How to Do Maths With Words

Neural Machine Learning Applications to Mathematics
and Their Philosophical Significance

Juan Luis Gastaldi

Abstract

Recent years have seen a remarkable development of deep neural network techniques for data analysis, along with their increasing application in scientific research across different disciplines. The field of mathematics has not been exempted from this general trend. The present paper provides a survey of recent applications of neural models to mathematics and assesses their philosophical significance concerning the role of language in mathematics.

1 Introduction

On the 15th of February, 2023, Yuhuai (Tony) Wu, researcher in Artificial Intelligence at Google at that time, gave a talk at the UCLA Institute for Pure & Applied Mathematics (IPAM).[1] In his talk, Wu presented his most recent results in the automatic formalization of proofs through deep neural networks. The workshop was organized by a small and highly select committee, including two Field medalists, and the audience was composed of well-established world specialists in the field of automated theorem-proving. While explaining the details of the neural model in question and providing elements for its epistemological guarantees to this intrigued yet somewhat skeptical public, Wu affirmed: "We know for sure that the informal proof, the natural language proof, is correct because it is produced by a human". The statement was followed by a second of sincere bewilderment in the room, only resolved by one glorious collective laugh. Unable to hide his confused embarrassment, Wu tried, however, to reassure the crowd by adding: "Ok, so, we make sure that the human... we scrape the solutions from the Internet!", inevitably provoking a new burst of laughter, this time sounding like a redemption.

Wu's statement is not improvised or misguided, nonetheless. Let alone isolated. For a proof, take the official presentation of the second version of Google's Large

To be published in *Handbook of the History and Philosophy of Mathematical Practice*. Ed. by Bharath Sriraman. Springer, 2024.
This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 839730.
File version: 0.0.2023-09-07.

[1]Wu's talk was part of a Workshop on Machine Assisted Proofs held between the 13th and the 17th of February, 2023. The recording of the talk is available on the website of the IPAM.

Language Model PaLM (Ghahramani, 2023), released shortly after Wu's talk. In it, one can read the following passage, flagged as "weird" by the renowned researcher at the Santa Fe Institute, Melanie Mitchell:

> PaLM 2's wide-ranging dataset includes scientific papers and web pages that contain mathematical expressions. As a result, it demonstrates improved capabilities in logic, common sense reasoning, and mathematics. (Ghahramani, 2023)

The rapid proliferation of this peculiar kind of statement in the current state of research at the crossroads of mathematics and computing is not as much anecdotal as it is *symptomatic*. As is symptomatic the perplexed reaction of well-established specialists in the fields concerned. Both are the symptom of the surge of a perspective on mathematical knowledge and practices radically at odds with the one that governed our image of mathematics for more than a century. This circumstance is the effect of an ongoing revolution in the field of machine learning resulting from the remarkable development of Deep Neural Network models (DNNs) during the last two decades. As it has been widely observed, the fact that tech corporations are the main actors of this technological revolution entails that most research efforts associated with this process are inseparable from the development of market products and the increase of profit under the brand of "Artificial Intelligence" (AI). However, the past years have also witnessed the application of the new neural methods to various aspects of scientific research across the disciplinary spectrum, from natural to social sciences. Significantly, mathematics has not been the exception to this general trend. Indeed, since the mid-2010s, many efforts have been dedicated to treating numerous aspects of mathematical knowledge using DNNs of various kinds.

Mathematics has been in the sights of AI champions since the very emergence of the AI program. In his pioneering 1948 paper on "Intelligent Machinery", for instance, Turing (2004) already considered mathematics a privileged field for the application of "thinking machines". And the celebrated 1955 Dartmouth project proposal included a supplement by Newel and Simon with a plan to extend to "learning, self-programming and the self-selection of theorems to be proved (innovation)" their work on "Mathematics Machines" that "discover proofs in the propositional calculus [...] by employing search procedures that appear at least grossly similar to those used by humans in dealing with the same problems" (Newell and Simon, 1956). At any rate, considering the conception of intelligence motivating the AI program, one should not be surprised that mathematics falls within its scope.

