BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 61, Number 2, April 2024, Pages 211–224
https://doi.org/10.1090/bull/1833
Article electronically published on February 15, 2024

MATHEMATICAL REASONING AND THE COMPUTER

KEVIN BUZZARD

Abstract. Computers have already changed the way that humans do mathematics: they enable us to compute efficiently. But will they soon be helping us to *reason*? And will they one day start reasoning themselves? We give an overview of recent developments in neural networks, computer theorem provers, and large language models.

1. Using computers to compute

1.1. **Humans as computers.** In the early 1600s the Scottish mathematician John Napier constructed the first tables of logarithms. In his book [Nap89, p. 46], Napier reflected that "the learned, who perchance may have plenty of pupils and computers", would be able to take things further. You might think that having plenty of computers was rather uncommon in the 1600s, but back then the word "computer" referred to a human who would calculate.

The achievements of human computers should not be underestimated. Let us just give one example. Two such computers, Felkel and Vega, published tables of the factorizations of the positive integers up to 408,000 in 1776 and 1777.[1] As a consequence it became possible to create tables of the number of primes less than N, for N in this range. Based on these tables, Legendre conjectured in the late 1700s that the number of primes less than N was approximated by a function of the form $N/(C\log(N)+D)$. In the 1808 edition of his book *Essai sur la théorie des nombres* on number theory [Leg08], he conjectured that $C=1$, and the statement of the prime number theorem was born (although the young Gauss had apparently independently come up with the same conjecture some years earlier). The theorem remained open for 100 years and its resolution, using complex analysis, by Hadamard and de la Vallée Poussin was a triumph of late nineteenth century mathematics.

1.2. **Machines as computers.** Early electronic computers were the size of a house and orders of magnitude less powerful than a modern phone. By the time the University of Cambridge bought an EDSAC II machine in 1957, they were small enough to fit into a large room, and in those pre-transistor days they relied on bulky vacuum tubes (or "valves", as they were called in the UK) as switches. During the day, the EDSAC II was being used by oceanographers to do calculations which led the way to modern plate tectonic theory, but in the evenings the young Peter Swinnerton-Dyer would show up with a pile of punch cards and hand them to the

Received by the editors September 13, 2023.
2020 *Mathematics Subject Classification.* Primary 68T01; Secondary 68T07, 68V15, 68V20.

[1] Although the title "Tafel aller einfachen Factoren der durch 2, 3, 5 nicht theilbaren Zahlen von 1 bis 10,000,000" of their work indicates that they had planned to go much further.

operator; the machine would then spend all night counting the number of solutions N_p to equations such as $y^2 = x^3 + x + 37$ modulo prime numbers $p < 100$—at least if they were lucky. Swinnerton-Dyer describes the days when the only output handed to him the next morning would be a printout saying "error in punch card 4", whereupon he would retire to his office to try and figure out where the offending hole was.

The result of this laborious process was a collection of data. At the time, data production was all the computer could do—it was up to the humans to have the ideas. Birch and Swinnerton-Dyer tried to make rigorous sense of the idea that "the more solutions a plane cubic had in the rationals, the more solutions it should have on average modulo prime numbers". Let A and B be integers such that the complex roots of x^3+Ax+B are distinct. It had long been known that the rational solutions to the plane cubic equation $y^2 = x^3 + Ax + B$, plus the "point at infinity", formed an abelian group. Mordell had proved that this group was of the form $\mathbb{Z}^r \times T$ with T finite and r, the *rank* of the cubic, a natural number. The insight of Birch and Swinnerton-Dyer, coming from the computations, was that if p is prime and N_p is the number of the solutions to the plane cubic mod p, then the order of the growth of $\prod_{p\leq N} \frac{N_p}{p}$ should be $(\log N)^r$. This turned into the observation that the order of vanishing of the L-function of the curve at $s = 1$ should be r, and the conjecture in its current form was born. It remains open.

The delineation of the tasks in the above stories is clear. The computer performs the computations, and then the mathematician takes over, providing their insights and interpretations. The computer is simply a tool with which one can collect evidence at superhuman speed. Since the 1960s computers have become far smaller, far faster, and far more available to the mathematician. Nowadays, using computers to collect experimental evidence in mathematics is commonplace.

But generating more computational evidence for the Riemann hypothesis or the Birch and Swinnerton-Dyer conjecture will not win you a Fields Medal. The currency of the pure mathematician is the *theorem*. In this article we shall be concerned with new uses of computers, going beyond the merely mechanical work of computing for us. Can machines help us to think? It seems so! We will explain some examples. Can they even think for themselves? Yes—in some sense—but only at a very rudimentary level and only with certain kinds of problems. How far will things go? Nobody knows. Could machines one day start proving interesting deep conjectures by themselves? Some think so, others are more skeptical. Certainly there is no evidence of this happening right now—currently this idea is science fiction. Rather than speculating about the future, we will restrict the scope of this article to a survey of what has happened, and what is currently happening.

There are three distinct topics which we shall discuss in the next three sections:

- Use of neural networks as a search tool for theorems, conjectures, and counterexamples.
- Automated and interactive theorem provers and the mathematics they currently understand.
- Large language models such as ChatGPT and their efforts to do mathematics.

