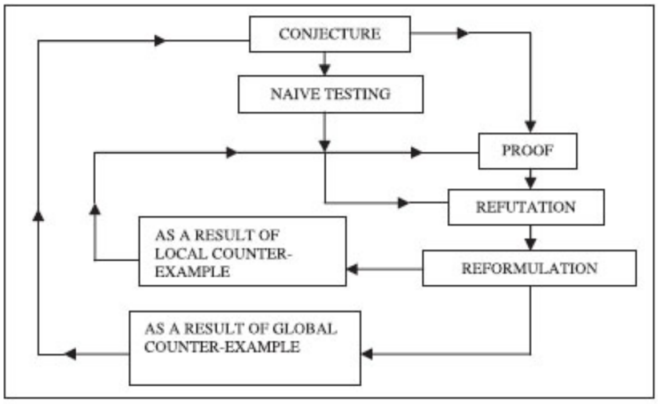


Figure 2: Lakatos method as explained in [7]

The two key elements are generating conjecture, 2.1 and verifying conjecture 2.3. This section explores how AI can assist in these two parts.[3]. Lastly, we note that there are fields of machine learning which has direct immediate impact to all natural sciences such as *semantic search* - once a conjecture is made, the first step would often be a search on whether similar results have been proven. For instance, a typical query would be of the form: "Is it true that X is satisfied for Y?". Such open-ended questions are hard. It is also unlikely that the same phrasing or even words were used in the hypothetical reference. [4] We limit our discussion to aspects of ML methods that are particular to the field of mathematics.

2.1 Generating conjectures and new concepts

One of the more prominent approaches to generate hypothesis, follows pipeline of Davies et al. [6]

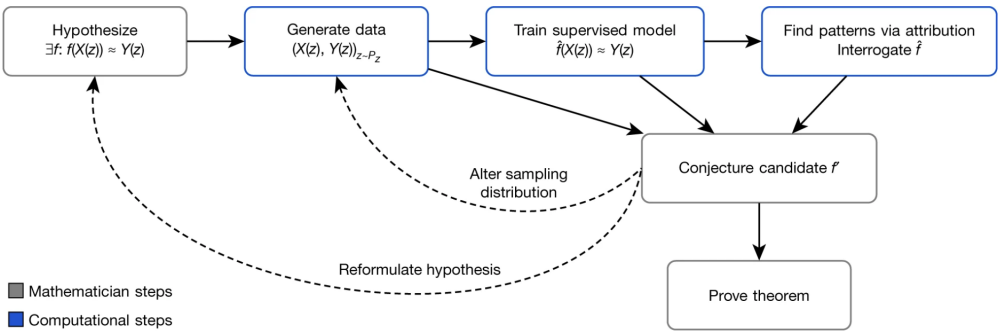


Figure 3: Pipeline for hypothesis generation in [6].

To illustrate this, consider the example of Euler-Poincaré formula, 1. One begins at the leftmost *blue* box. The *grey* boxes are required by human intervention. One conjectures a relation between features $X(z)$ and $Y(z)$ of a polyhedra z. In our case we let

$$X(z) = (V(z), E(z), \mathrm{Vol}(z), \mathrm{Sur}(z))$$

[3] In a somewhat similar spirit, Harris' blog post, [10], made an intriguing thought exercise of whether a machine can recreate the Poincaré conjecture in Thurston's paper [29]

[4] Progress towards autoformalization, 3.4, could help with this.

be the number of vertices, edges, volume and surface area of z, respectively. Let

$$Y(z) = (F(z))$$

be the number faces of z. Our goal is to find a function $\hat{f}(X(z)) = Y(z)$. Using classical supervised learning, we can obtain

$$\hat{f}((X(z)) = X(z) \cdot (1, -1, 0, 0) + 2$$

However, when data is high dimensional and the underlying relation is non-linear, $\hat{f}$ is only there for *intution*. The outcome of this training process is thus accompanied with *attribution techniques* , such as *gradient based techniques*, [26]. Another related work is the Ramanujan[5] machine, [18]. The machine attempts to find equality of the following form

$$x = a_0 + \frac{b_1}{a_1 + \frac{b_2}{a_2 + \cdots}} \quad a_n, b_n \in \mathbb{Z}$$

One often refers to the right-hand side as polynomial continued fractions (PCFs). For example, the Golden ratio can be represented as

$$1 + \frac{1}{1 + \frac{1}{1 + \cdots}}$$

The Ramanujan machine algorithm produced several *new* equalities[6] and new conjectures. The approach is to define a *space of meaningful PCFs* and reduce the problem to a search problem. There were two search algorithms suggested: one referred to as Meet-In-The-Middle[7], and the other is gradient descent, which is the same as that used in machine learning. An interesting step forward, as suggested in [18, 5.2], is whether one can more effectively create this space by using recent advances in NLP, via finding latent representations from widely available scientific corpus. This was used in the context of chemistry, [30].