However, given the nature of recent neural techniques and the unexpected results they can afford, as natural as it may be for the AI ideology to address mathematical tasks, the very possibility of their successful application to mathematical knowledge cannot but come as a surprise from a conceptual and philosophical perspective. Hence, faced with the results of the renewed AI momentum, it seems inevitable for well-established conceptual positions to return to their foundations and reopen old questions: Wasn't the whole idea of proof assistants and automated theorem provers to *prevent* the frailty of human proving practices instead of *reproducing* them and *relying* on them as a gold standard? Wasn't mathematical competence a *condition* to write and read scientific papers and mathematical expressions rather than their miraculous *effect*? Wasn't natural language the *cause* of rather than the *solution* to the multiple problems preventing mathematics from achieving higher degrees of precision? And more generally, didn't the *formal* nature of mathematics, finally conquered by the logical regulation of its symbolic means, make it impassive to the strong *empirical* position assumed by connectionist approaches guiding the application of DNNs?

In the current state of the art, it is not possible to determine who will have the last laugh. Yet, whatever the outcome, the rapid spread of deep learning techniques will likely have, for better or worse, a significant impact on future mathematical practices. For this reason, the current situation represents a most exciting moment in the eyes of the historian and philosopher of sciences. A moment where established certainties can and need to be regarded afresh to assess if they hold still in a context that suddenly is no longer the same. Hence, even as the symptom of a bewildered state, the philosopher can only rejoice at the reopening of those fundamental epistemological questions, hoping that a philosophical treatment of the latter can contribute to critically assessing the stakes of the new situation and indicating novel orientations.

The present paper intends to make a first step in this direction. As such, its primary purpose is to offer a survey of the many recent applications of neural machine learning techniques to different aspects of mathematical knowledge, calling attention to what can be of significance for a philosophy of mathematics. It will appear that, disregarding traditional boundaries in the mathematical landscape, the orientations in this emerging research area tend to be distributed following different mathematical *practices*. Yet, despite their apparent dispersion across those practices, all applications share, if only unwittingly, a common philosophical assumption. It is this assumption that the statements opening these pages announced. Namely, the idea that whatever we do when we do mathematics, we do it with *words*. More precisely, the analysis of different mathematical applications of DNNs shows that their potential success should force us to revise our conception of the relation mathematics maintains with language, and understand that and how *natural language, in its most immediate and unaffected expression, constitutes a decisive component of mathematical knowledge*.

Accordingly, the plan of the chapter is as follows. In the next Section (2), we provide a brief presentation of deep neural network models. Section 3 constitutes the principal contribution of this paper, offering a survey of recent applications of DNN models to mathematical tasks. The section is organized into four parts, featuring four distinct lines of research identifiable in the field, according to the task assumed by researchers to characterize mathematical practice: proving theorems, manipulating objects, acquiring skills, or addressing open problems. Finally, in Section 4, we address the philosophical significance of those results, focusing on the relation between mathematics and natural language and the challenges for a theory of language where the semantic aspects of mathematical knowledge can be inferred from pure syntax. We conclude in a programmatic tone, indicating the perspectives open to a language-driven philosophy of mathematical practice.

2 What are Deep Neural Nets?

A detailed presentation of DNNs falls outside the scope of these pages. Our account of DNNs will thus remain deliberately simplistic, offering the elementary features needed to grasp the main characteristics of their application to mathematical tasks in the following section. Fortunately, owing to the growing popularity of AI applications in recent years, a wide range of presentations are now readily accessible to the interested reader, featuring different levels of granularity and difficulty. In particular, for a systematic and detailed presentation, we refer the reader to Goodfellow et al. (2016) or Brunton and Kutz (2022). Also, Williamson (2023) does an excellent job at providing a rigorous yet brief and accessible introduction to neural nets addressed to the uninformed working mathematician.

From a bird's-eye view, artificial neural networks are *parametric functions between two vector spaces*. Therefore, their application to any field relies on the possibility of encoding some information as a vector (i.e., a list of numbers), feeding the latter into the function as an argument (or the "model" 's input), and retrieving a target vector as output, to be decoded as some other kind of information. The nature of the information is assumed to be of little significance as long as it can be encoded as a vector and a vast set of examples of pairs of input and target vectors is available to train the model. Indeed, neural network architectures have been proven to be universal function approximators (Hornik et al., 1989). Therefore, given enough examples of input and target vectors, a neural net will approach a function transforming one into the other, hopefully in a way that performs as expected for arguments outside the training data.[2] To encourage this "out-of-distribution generalization", neural models are typically trained on only a fraction of the available data (typically around 90%) and tested on the remaining held-out part.