This article is written for mathematicians, and we will assume no background knowledge in any of these areas. In particular, we will not discuss the technical advances in computer science which have enabled these tools to exist. Instead we will explain what these things *do*, and how they are being used by mathematicians.

I strongly recommend Tom Hales's article "Mathematics in the Age of the Turing Machine" [Hal14]. One role of this article is to give an update of the area since Hales's article was written.

2. Neural networks in mathematics

In this section we discuss various uses of neural networks in mathematics. We start by giving a basic description of deep learning, one of the things a neural network can be used for.

2.1. **Introduction to deep learning.** The kind of problem which deep learning (or just "learning") attempts to solve is the following. We have two finite-dimensional real vector spaces $V = \mathbb{R}^m$ and $W = \mathbb{R}^n$ and a subset S of V, which is the domain of a (nonlinear) function $f : S \to W$. We have a finite table of pairs $(s, f(s))$ consisting of many elements $s \in S$ and their images under f. The table might have been expensive to calculate (it may have been computed using an algorithm which takes a long time to run, or it may have been produced manually by humans). What we would like to do is to make a computer program which can "learn" f, enabling us to compute its value on many more elements of S, quickly and with high accuracy. Let us give some examples.

A computer might internally represent a colour as an "RGB value", that is, as three numbers between 0 and 255 representing the amount of red, green, and blue present in the colour. A 1000×1000 digital photo can be represented as a list of million colours, and hence as a list of three million integers between 0 and 255. Thinking of these integers as real numbers, we can think of a digital photo as an element $s \in \mathbb{R}^{3000000}$. Let S be the set of 1000×1000 digital photos on the internet. We define $f : S \to \mathbb{R}$ by $f(s) = 1$ if s is a photo which contains a bicycle, and $f(s) = 0$ otherwise. When a website wants to check that you are a human, it might give you a collection of photos and ask you which ones contain bicycles. The website might give many humans the same question, and define f using the most popular answer. As time goes on, we generate a table of pairs. The question is whether we can now write a computer program which can accurately predict whether a given picture contains a bicycle.

Here is an example which might be more relevant to readers of this article. Mathematicians generate tables of data associated to mathematical objects. For example the L-functions and modular forms database [LMF23] contains a very large list of elliptic curves over the rationals. An elliptic curve over the rationals has some kind of "canonical" minimal model, represented by 5 integers $(a_1, a_2, a_3, a_4, a_6)$ (the subscripts are to do with weights of the corresponding modular forms and the absence of a_5 is not a typo). To an elliptic curve over the rationals one can associate the rank (a natural number). So here we can set $V = \mathbb{R}^5$, we can let S be the subset of V corresponding to 5-tuples of integers such that the corresponding curve $Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$ is in "canonical" minimal form, and we can define $f(a_1, a_2, a_3, a_4, a_6) \in W = \mathbb{R}$ to be the rank of the corresponding elliptic curve. Our table of values of f will be the table extracted from the L-functions and modular forms database. In this case we already have a computer program which can compute f; However, the problem is that for some inputs it might be very slow, or may not terminate at all. The question is whether we can write a computer program which computes f more quickly, with high accuracy.

Given V and W and the data of finitely many input and output values of f, an appropriate neural network will, after a period of "training", produce a function from V to W which is some kind of attempt to approximate or "learn" f from the values we have trained on. The function is "explicit" in the sense that one can feed in numerical elements of V and quickly get out elements of W. We will not go into the architecture of neural networks or an explanation of how this function is constructed.

To get an idea of how accurate the network's guess for f is, the system is typically only trained on most of the available data (say, 90%). When the network has come up with a function interpolating the data it has seen, we can feed in the remaining 10% and compare the function's output with known results. A high success rate indicates that the network is accurately learning the function, and one can go on to now try running the function at points of S not in our original dataset.

Geordie Williamson has been trying to understand the kind of mathematical problems which one might expect this sort of method to succeed. In some sense it is surprising that it succeeds at all, however, over the last few years computers have become extremely good at deciding whether or not a given digital image contains a bicycle, so there is certainly some merit in the idea. A team led by Yang Hui He tried to use the techniques to compute ranks of elliptic curves [ABH19], and was less successful. Williamson has observed that one key difference between the two problems is that the bicycle problem has a continuity property which the elliptic curve problem lacks: the output is unchanged by small changes in the input. If one slightly changes the colour of one pixel in a digital image, then the image remains essentially the same, so the answer to the question of whether it contains a bicycle will not change. However changing one coefficient of an elliptic curve can completely change the arithmetic of the curve; if you completely understand all the rational solutions to $y^2 = x^3 + 37$ (a rank two curve) then this tells you nothing about the rational solutions to $y^2 = x^3 + 38$ (a rank one curve). However, Yang's team achieved a success rate of well over 90% at rank prediction when instead of taking the Weierstrass equation as input, they took the first few hundred coefficients of the associated modular form. Yang's team has also had far more success [He21, Section 4.3.1] in predicting whether the natural number invariant $h^{2,1}$ of a Calabi–Yau threefold over $\mathbb{Q}$ is greater than 50 or not; perhaps this is because the latter invariant is locally constant on the moduli space.