Another key aspect of mathematics is creating definitions, [32]. [8] Examples of recent creations include the notion of *quasicategories* [21] and *perfectoid spaces* [23], both of which have brought great advancements in mathematics. We end this subsection with the open question: could AI create new *meaningful* definitions?

2.2 Machine learning for experimentation

There are many unexpected connections in mathematics. Most of these come from *experimentation* and *computation*. In number theory, one has the *j-invariant* function, which is a *modular function* on the upper half plane $\mathbb{H} := \{\tau \in \mathbb{C} : \text{im}\tau > 0\}$.

$$j(\tau) := \frac{1}{q} + 196884q + 2149370q^2 + \cdots \quad \text{where } q = e^{2\pi i \tau}$$

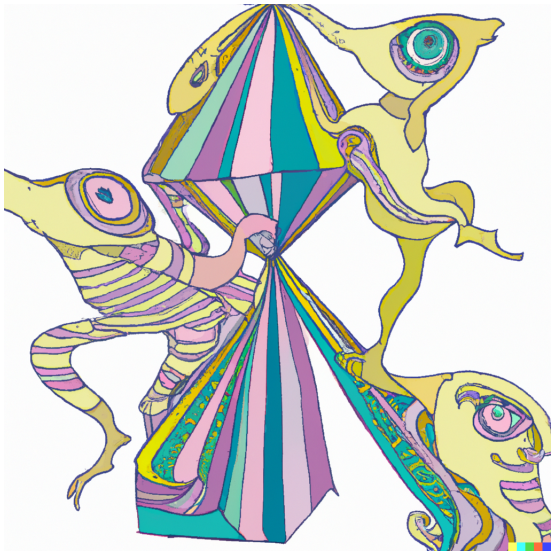
This is an object of close connection to *elliptic curves*, [24]. John-McKay found that these coefficients have close relations with dimensions of irreducible representations of Monster groups (which led to subsequent work vertex operator algebras.) This involved many linear algebra computations.

[5] Ramanujan is famous for finding surprising relations.

[6] They were proven soon after

[7] The approach here is intuitive: to compare two lists of sizes M and N of values, the time complexity is $O(MN)$. The algorithm makes time complexity as $O(M + N)$.

[8] This is in contrast to daily life. In Fodor et al's paper "against definition", the paper discusses how in the non-technical language one does not actually work with definitions.



From a more naïve and cruder point of view, we question whether ML can speed up experimentation in the context of mathematics. Examples include the work of Charton, [3], which trains transformers to perform numerical computation, [4], which *learns* mathematical properties of differential systems, and Bao et al, [1], which trains a machine to find properties of algebraic objects. Albeit varied attempts, such areas still remained quite unexplored.

2.3 Conjecture verification

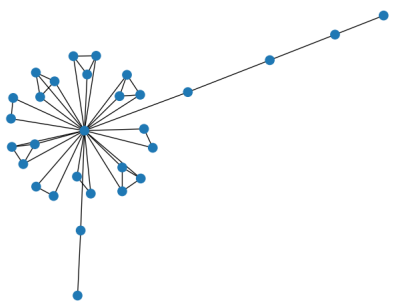
We give two instances of which methods in machine learning can *aid* in disproving conjectures.
In the field of solving PDE, neural networks has already been used in closely related context of physics, [20]. For instance, The physics of an equation is explained by a PDE [9]

$$P(u(x,t)) = 0$$

where u is a function of location x and time t. One then apply supervised learning to solve for u. This is the basis of Physics-informed neural networks (PINNs) and similar in spirit of the pipeline described in Fig. 2. This has inspired many subsequent works. A recent example in the same spirit is the work of Wang et al. [34].
Wagner's paper, [33], on the other hand, used reinforcement learning to guide the construction of counterexamples in combinatorics. If $G = (V = \{v_1, \ldots, v_n\}, E)$ is a n-vertex graph, we can form $D(G) := (d(v_i, v_j)_{1 \le i,j \le n})$ the *distance matrix*, whose eigenvalues we denote $\partial_1, \ldots, \partial_n$. [10] One application was to disprove the following conjecture