In the most elementary cases, the internal structure of the model is composed of a series of transformations, each having a similar configuration: a linear transformation (i.e., a matrix), a bias (an added vector), and an activation (non-linear) function. More precisely, a DNN can be described as a function $f : \mathbb{R}^n \to \mathbb{R}^m$ explicitly expressed as a composition:

$$f\colon \mathbb{R}^n \xrightarrow{f_1} \mathbb{R}^{n_1} \xrightarrow{f_2} \cdots \xrightarrow{f_K} \mathbb{R}^{n_K} \xrightarrow{g} \mathbb{R}^m \tag{1}$$

with

$$f_i(\mathbf{x}_{i-1}) = a(\mathbf{M}_i\mathbf{x}_{i-1} + b_i) \tag{2}$$

where $\mathbf{x}_i = f_i(\mathbf{x}_{i-1})$ and $\mathbf{x}_0$ is the input vector; the $\mathbf{M}_i$ are $n_i \times n_{i-1}$ matrices representing the linear functions; the $b_i \in \mathbb{R}^{n_i}$ are the biases; the a is the non-linear activation function; and g is the output function.

Each one of these complex transformations (i.e., each f_i) constitutes a *layer*. Hence, when a vector is fed into a layer, it is multiplied by the matrix, the bias vector is added, and the non-linear function is applied element-wise. The *deep* character of neural nets is determined by the composition of many of these layers so that the output of one layer becomes the input of the next one until a final output vector is produced.

Finally, the model is *trained* by iteratively updating the components of the matrices and bias vectors (or *weights*) of the different layers. This is achieved by performing a gradient descent on a *loss function*, that is, on a function that measures the error between the vector output by the model and the target vector provided by the training set. Through an algorithm known as backpropagation, the weights of different layers are progressively adjusted so that the average error over the set of examples hopefully converges to some minimum value.

There exist many training strategies. In *supervised learning*, the target vectors are handcrafted to encode a property of the data encoded in the input vector. In *self-supervised learning* (sometimes mistaken for unsupervised learning), the target vector is some part of the input vector masked when the latter is fed to the model. Finally, in

[2]This does not mean that any task is solvable by means of neural models, as a significant part of the AI community tends to think. Many problems, maybe most of them, do not accept being constructed as a predictive task based on a finite number of supposedly similar cases expressible as vectors. Problems in politics, law, art, and many other domains of social life cannot, by their nature, be framed in such terms without transforming, if not wholly obliterating that very nature.

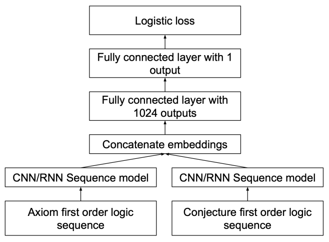
reinforcement learning, each output by the model receives a score through a carefully designed reward function, and the model's objective is to maximize that score.

As we said, the scheme presented here corresponds to the most elementary neural models. In recent years, DNNs have become increasingly complex and diversified, from convolutional neural nets (CNNs) and recurrent neural nets (RNNs, LSTMs) to transformers and diffusion models, to name only the most popular architectures. Those architectures remain particularly sophisticated cases of the basic model presented above. Due to their significance in the current state of the art, the case of *transformers* deserves special mention. Introduced by Vaswani et al. in 2017 and popularized by models like BERT (Devlin et al., 2018) and GPT-3 (Brown et al., 2020), the transformer architecture is characterized by the implementation of an *attention* mechanism. Simply put, the idea is to enhance each layer f_i of the net with a set of vectors capable of determining, for each component of the input vector $\mathbf{x}_{i-1}$, which other components of that vector are relevant to compute the correct output vector (i.e., which other components the component in question "attends" to). Like the linear components of the architecture, attention vectors are randomly initialized and jointly trained with the rest of the model. Among other technical advantages, the attention mechanism allowed the exploitation of parallelism during training, significantly reducing the training time and paving the way for a massive increase in scale, both of model size (parameters) and training data through self-supervised learning. Since their introduction, transformers have exhibited surprising capabilities across the most diverse tasks and domains and have been widely adopted, defining the state of the art at the moment of writing these pages.