A tool which quickly predicts whether a certain invariant associated to a Calabi–Yau manifold is greater than 50, with 93% accuracy, can certainly be regarded as progress, but one can certainly argue that such a system is not producing *theorems*, just different ways of doing computations. Let's now talk about three examples where neural networks have actually helped humans to *prove new theorems*. One particularly nice fact about the work below is that in every case, mathematicians have been involved in the writing of the papers, and have ensured that the work is readable by mathematicians who may not have detailed knowledge of the theory of neural networks.

2.2. **Neural networks and knot theory.** The first example I discuss is the paper "The signature and cusp geometry of hyperbolic knots", by Davies, Juhász, Lackenby, and Tomasev [DJLT22]. This was a collaboration between two Oxford mathematicians and two computer scientists from the tech company DeepMind. The mathematicians are low-dimensional topologists, and the goal was to see if computers could help in the discovery of a new theorem in knot theory. There are countably infinitely many knots (up to ambient isotopy), and they can be tabulated in a reasonable way. Mathematicians over the years have discovered invariants of knots; some are "discrete" (for example, the signature of a knot is an integer), and some are "continuous" (for example, the volume of a (hyperbolic) knot is a real number). In fact, there is a more fine-grained classification here: there are quantum invariants, topological invariants, and gauge-theoretic invariants. There might be many relationships between these invariants, and the first question to ask is what kind of a theorem in this area would be interesting to mathematicians. One example of an interesting result might be some kind of link between continuous and discrete invariants, as this might be regarded as more profound.

With this in mind, the team used open source software to generate a huge table of knot invariants, and then used a neural network to try and spot hitherto unknown relationships between these invariants, and in particular to search for relationships between invariants with different origins. The team found that an appropriate neural network was able to accurately predict the signature of a knot (a discrete, cohomological invariant) given the data of the continuous invariants in the tables. The computer scientists on the team were then able to analyze the parameters of the neural network and isolate which of the continuous invariants were playing the most important role in the prediction. It was not possible to meaningfully understand "in human terms" the function which the neural network had learned, however, the knowledge of which continuous invariants were playing a major role was enough of a clue to continue. The baton was then passed on to the mathematicians, who came up with conjectural ideas relating the signature to the relevant continuous invariants. Their first conjecture was false, but they managed to prove the second one:

Theorem (Davies, Juhász, Lackenby, Tomasev). *There exists a constant c such that for any hyperbolic knot K,*

$$|2\sigma(K) - \mathrm{slope}(K)| \leq c\,\mathrm{vol}(K)\,\mathrm{inj}(K)^{-3}.$$

The theorem relates the signature $\sigma(K)$ of a knot to its slope, volume, and injectivity radius. For more information about this result, I would recommend Lackenby's talk at the 2023 IPAM conference at UCLA [Lac]. Also highly recommended is Ernie Davis's rebuttal [Dav21] of the work, who argues that the role of deep learning in this story has been over-stated and that a conventional statistical analysis would probably have sufficed. Davis also has something to say about our next example, coming from representation theory.

2.3. **Neural networks and representation theory.** The same DeepMind team collaborated with other mathematicians on the combinatorial invariance conjecture, a problem in representation theory. Here we briefly discuss the results, contained in the paper [BBD$^+$22] by Blundell, Buesing, Davies, Velickovic, and Williamson. Given a pair of elements of a finite symmetric group S_n, one can associate two invariants to the pair. The first, the Bruhat graph, is a directed graph which is typically a very unwieldy object but is easy to compute. The second, the Kazhdan–Lusztig polynomial, is a very simple object but is typically very hard to compute. The idea behind the conjecture is that the Kazhdan–Lusztig polynomial can (somehow) be computed from the Bruhat interval graph.

The basic idea behind the new approach is similar to the knot theory work. First, one computes a large database of Bruhat interval graphs and Kazhdan–Lusztig polynomials. Next, one encodes the graphs as sequences of numbers and thus as elements of a large real vector space. Then, one tries to train a neural network to predict the Kazhdan–Lusztig polynomial. Once one has made the right design decisions, it turns out that the neural network can become good at this task. One then attempts to analyze the neural network for hints about what it is doing. Humans managed to distill from the network the idea that so-called extremal reflections in the graph were playing an important role in the prediction, which led them to a new definition in the theory, namely the concept of a hypercube decomposition (see [BBD$^+$22, Section 3.4]). This then led to a new formula for Kazhdan–Lusztig polynomials, which, unfortunately, depended on slightly more than the Bruhat decomposition graph, so did not resolve the conjecture. However, it also led to a new conjecture, which implies the combinatorial invariance conjecture and which the authors were able to check in over a million cases.

Williamson says the following about this work: “For me these findings remind us that intelligence is not a single variable, like an IQ number. Intelligence is best thought of as a multi-dimensional space with multiple axes: academic intelligence, emotional intelligence, social intelligence. My hope is that AI can provide another axis of intelligence for us to work with, and that this new axis will deepen our understanding of the mathematical world.”

Davis, on the other hand, argues in [Dav21] that although deep learning did play a key role in this work, it should be viewed as just “another analytic tool in the toolbox of experimental mathematics rather than a fundamentally new approach to mathematics”. Although, as he goes on to say, “How powerful a tool it is and how broadly applicable remains to be seen”. Time will tell.