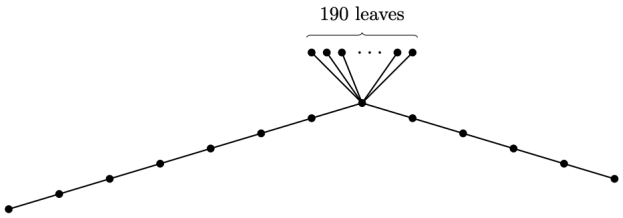
> *Conjecture: Let G be a connected graph on $n \ge 4$ vertices with diameter D, proximity $\pi := \frac{1}{n-1} \min_{v \in V} \sum_{w \in V} d(v,w)$ and distance spectrum, then*
>
> $$\pi + \partial_{\lfloor \frac{2D}{3} \rfloor} > 0$$

The algorithm ran with $n = 30$, produced a graph with value $\pi + \partial_{\lfloor \frac{2D}{3} \rfloor} = 0.4$:



[9] For instance, the viscous Burger's equation is $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = v\frac{\partial^2 u}{\partial x^2}$
[10] Study of eigenvalues goes back as early as 1970s.

Albeit not a counter-example, its shapes were sufficient for the author to come up with the general pattern and construct the actual counter-example,



Both diagrams above are sourced from [33]. This example illustrates again, the case that ML methods *useful* and are there to *guide* a human researcher - as depicted in 3, the incorporation of human feedback important.

3 Formal methods and the calculus of thoughts

Historically, the idea of inferring mathematical truths through a system of logical deductions goes back to Aristotle (c.f. Prior Analytics). In the 17th century, the polymath Leibniz (1647-1716) dreamed of a universal mathematical language, *Calculus Ratiocinator*, or the *Calculus of thoughts*, [22], where logical and mathematical truths could be encoded and systematically deciphered.

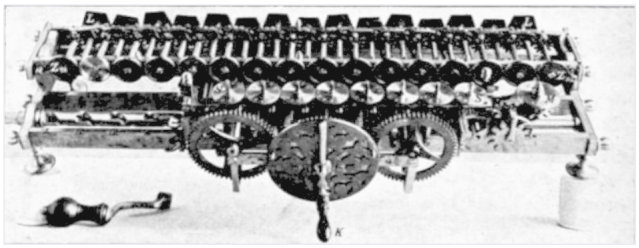


Figure 4: A photo (Wikipedia) of the *stepped reckoner* , the first calculator that is able to do all four arithmetic operations, created by Leibniz. Arguably the first physical manifestation of his philosophy.

The work of George Boole (1815-1864) in algebraic treatment of Aristotle's syllogisms and the work of Gottlob Frege in extending Aristotle's propositional logic to what is nowadays referred to as the firs order logic (c.f. Begriffsschrift) were the significant steps in the direction of logicizing mathematical reasoning in the 19th century. Frege's work in particular led to a philosophy of mathematics called *logicism* whose main tenet is that all mathematical facts are analytic. In the late 19th century and the early 20th century, the logicist/formalist approach led to the search for secure foundations for mathematics. ZFC set theory and later type theory emerged as the two main favourite logical foundations for mathematics. In the foundationalist approach to mathematics, every mathematical object is constructed from the most basic objects such as the type of natural numbers subject to the derivation rules of logic. For several important reasons, type theory, has been favored in formalizing mathematics in computers.

We remark that, for a long while, the foundationalist approach seemed to be at odds with the structuralist view of mathematics (e.g. the mathematics of Riemann, Dedekind, Hilbert, Bourbaki, and Grothendieck). This was mainly due to the ergonomical limitations of ZFC set theory in allowing a faithful encoding of mathematical structures. Often the encoding of mathematical structures (e.g. manifolds) are completely unrecognizable from their intuitive and informal definitions. Set theory although a complete foundation for mathematics, distorts the structures in the process of formalizing them: think of mathematical structures as specifications in programming language with high-level abstractions and the set theoretic encoding as binary code after compilation. We can learn little from the binary code about what the programs do.