3 How to Do Maths With Neural Nets

Applying DNN models to the processing of mathematical knowledge is not a straightforward and unambiguously defined task. As is usually the case with current AI applications across domains, craftsmanship in computational models is sometimes more prominent than proficiency in the objects those models are expected to analyze. Accordingly, the different DNN applications to mathematics are less determined by a refined elaboration on the nature of mathematical objects and knowledge than by the AI researcher's implicit assumptions as to what characterizes a mathematical task. Given the current exploratory state of a field that evolves at a remarkable speed, the answers to this unraised question are multiple, with new results and sometimes original perspectives released on a monthly basis. Any ambition to provide an exhaustive account of the current state of the art is, therefore, doomed to fail from the start. Hence, we will content ourselves with presenting what we recognize as the main trends underlying these developments. Significantly, those trends are informed by their implicit understanding of what it is that we do when we do mathematics. From this perspective, it is possible to identify four main trends in the field, depending on whether the research is oriented toward finding *proofs*, manipulating *objects*, acquiring *skills*, or supporting *heuristics*. The following sections present each one of these orientations, providing some details of what we identify as their respective seminal papers, and briefly accounting for a few landmark results and developments.[3]

[3]The reader should keep in mind that, despite our efforts to cover as much of the field as possible, the following constitutes only a partial view on the state of the art, biased toward works exhibiting philosophical interest. A more precise perspective can be obtained by inspecting the "Related Work" section usually included in the papers we will present.


Logistic loss
Fully connected layer with 1 output
Fully connected layer with 1024 outputs
Concatenate embeddings
CNN/RNN Sequence model
CNN/RNN Sequence model
Axiom first order logic sequence
Conjecture first order logic sequence

```
::  t99_jordan:  Jordan curve theorem in Mizar
for C being Simple_closed_curve holds C is Jordan;

::  Translation to first order logic
fof(t99_jordan, axiom, (![A] : (
(v1_topreal2(A) & m1_subset_1(A,
k1_zfmisc_1(u1_struct_0(k15_euclid(2))))) =>
v1_jordan1(A)) ) ).
```

Figure 1: Network structure (left) and input example (right) for the premise selection task in *DeepMath* Alemi et al. (2016).

3.1 Proof-Oriented

What can certainly be taken as the first significant attempt to apply by then well-established DNN models to the solution of mathematical tasks dates back to 2016, with the *DeepMaths* paper by Alemi et al. (2016). This work pioneered a prolific line of research oriented by the implicit assumption that theorems are the key to mathematical knowledge and theorem proving constitutes, therefore, the heart of mathematical practice. Accordingly, from this perspective, the main goal is to train DNN models to find *proofs*. Ideally, one would train a neural net on an extensive corpus of existing proofs, after which one could feed a yet unproved mathematical statement (a "conjecture") to the model, which would output a proof, if any. Notice that this is not the same as constructing a logical framework for automatic theorem proving because there is no definition of an explicit logical system in this case. One could imagine logical systems to be just a specific kind of function—even if a partial one—mapping the domain of statements onto that of proofs, and we are asking the neural model to approximate such a function.

Researchers within this trend tend to think that solving this task is practically sufficient to automate mathematical practice altogether. Indeed, in a paper significantly entitled *Towards the Automatic Mathematician*, Rabe and Szegedy (2021) affirm: "Ideally, we could simply talk to an automatic mathematician like a colleague, and it would be able to contribute to mathematical research, for example by publishing papers without human support." While the authors vaguely mention other tasks such as "formulate and explore its [the automatic mathematician's] own theories and conjectures", theorem proving is acknowledged as "an important instrument of our plan" and, indeed, the only strictly mathematical task considered in their manifesto.