2.4. **Searching for counterexamples in graph theory with neural networks.** Moving away from deep learning, another technique in this area is *reinforcement learning.* In contrast to deep learning, where one has the data and wants to learn the function, in reinforcement learning one has the function (a “reward function”) and one wants to construct appropriate data to maximize the reward. These sorts of techniques can be used to teach computers how to play video games, for example. But they can also be used to search for interesting mathematical objects. Adam Wagner used reinforcement learning to search for counterexamples in graph theory. We cite one example from his beautiful paper [Wag21]. A conjecture from 2010 stated that if G is a connected graph with $n \geq 3$ vertices, having largest eigenvalue λ_1 and matching number μ, then $\lambda_1 + \mu \geq 1 + \sqrt{n-1}$. Wagner considered graphs on $n = 19$ vertices, with the source vector space V encoding the presence of edges between these vertices. The network continually modified the input graph, attempting to minimise $\lambda_1 + \mu$. Eventually the network found a graph for which $\lambda_1 + \mu$ was so small that the inequality was violated. There are several other examples in the paper, including illustrative examples where the network did not directly find a counterexample to a given question, but started to make graphs which had a clear structure, enabling the human to take over and finish the job. An overview of the work is given in Wagner’s IPAM talk [Wag].

The above examples indicate novel applications of these new tools, however, it might be the case that the applicability of these tools is limited to areas where computation is possible and tables exist, so they might not be useful if you are interested in, say, perfectoid spaces. In the next section however, we discuss tools which seem to be able to access all of pure mathematics.

3. Automated and interactive theorem provers

Our second example of new uses of computers in mathematics is an overview of the capabilities of ATPs (automated theorem provers) and ITPs (interactive theorem provers). These are both systems which know or can be told various axioms of a theory, and have a language rich enough to express theorems in the theory. The job of an automated theorem prover is to attempt to automatically generate proofs of theorems in the theory. The job of an interactive theorem prover is to offer a powerful front end where humans can type in their own proofs of theorems and the system will check that they are valid.

Despite the superficial similarities between ATPs and ITPs, in practice they are used to perform very different tasks. Let us first give a brief overview of ATPs.

3.1. **Automated theorem provers.**

Automated theorem provers are typically used to prove (possibly very complex) theorems in a simple theory, perhaps using first order logic. An example might be the first order theory of groups. The system is given the axioms for a group, and a formula which is true in all groups—for example, $(ab)^{-1} = b^{-1}a^{-1}$—and it attempts to prove the theorem from the axioms. These systems play an important role in software verification, but we shall focus on their applications in mathematics.

One of the early successes of these systems was a proof of the Robbins conjecture. In the 1930s, Robbins came up with a collection of axioms which he conjectured were an axiomatisation of Boolean algebras; the hard part was verifying the axioms of a Boolean algebra from Robbins's axioms. William McCune proved Robbins's conjecture in 1996, using an ATP; the story made the *New York Times*. Allen Mann's write-up [Man] contains a human-readable 14-page account of McCune's computer proof: in particular, the output of the ATP is comprehensible to a human in finite time. Within a year, Bernd Dahn had simplified the proof and the resulting 7-page paper [Dah98] was published in the Journal of Algebra.

However these systems are no longer limited to discovering short proofs. In 2012, Phillips and Stanovský, and Waldmeister (an ATP) proved [PS12] that a Bruck loop[2] with abelian inner mapping group has nilpotency class at most 2. The proof generated by the ATP was over 30,000 lines long and when written out in a locally human-readable format (surely no human would ever read the whole thing) fills just over 1,000 PDF pages. Note, of course, that the proof was not found by "brute force search"—these systems are far cleverer. We cannot, however, rule out the existence of a far shorter, and perhaps even human-comprehensible, proof.

While the proofs which these systems can discover can be huge, the *nature* of these proofs seems to be inherently very "low-level". A common usage nowadays for automated theorem provers is as components of interactive theorem provers, which we now describe.

[2] A loop is basically a group with the associativity axiom dropped.

3.2. **Interactive theorem provers.** Like automated theorem provers, interactive theorem provers (ITPs) know the rules of a logic and the axioms of a theory, and can be used to prove theorems in the theory. However. typically now the logic and theory are richer and more complex, making it practical to formalize cutting-edge modern mathematics. One can think of an ITP as a programming language where the code corresponds to mathematical definitions, theorems, and proofs. The big difference between interactive and automated theorem provers is that in an ITP, the user is expected to type in the key ideas of the proof themself. A modern ITP will typically have tactics, which are little computer programs capable of filling in steps which are clear to a mathematician but which would be tedious to prove from the axioms directly. For example, a proof that $(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$ directly from the axioms of a ring is surprisingly long, because for example the step after expanding out the brackets—"rearrange these eight terms into the right order and put all the brackets in the right place"—corresponds to many applications of commutativity and associativity of addition and multiplication. A ring theory tactic would perform this computation in just one line and hide these tedious arguments from the user. Tactics make the process of teaching more advanced proofs to the ITP feasible in a reasonable time. However, it is currently still far more time-consuming than writing the analogous results in a LaTeX document, even for expert users. Bringing down the time it takes to write a proof in an ITP to something nearer to the time it takes to write it on paper would surely increase the use of these systems by mathematicians. Novel ideas are being tried all the time. Paulson and Blanchette created a "hammer" tactic for the Isabelle/HOL ITP, which attempts to prove intermediate results in the ITP by sending them off to an ATP and then attempting to construct a proof valid in the ITP from the output. Abby Goldberg, Druv Bhatia, and Rob Lewis wrote the `polyrith` tactic for Lean, which calls an instance of the Sage computer algebra system on the cloud to answer a certain question in algebra, and then gets Lean to formally check the answer. Finally, Scott Morrison has written a general purpose `sagredo` tactic for Lean which establishes a dialogue with a large language model (more on these later) and attempts to use it to generate a Lean proof.