Dependent Type Theory and Homotopy Type Theory offer a vastly superior encoding of mathematical structures. They are implemented in various Interactive Proof Assistants (ITP) such as Lean, Coq, Agda, Hol Light, Isabelle, etc.

3.1 Automated Reasoning

Automatic theorem provers (ATPs) have grown in past decades, and is used in conjuction with interactive proof assistants ITPs, 3.2, such as [8]. Historically, emphasis have been placed on SAT or Satisfyability modulo

theories (SMT) problems, mainly focusing imporving the efficiency and performance.[11] One fundamental task is *premise selection*, [19]. SAT solvers are employed in Lean and Isabeller to increase user's productivity: for instance automating long chains of reasoning steps to prove formulas (c.f. tactic `tauto` in Lean), as well as reasoning with and about equations and inequalities, and various algebraic decision procedures. In Isablle external first-order provers can be invoked through sledgehammer.

3.2 Interactive Theorem Provers

Interactive Theorem Provers (ITPs) are the reasoning engines for formalization of mathematical proofs. Various ITPs have been developed over the past decades, notably Mizar (1973), Isabelle (1994), Coq (1997), Agda (2007) and more recently Lean (2013).

Here we shall focus on the most popular proof assistant, namely Lean 4. Lean is a functional programming language and an ITP developed by Leonardo de Moura at Microsoft Research. At its core, Lean is a *proof checker*, a system that checks the validity of mathematical proofs. But it is more than that. It can find errors in the proofs, point them out, and explain their cause. If we leave a proof incomplete (e.g. by leaving the `sorry` placeholders in the proof, see the figure below) it will remind us (in the InfoView displayed on the right panel in the figure below) about all the proof obligations and the remaining goals when we click though the proof. Moreover, using automated reasoning, it can suggest proofs. In the figure below it suggests a lemma from the Lean's mathematical library `Mathlib` which completes the proof.

```
import data.nat.prime
import tactic.linarith

open nat

theorem infinitude_of_primes : ∀ N, ∃ p > N, prime p :=
begin
  intro N,
  let M := factorial N +1,
  let p := min_fac M,

  have pp : prime p := begin
    refine min_fac_prime _,
    have : factorial N > 0 := factorial_pos N,
    linarith,
  end,
  use p,
  split,
  -- Proof that p > N
  {by_contradiction,
    have h1 : p | factorial N +1 := by sorry,
    have h2 : p | factorial N := by sorry,
    have h : p | 1 := by sorry,
    library_search,
  },
  -- Proof that p is prime
  {exact pp, },
end
```

```
▼ Tactic state                    widget
1 goal                  filter: no filter
N : ℕ
M : ℕ := N.factorial + 1
p : ℕ := M.min_fac
pp : prime p
h : ¬p > N
h1 : p | N.factorial + 1
h2 : p | N.factorial
h : p | 1
⊢ false
▼ Messages (1)
▼ example2.lean:24:4
Try this: exact prime.not_dvd_one pp h
▶ All Messages (2)
```

Figure 5: Lean Formalization of the proof of the infinitude of primes by Euclid, circa 300 BC, in Book IX, Proposition 20 of Euclid's work "Elements." – created by the first author

[11] ATPS are often based on first-order logic, which limits their expressibility.

Over the last few years, we have seen breakthroughs in large-scale deployment of ITPs in the cutting-edge mathematical research such as the Liquid Tensor Experiment [2]. LTE is one of the first big tests where proof assistants can organize mathematical proofs at scale.

We now discuss the many potentials of LLMs. Despite the fact that LLMs are not good at reasoning, they can still be very useful when formalizing mathematics. They are great tools for pattern matching and this feature can be well combined with the architecture of the proof assistants. such as suggesting next proof steps inside a proof tactic bloc in Lean. This can be part of a feedback loop between a user using Lean and a LLM, this is explored further in 4. Another application which we envision is the better UIUX with Lean's mathematical library (Mathlib).[12] LLM-based tools can be useful in hallucinating and finding the formal statements which are possible matches for the natural language prompt. Both these tools (proof suggestions in tactic blocs, and searching lemmas/theorems based on natural language prompts) can be integrated with Lean as plugins in the VSCode editor.

3.3 Some Challenges for Interactive Theorem Provers

We discuss three challenges to further the field of ITPs. In the practice of mathematics, lots of time is spent on examples, concepts, and conjectures, as discussed in 2. These activities are not reflected in the final formal libraries mathematics. We need AI tools to help us with suggesting, forming and testing conjectures. This involves a good design of benchmarks.