In practice, however, things are more subtle. Actual research in this direction has focused chiefly on specific subtasks identified within the framework of existing interactive or automated theorem provers (ITP/ATP), such as E, HOL Light, Mizar, Metamath, Isabelle, or Lean. The focus is mainly put on those tasks that represent computational bottlenecks. In particular, there has been intensive research on *premise selection*, namely "the selection of a limited number of most relevant facts for proving a new conjecture" (Alemi et al., 2016, p. 1). This task was the one motivating Alemi et al.'s (2016) pioneering work, where it was implemented as establishing the pairwise relevance between conjecture and axioms by learning to predict the usefulness of given axioms in the proof of given conjectures. The overall strategy consists in feeding both a conjecture and an axiom into the model, processing the input vectors independently through different network architectures producing intermediate vector representations of various kinds, concatenating those vectors, passing them to a fully connected DNN, and outputting a probability measure of the usefulness of the axiom for proving the conjecture (see Fig. 1). The dataset used was the Mizar Mathematical Library, a library of almost 60K theorems proved in Mizar, together with unnamed top-level lemmas. The input axioms and conjectures were then the strings (i.e., sequences of digital characters) representing the first-order logic formalization extracted from the Mizar Library (see Fig. 1 for an example). The authors tested two ways of feeding these texts into the model: character by character and word by word (token by token). In the latter case, tokens recognized as identifiers were mapped onto their already processed definition. Other than that, the expressions were fed sequentially with no parsing information. For each conjecture, the model was trained to produce a probability distribution over the set of possible premises. To evaluate the model, the authors assessed the number of conjectures that could be proved from the top-k premises in the corresponding probability distributions (for different values of k) using the E automatic prover. They also evaluated the ranking quality, measuring if relevant premises appeared at the top of the rank. Such measures were then used to compare the different DNN architectures.

The results obtained by the best of those early models exhibited a slight increase in performance compared to existing benchmarks. Although modest, the outcome was far from wholly unreasonable, showing neural models could compete with more classically engineered techniques and handle mathematical proofs to some extent. This work sparked then a series of other studies in the same direction, where the integration of neural techniques within ITPs and ATPs can be seen to increase. For example, Loos et al. (2017) provided evidence that integrating neural techniques for premise selection in the proof search procedure of the E ATP results in an increased amount of theorems proved under time and memory constraints. On the training side, Kaliszyk et al. (2017) proposed a new dataset for higher-order theorem proving, comprising over 2 million training examples of proof steps extracted from almost 11,400 proofs formalized by humans in the HOL Light ITP. The authors also provide a series of benchmarks for different machine learning models over that dataset. HOL Light was also used by Bansal et al. (2019) to propose a different training strategy, adding to the usual supervised learning on human proofs a reinforcement learning step. Instead of being trained to predict the correct premise, the model interacts as an "artificial agent" with the ITP, proposing both a tactic to be applied and a rank over the set of possible premises. The success or failure of the ITP to apply the received tactics on the premises is then used as a reinforcement signal to train the model to predict pairs of tactics and premises.

With the evolution of this line of work, researchers started considering a wider range of theorem proving subtasks. Kaliszyk et al. (2017), for example, mention, among others: "Predicting whether a statement is useful in the proof of a given conjecture", "Predicting whether a statement is an important one (human named)", "Predicting the name given to a statement", or "Generating the conjecture the current proof will lead to" (p. 4). Bansal et al. (2019) add some others to that list, such as predicting the same tactics as the human proofs or predicting goals and subgoals. Also, *tactics prediction* has gained increasing attention due to the significance of ITPs and ATPs in the concrete developments of this research orientation (Bansal et al., 2019; Paliwal et al., 2019; Lee et al., 2020). More generally, recent years have seen a proliferation of datasets, models, and benchmarks, which contribute to the expansion of this perspective.

The arrival of the *transformers* architecture (Vaswani et al., 2017) brought about new developments in neural theorem proving. As we saw, transformers are designed to deal with sequential data in a self-supervised and highly scalable way. Their success in the field of AI is inseparable from the surprising capabilities exhibited in the processing of natural language, motivating the emergence of so-called *Large Language Models* (LLMs). We will have the opportunity to say more about LLMs in the *Skill-Oriented* section below. For now, it is sufficient to note that, encouraged by their linguistic capabilities, researchers have explored multiple ways to utilize transformer-based models for neural theorem proving.