Historically, ITPs were used by computer scientists to verify basic results in undergraduate level mathematics. This century users became more ambitious. In 2004, Jeremy Avigad led a team which verified the prime number theorem in Isabelle, arguably the first "serious" mathematical result verified in an ITP. Within a few years there had been several more. Georges Gonthier showed that the Coq theorem prover was powerful enough to formally verify the four colour theorem. The original Appel–Haken proof was a computer-assisted proof in the sense that it relies at some point on a brute force computation which is too large to be done by humans. Indeed, every currently known proof of the four colour theorem is computer-assisted. This raises the question of whether the computer program used to finish the job has bugs. Gonthier's work is a full proof of the result in the Coq theorem prover, and, in particular, the computer-assisted part of the calculation is verified as bug-free, according to the prover. This work was a tour de force at the time. One could be paranoid and suggest that we have reduced the proof of the four colour theorem to the assertion that Coq has no bugs. However, we have done more than this: one can run the compiled output of Gonthier's code through an independent type checker, a very simple program which one can check by hand

and which just answers the question "does this output correspond to a proof of this theorem from these axioms?" (in contrast to an ITP, which contains a wide array of tools for actually constructing proofs). If the type checker is happy with the proof produced by Coq then the question of whether Coq's kernel has bugs is now no longer relevant. And if you are concerned about, for example, the compiler used to compile the type checker, or the chipset of the computer used to run it, then you can just write another typechecker in a different language and run it on a different computer. At any rate, you can choose your level of paranoia; the conclusion of the work is, at the very least, the assertion that the proof of the four colour theorem has now been checked to a *far* higher standard than before Gonthier's work.

Having demonstrated that ITPs could be used to check computer-assisted proofs, a team led by Gonthier went on to demonstrate that they were also capable of checking twentieth century Fields Medal–level mathematics. The team typed a full proof of the Feit–Thompson odd order theorem into Coq over a period of six years. The original proof of the theorem was part of the reason Thompson was awarded the Fields Medal in 1970.

Thomas Hales then showed that the systems were powerful enough to handle a modern computer-assisted proof. His proof with Ferguson [Hal05] of the Kepler conjecture was computer-assisted, and the Annals would only publish the result with a disclaimer saying that the referees were not able to say with 100% confidence that the proof was complete. Hales's response was to put together a team of people who formalized the entire proof in an ITP and thus put any question of a bug beyond all reasonable doubt. The disclaimer has since been removed.

More recently, the Lean community have been demonstrating that it is now possible to formalize modern research level mathematics "in real time". Recent examples are the 2019 Dahmen–Hölzl–Lewis formalization [DHL19] of the 2016 Ellenberg–Gijswijt solution to the cap set problem [EG17], the 2022 Gadgil–Tadipatri formalization [Gad] of Gardam's 2021 disproof [Gar21] of the Kaplansky unit conjecture, and the 2022 Bloom–Mehta [BM] formalization of Bloom's 2021 resolution [Blo21] of a conjecture of Erdős and Graham on continued fractions. Note that in 2021, Thomas Bloom knew nothing about Lean, but he was introduced to it by Bhavik Mehta, and between them they formalized the full proof before the paper had even received a referee's report. There is currently ongoing work of Mehta and Yaël Dillies (currently an undergraduate at the University of Cambridge) formalizing important 2022 and 2023 results in additive combinatorics, adding to the evidence that, at least in some areas of mathematics, real-time formalization is becoming a reality.

The results cited above are certainly complex mathematics, but the arguments remain entirely in the domain of "low-level" (prime numbers, planar graphs, finite groups, balls in 3-space, fractions, etc.). More recently the Lean community has been attempting to engage with modern and far more subtle mathematical objects, such as perfectoid spaces and the homological algebra of condensed abelian groups. Johan Commelin and Adam Topaz successfully led a project called the "Liquid Tensor Experiment" [Sch22], whose goal was to formalize a proof of a theorem of Clausen and Scholze in the Lean ITP; see [Com] and [CT23] for more details. One notable consequence of the work was that the dependency on the theory of stable homotopy groups of spheres was removed during the formalization process; the computer kept track of precisely what was needed from this area, and ultimately it turned out that one could make do with a lot less than the theorem of Breen and Deligne initially cited by Clausen and Scholze.