Proof by analogies pose another interesting challenge. In mathematical discourse, there are often words such "analogously" and "similarly" which mathematicians broadly understand. Here is a challenge: How do we formalize proofs by similarity or even more difficult analogical proofs?

Proofs and intuition by picture and diagrams are even harder to formalize. But perhaps here we can find a useful application for multi-modal learning. Consider the formal Lean statements about geometric objects such as manifolds or fibre bundles. Can we train a model which spits out diagrams and pictures for geometric constructions from formal Lean statements.

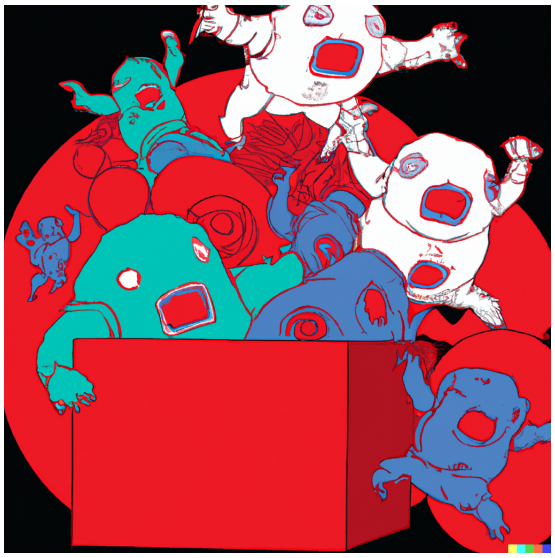
Lastly, as with many fields of applications, there is a lack of data to train neural network models for formalized proofs. Many techniques have been developed to compensate this difficulty, which were briefly discussed in 1.1. Given fixed compute training budget, it has been suggested that the expert iteration outperforms proof search only (e.g. Lean GPT-f which solved IMO problems with 36 layers and 774 million trainable parameters).

3.4 Autoformalization

Autoformalization is the task of turning informal descriptions to formal mathematics. Examples of formalization include the Kepler conjecture, the Four-Color theorem, and the Feit-Thompson theorem, giving certainty to the correctness. Autoformalization is seen as essential, [27, 3] in training language model. Compared to large body of corpus on the web, the Archive of Formal Proofs consists of less than 0.5% of the training data than large language model like Codex. Recent progress uses large language models, Wu et al. [37], which contrast to older series of work [17]. The main challenges of autoformalization include (1) bridging the logical gaps left in pen-and-paper proofs, (2) assuming the implicit contexts and assumptions, and (3) aligning informal definitions/concepts to formal ones.

[12]When proving statements, we know which lemmas and theorems of informal mathematics we need but we don't know whether they exist in the vast

4 Future directions: the scientific coauthor



As suggested in a similar form by a professor of Mathematics, Alex Kontrovich, and already existing in many fields of research, [25] , a natural holy grail is to have a scientific coauthor. A typical pipeline should look like:

$$\text{Prompt} \xrightarrow{(1)} \text{GPT?} \xrightarrow{(3)} \text{Response}$$
$$\text{GPT?} \overset{(2)}{\circlearrowleft} \text{Lean?}$$

In (1), the reader suggests an idea. In (2) the model runs inferences and checks against itself using interactive theorem provers. In (3) it makes it final inference. Lastly, the process is cycled.
Combining all of (3) , (2) into one seamless pipeline is ambitious and is far from current capabilities. Works in this direction include, *Draft, Sketch and Proof*, of Jiang et al, [13], and LeanDojo. [13] To name a few further desirable traits: proof explaining/outlining, proof repair, library design.

Lastly, whether in use of autoformalization, 3.4, or response generation, as 2, it is desirable that the output to be sound and reliable. LLM outputs can have unwanted repetition, [38], lack of generalization, lack of robustness [35], and even non-human interpretable, as can be seen in [37]. Some future avenues are suggested in [15, 7.2].

[13] Unfortunately, this still has many drawbacks.

References

[1] Jiakang Bao, Yang-Hui He, Edward Hirst, Johannes Hofscheier, Alexander Kasprzyk, and Suvajit Majumder. Hilbert series, machine learning, and applications to physics. *Physics Letters B*, 827:136966, 2022.