Those concerned about whether the nature of these systems mean that they are limited to algebraic rather than geometric results should rest assured: Massot, van Doorn, and Nash have formalized a proof in Lean of sphere eversion [MvDN22], the fact that one can turn a sphere inside out in 3-space (allowing it to pass through itself, but with creasing or cutting not allowed). The success of these projects is an indication that the sky now seems to be the limit when it comes to the mathematics which these systems can handle, although there is still plenty of scope for formalization of geometric results; in particular, arguments which rely heavily on pictures will present an interesting challenge. A related question is how long the process of formalization takes, and whether this time varies between subfields (in particular, whether formalizing geometry takes longer than formalizing arithmetic). Another important question is how to decrease this time.

The Liquid Tensor Experiment and the sphere eversion project were both built on Lean's mathematics library `mathlib`; this is a library which now contains essentially all of a standard undergraduate mathematics degree, as well as many results at masters level in algebra, number theory, and algebraic geometry. The library grows daily, and more and more mathematicians are getting involved. One way of thinking about the library is that it is a twenty-first century Bourbaki, always working in a large generality and focusing mostly on theorems rather than examples.

ITPs and ATPs turn mathematics into a game, like chess or go. Breakthroughs in AI have created machines which are superhuman at both chess and go, and this raises the possibility that future AIs will become superhuman at mathematics, proving theorems which humans are interested in but which they cannot solve themselves. This idea is still science fiction right now; while AIs can write code in these languages, their current level is that of a strong schoolchild or average first-year undergraduate. We highlight two of the difficulties present in translating the successes of the game domain into mathematics. Firstly, mathematics is a single-player game rather than a two-player game; it is difficult to ascribe a "score" to a partial proof of a theorem, other than the obvious "0%" if you haven't finished and "100%" if you have; pure mathematicians may well not be interested in a theorem which is "95% proved", if this even has any meaning. And second, mathematics has an infinite and high-dimensional action space, by which I mean that at any point in a proof there are infinitely many things which you can do next, including applying any one of thousands of applicable lemmas, many with large numbers of parameters. It is difficult right now to "guide" the systems towards a successful proof. In Section 4 we will discuss large language models such as ChatGPT, and, in particular, ask whether these systems are capable of helping with this problem.

Here are some more realistic near-term goals for these systems. Firstly, one could imagine them powering interactive error-free textbooks. Recent work of Massot and Miller [Mas] has shown that this is already becoming a reality; they have created a system which takes a Lean proof as input and outputs a web page containing a human natural language proof. The viewer can then interact with the proof and "unfold" arguments right down to the axioms of the system. This is only one carefully curated example but it is strong evidence that the technology is ready for a much larger project.

Secondly, one could imagine "chatbots" who are experts in a given domain such as algebraic geometry, backed up by a database such as a formalized version of the Stacks Project [Sta18] and where students are able to query the system for theorems, examples and counterexamples. If the systems work in the language of the theorem prover in question, then the user would have to learn this programming language. We teach mathematics undergraduates python—it is not unreasonable to imagine that we could also be teaching them a language such as Lean (and indeed such courses are starting to appear around the world). However, such a system would be much easier to use if the bots were able to understand queries and respond in English, so perhaps it is time to end this section and begin to talk about recent developments in AI generation of natural language.

4. Large language models

Large language models are neural network-based systems which provide a probability distribution as an answer to the question "what is the next token in this sequence of tokens?" (where a token might be a word or perhaps a letter or number). Applications of this technique include things such as "write a paragraph of text answering a given question" and "write computer code in the language of an ITP proving a given theorem". Recent breakthroughs in this area have got the systems to the state where they are capable of writing coherent, correct, and relevant English sentences, something which a few years ago was a hard unsolved problem. Right now the most famous of these systems is undoubtedly OpenAI's ChatGPT [Ope], a large language model which one can access for free on the internet and which seems to have opened the eyes of the general public to the current power of these systems. The systems are trained on extremely large bodies of text (e.g., the entire internet) and are now able to respond to many questions "in the way a human would". Computer scientists inform us that progress in this area is "exponentially fast" right now, and some would like to infer from this that progress will continue to be exponential for some time, which will presumably quickly render the contents of this section out of date. For now, let us ask: can these things currently do mathematics?

If you ask ChatGPT to prove that there are infinitely many primes, then (because it is trained on the internet) it will happily rattle off some variant of Euclid's proof. It may get a little confused in its treatment of the case where $1+\prod_{i=1}^n p_i$ is composite, but one could imagine that a random proof that one finds on the internet might also be slightly confused at this point. However, what happens when we ask it for a proof of the much harder result that there are infinitely many primes which end in 7? Like a first year undergraduate faced with this question, it thinks about the proof which it knows of the infinitude of primes, and tries to generalize it to this situation. Unfortunately, it is not true that an arbitrary number of the form $10\prod_i p_i+7$ must have a prime factor which ends in 7, but this does not stop the model from confidently arguing that this is the case, like the undergraduate might. Worse—if asked to prove that there are infinitely many even numbers which end in 7, the system might well try a similar strategy and write a nonsensical paragraph. The reader should feel free to try these examples themselves in whatever large language model they have access to, to get a feeling for what these things can or cannot do right now.

At the time of writing, a big problem with these systems when it comes to writing mathematics in a "natural language" such as English is that they will happily assert false statements. Multiple systems now exist, and some (for example, GPT4 [Ope23]) have performed competently in difficult school level multiple choice mathematics exams such as AMC12. Here, a false statement is not the end of the world—it might lose you a mark, but you can still certainly end up passing the exam. But one false assertion in a computer-generated proof of the Riemann hypothesis and the entire edifice comes tumbling down, so if we are to take these systems further, then unjustified or incorrect claims need to be eliminated. Some people hope that better training will somehow drastically lessen the likelihood of the system emitting false statements, but we are not there yet.

A different approach would be to instead start training the systems to write not in natural languages such as English but in the language of an ITP. If a system generates code then this code can be run through the ITP and the system can be immediately informed if they are talking nonsense. Writing code which compiles and corresponds to a mathematical proof is much harder than writing an English language text which can be passed off as a proof, not least because proofs in an ITP must leave no stone unturned. Both Meta and OpenAI have produced work in this direction recently [PHZ$^+$22], [LLL$^+$22]. Both systems have managed to automatically generate Lean code corresponding to proofs of theorems at olympiad level (see [PHZ$^+$22], [LLL$^+$22] for more details of what has actually been achieved). One challenge, surely still at least a few years away, is the IMO Grand Challenge: to write a system which can get a gold medal in the International Mathematics Olympiad. While questions which can be solved by very smart schoolchildren are still a long way from mathematical research, it would still represent a very impressive milestone if the systems could be pushed this far. It is not at all clear to the author whether success is more likely to come from natural language systems or ITP-backed systems—or perhaps even from a hybrid approach. One issue with an ITP-based approach is that the questions will have to be translated into the language of the ITP, and it is not always clear how this should be done: for example, a question of the form "find the smallest natural number with property X" is clearly expecting an answer of the form "37" rather than one of the form "it's n, where n is the smallest natural number with property X", an answer which is logically correct but not what is intended. It is not immediately clear what to do here without leaking information which the human candidates will not have. Another issue is that training is difficult: there are solutions to all IMO problems on the internet already, so any system which has been trained on the internet has already seen all of the answers to all of the questions, and questions at the appropriate level which are not already publically available can be difficult to come by.

5. Summary

Machines have already changed mathematics by helping us to compute more quickly; this happened decades ago. But now we seem to be at the dawn of a new era, where computers are able to engage with the concept of proof. Neural networks have helped human mathematicians to discover new theorems and new counterexamples. ITPs have helped humans to simplify the proofs of recent results in the literature, and in some areas of mathematics (for example, additive combinatorics), formalizing a modern paper in a matter of months is now often possible. Large language models have currently had little effect on mathematics beyond school level, but *if* the current rate of improvement in the area continues, then they too will one day be playing a role in mathematical research.

I will end with the following observation. A lot of the research in this area is coming out of tech companies or computer science departments; indeed, in the recent past, few mathematics departments have hired in this or related areas, and so researchers are more likely to be found in computer science departments. Publications in this area are often found in journals unfamiliar to mathematicians, though things are beginning to change: DeepMind has collaborated with mathematicians from Oxford and Sydney, and courses on how to use an ITP are being taught in mathematics departments at Paris Saclay, Exeter, Fordham, Imperial College London, University College London, Carnegie–Mellon, and Harvard, and others are springing up every year. It is essential that mathematicians remain at the forefront of current developments, however, so that the area is being guided by experts and remains relevant and representative of the mathematics which is currently happening in our departments.

6. Acknowledgments

The author would like to thank Patrick Massot, Michael Harris, Siddharta Gadgil, Filippo Nuccio, Oliver Nash, Andrew Granville, and the anonymous referees for all their feedback. Any errors or omissions remaining are, of course, the fault of the author.

References

[ABH19] L. Alessandretti, A. Baronchelli, and Y.-H. He, *Machine learning meets number theory: The data science of Birch-Swinnerton-Dyer*, Machine learning in pure mathematics and theoretical physics, World Sci. Publ., Hackensack, NJ, [2023] ©2023, pp. 1–39, DOI 10.1142/9781800613706_0001. MR4619158

[BBD$^+$22] C. Blundell, L. Buesing, A. Davies, P. Veličković, and G. Williamson, *Towards combinatorial invariance for Kazhdan-Lusztig polynomials*, Represent. Theory **26** (2022), 1145–1191, DOI 10.1090/ert/624. MR4510816

[Blo21] Thomas F. Bloom, *On a density conjecture about unit fractions*, Preprint, `arXiv:2112.03726`, (2021).

[BM] Thomas F. Bloom and Bhavik Mehta, *Unit fractions*, `https://b-mehta.github.io/unit-fractions/`, April 24, 2023.

[Com] J. Commelin, *Liquid tensor experiment* (Dutch, with Dutch summary), Nieuw Arch. Wiskd. (5) **22** (2021), no. 4, 231–234. MR4477665

[CT23] Johan Commelin and Adam Topaz, *Abstraction boundaries and spec driven development in pure mathematics*, Bull. Amer. Math. Soc. (N.S.), **(61)** 2024, no. 2, ISSN: 0273-9079.