[2] Reid Barton, Johan Commelin, Patrick Massot, Scott Morrison, Adam Topasz, and Peter Scholze. Liquid tensor experiment. https://github.com/leanprover-community/lean-liquid, 2021.

[3] François Charton. Linear algebra with transformers. *ArXiv*, 2022.

[4] François Charton, Amaury Hayat, and Guillaume Lample. Learning advanced mathematical computations from examples. *ArXiv*, 2021.

[5] Maurice Chiodo and Toby Clifton. The importance of ethics in mathematics. *European Mathematical Society Magazine*, (114):34–37, 2019.

[6] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

[7] Philip J Davis, Reuben Hersh, and Elena Anne Marchisotto. *The mathematical experience, study edition*. Springer, 2012.

[8] Burak Ekici, Guy Katz, Chantal Keller, Alain Mebsout, Andrew J. Reynolds, and Cesare Tinelli. Extending SMTCoq, a certified checker for SMT (extended abstract). *Electronic Proceedings in Theoretical Computer Science*, 210:21–29, jun 2016.

[9] Timothy Gowers, June Barrow-Green, and Imre Leader. Viii.6 advice to a young mathematician. 2010.

[10] Michael Harris. What is "human-level mathematical reasoning"?, part 1: which humans? Nov 2021.

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *ArXiv*, 2021.

[12] Reuben Hersh. What is mathematics, really? *Mitteilungen der Deutschen Mathematiker-Vereinigung*, 6(2):13–14, 1998.

[13] Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *ArXiv*, abs/2210.12283, 2022.

[14] Imre Lakatos. *Proofs and refutations: The logic of mathematical discovery*. Cambridge university press, 2015.

[15] Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. A survey of deep learning for mathematical reasoning. *ArXiv*, 2023.

[16] Alison Pease and Ursula Martin. Seventy four minutes of mathematics: An analysis of the third mini-polymath project. *ArXiv*, 2012.

[17] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *ArXiv*, 2022.

[18] Gal Raayoni, Shahar Gottlieb, Yahel Manor, George Pisha, Yoav Harris, Uri Mendlovic, Doron Haviv, Yaron Hadad, and Ido Kaminer. Generating conjectures on fundamental constants with the ramanujan machine. *Nature*, 590(7844):67–73, feb 2021.

[19] Markus N. Rabe, Dennis Lee, Kshitij Bansal, and Christian Szegedy. Mathematical reasoning via self-supervised skip-tree training. *ArXiv*, 2020.

[20] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[21] Charles Rezk. Introduction to quasicategories. *Lecture Notes for course at University of Illinois at Urbana-Champaign*, 2022.

[22] Daniel M. Rice. Calculus of thought: Neuromorphic logistic regression in cognitive machines. 2013.

[23] Peter Scholze. Perfectoid spaces. *Publications mathématiques de l'IHÉS*, 116(1):245–313, 2012.

[24] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.

[25] Chris Stokel-Walker. Chatgpt listed as author on research papers: many scientists disapprove. *Nature*, 613:620–621, 2023.

[26] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[27] Christian Szegedy. A promising path towards autoformalization and general artificial intelligence. In *International Conference on Intelligent Computer Mathematics*, 2020.

[28] Terence Tao. Embracing change and resetting expectations. ai-anthology/terence-tao/, 2023.

[29] William P. Thurston. Three dimensional manifolds, kleinian groups and hyperbolic geometry. *Bulletin of the American Mathematical Society*, 6:357–381, 1982.

[30] Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A Persson, Gerbrand Ceder, and Anubhav Jain. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95–98, 2019.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, 2017.

[32] Shlomo Vinner. The role of definitions in the teaching and learning of mathematics. 2002.

[33] Adam Zsolt Wagner. Constructions in combinatorics via neural networks. *ArXiv*, 2021.

[34] Yongji Wang, Ching-Yao Lai, Javier Gómez-Serrano, and Tristan Buckmaster. Asymptotic self-similar blow-up profile for three-dimensional axisymmetric euler equations using neural networks. 2023.

[35] Sean Welleck, Peter West, Jize Cao, and Yejin Choi. Symbolic brittleness in sequence models: on systematic generalization in symbolic mathematics. *ArXiv*, 2022.

[36] Geordie Williamson. Is deep learning a useful tool for the pure mathematician? *ArXiv*, 2023.

[37] Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *ArXiv*, 2022.

[38] Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation. *ArXiv*, abs/2206.02369, 2022.