[Dah98] B. I. Dahn, *Robbins algebras are Boolean: a revision of McCune's computer-generated solution of Robbins problem*, J. Algebra **208** (1998), no. 2, 526–532, DOI 10.1006/jabr.1998.7467. MR1655464

[Dav21] Ernest Davis, *Deep learning and mathematical intuition: A review of (Davies et al. 2021)*, `arXiv:2112.04324`, (2021).

[DHL19] S. R. Dahmen, J. Hölzl, and R. Y. Lewis, *Formalizing the solution to the cap set problem*, 10th International Conference on Interactive Theorem Proving, LIPIcs. Leibniz Int. Proc. Inform., vol. 141, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019, pp. Art. No. 15, 19. MR4008934

[DJLT22] Alex Davies, András Juhász, Marc Lackenby, and Nenad Tomasev, *The signature and cusp geometry of hyperbolic knots*, Preprint, `arXiv:2111.15323`, (2022).

[EG17] J. S. Ellenberg and D. Gijswijt, *On large subsets of $\mathbb{F}_q^n$ with no three-term arithmetic progression*, Ann. of Math. (2) **185** (2017), no. 1, 339–343, DOI 10.4007/annals.2017.185.1.8. MR3583358

[Gad] Siddhartha Gadgil, *Formalizing Gardam's disproof of Kaplansky's unit conjecture*, `https://siddhartha-gadgil.github.io/automating-mathematics/posts/formalizing-gardam-disproof-kaplansky-conjecture/`, April 24, 2023.

[Gar21] G. Gardam, *A counterexample to the unit conjecture for group rings*, Ann. of Math. (2) **194** (2021), no. 3, 967–979, DOI 10.4007/annals.2021.194.3.9. MR4334981

[Hal05] T. C. Hales, *A proof of the Kepler conjecture*, Ann. of Math. (2) **162** (2005), no. 3, 1065–1185, DOI 10.4007/annals.2005.162.1065. MR2179728

[Hal14] T. C. Hales, *Mathematics in the age of the Turing machine*, Turing's legacy: developments from Turing's ideas in logic, Lect. Notes Log., vol. 42, Assoc. Symbol. Logic, La Jolla, CA, 2014, pp. 253–298. MR3497663

[He21] Y.-H. He, *The Calabi-Yau landscape—from geometry, to physics, to machine learning*, Lecture Notes in Mathematics, vol. 2293, Springer, Cham, [2021] ©2021, DOI 10.1007/978-3-030-77562-9. MR4301304

[Lac] Marc Lackenby, *Using machine learning to formulate mathematical conjectures*, `https://www.youtube.com/watch?v=0ekP5M7w3dQ`, April 24, 2023.

[Leg08] A.-M. Legendre, *Essai sur la théorie des nombres* (French), Cambridge Library Collection, Cambridge University Press, Cambridge, 2009. Reprint of the second (1808) edition, DOI 10.1017/CBO9780511693199. MR2859036

[LLL+22] Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet, *Hypertree proof search for neural theorem proving*, Advances in Neural Information Processing Systems (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, Curran Associates, Inc., 2022, pp. 26337–26349.

[LMF23] The LMFDB Collaboration, *The L-functions and modular forms database*, `https://www.lmfdb.org`, 2023, April 24, 2023.

[Man] Allen Mann, *A complete proof of the Robbins conjecture*, `http://math.colgate.edu/~amann/MA/robbins_complete.pdf`, April 24, 2023.

[Mas] Patrick Massot, *Formal mathematics for mathematicians and mathematics students*, `https://www.youtube.com/watch?v=tp_h3vzk0bo`, April 24, 2023.

[MvDN22] Patrick Massot, Floris van Doorn, and Oliver Nash, *Formalising the h-principle and sphere eversion*, CPP 2023: Proceeding of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, January 2023, pp. 121–134.

[Nap89] John Napier, *The construction of the wonderful canon of logarithms*, W. Blackwood and sons, 1889.

[Ope] OpenAI, *Chatgpt*, `https://chat.openai.com`, April 24, 2023.

[Ope23] OpenAI, *Gpt-4 technical report*, 2023.

[PHZ+22] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever, *Formal mathematics statement curriculum learning*, 2022.

[PS12] J. D. Phillips and D. Stanovský, *Bruck loops with abelian inner mapping groups*, Comm. Algebra **40** (2012), no. 7, 2449–2454, DOI 10.1080/00927872.2011.579587. MR2948838

[Sch22] P. Scholze, *Liquid tensor experiment*, Exp. Math. **31** (2022), no. 2, 349–354, DOI 10.1080/10586458.2021.1926016. MR4458116

[Sta18] The Stacks Project Authors, *Stacks Project*, `https://stacks.math.columbia.edu`, 2018.

[Wag] Adam Wagner, *Finding counterexamples to conjectures via reinforcement learning*, `https://www.youtube.com/watch?v=vMLVH6IEwlM`, April 24, 2023.

[Wag21] Adam Zsolt Wagner, *Constructions in combinatorics via neural networks*, 2021.

Department of Mathematics, Imperial College London, London SW7 2AZ, United Kingdom

Email address: `k.buzzard@imperial.ac.uk`