A noteworthy case is *GPT-f* (Polu and Sutskever, 2020), an automated prover and proof assistant for the Metamath language, which was capable of significantly outperforming the state of the art in neural theorem proving at the moment of its publication. *GPT-f* is based on a model architecture similar to GPT-2 and GPT-3, the celebrated LLMs by OpenAI. It is designed to perform backward proof search, starting from a root goal and exploring tactics possibly leading to such a goal. The process may involve identifying other subgoals for which new tactics are explored iteratively (with a pre-specified bound on the iterations) while keeping track of the tree structure. The neural model is therefore trained to predict a “proofstep” (i.e., a tactic and the corresponding subgoals) for a given goal, both encoded as strings, which are then passed on to the Metamath kernel for formal verification. The use of transformers has the advantage of unifying the neural architecture, as opposed to using complex combinations of different architectures as in previous models. It also allowed for pretraining as a language model over various kinds of corpora (cf. LLMs in the *Skill-Oriented* section), including the mathematical fragments of *arXiv* and *StackExchange* as well as computer code from *GitHub*. Other than achieving a new state of the art for Metmath (56.22% closed proofs in the test set vs. 21.16% for the best existing models at the time), the authors could show that, within this framework, both language modeling pretraining (especially on mathematical corpora) and increased model size result in improved performance. Interestingly, *GPT-f* could generate 23 shortened proofs of existing statements in the Metamath library, reported as enthusiastically received by members of the Metamath community. For the authors, this was “the first effective contribution of a deep learning system to a formal mathematics library” (Polu and Sutskever, 2020, p. 14).

One of the drawbacks of transformer architectures is that their high performance is conditioned to training over a massive amount of data, orders of magnitude larger than existing hand-crafted mathematical libraries. A common strategy has been to augment the latter with synthetic data generated in various ways out of known mathematical properties and algorithms (cf., for instance, Wang and Deng, 2020). However, Polu and Sutskever (2020) showed that scaling the portion of synthetic data up to 5% of the training data actually hurt the performance of their model. While the authors did not inquire into the reasons for this empirical behavior, they proposed an alternative way to circumvent the scarcity of proof data by training a “value function” helping to guide the proof search more efficiently. Essentially, such a function predicts the probability that a subgoal is actually used in the final proof. The function is trained on a synthetic dataset produced by sampling proofs *from the trained neural model itself*, keeping *only those that the formal verifier can prove*. Their experiments show that the iterative use of this procedure, also known as *expert iteration* leads to significantly better results, suggesting a strategy of “continuous self improvement”. Following this line of research, Lample et al. (2022) showed that replacing this iterative process with an *online training* over proofs generated by the model and verified by an ITP results in substantial performance gains.

In recent years, the success of DNN models—and transformers in particular—in the processing of natural language motivated what can be conceived as a subtle but notable reorientation of the research efforts within proof-oriented approaches toward what came to be known as *autoformalization*. In a highly programmatic paper, Szegedy (2020) defines an autoformalization system as "an automated system that is capable of automatically formalizing significant portions of mathematics from a natural language input and verifying it automatically" (p. 4). From this perspective, formalizing essentially means expressing statements as computer programs susceptible to be verified computationally following transformation rules. Implicit assumptions concerning the nature of mathematics within this body of work become here explicit: "Mathematical reasoning is just reasoning about anything specified formally" (p. 4). Yet, this recent turn pushes these assumptions further and presents autoformalization as a natural step toward AI models endowed with "general reasoning" capabilities. In this way, mathematical practice, which had already been reduced to formalizing practices, is spontaneously raised to the level of "the discipline of pure reasoning": "Mathematical reasoning is not about mathematics per se, it is about reasoning in general" (p. 3). The distance between thought and mathematical thought becomes thus so misty it is then "natural to ask: Will we ever arrive at the point where an AI agent can learn to do reasoning as well as the best humans in the world in most established domains of mathematics." (p. 4)

However, all these prejudices about mathematics, reasoning in general, and the best humans in the world should not divert our attention away from an essential fact: This somewhat unexpected emergence of informal mathematics through natural language at the heart of the most extreme attempts to fully automate theorem proving looks more like an admission of weakness than a *tour de force*. Indeed, when one examines the recent evolution of DNN models, it is easy to identify a subtle glide away from highly measurable functions toward more informal tasks, such as text and image generation. For the latter, unforeseen capabilities can be achieved across multiple fields, including mathematics and programming. Yet, the counterpart of that originality is a loss of robustness and reliability, all the more fundamental that failure is intrinsically more difficult to assess in informal settings (how can a poem or a picture be "incorrect"?). The very nature of this success, *informal by design*, together with the relative scarcity of formal proof corpora, left researchers in the area little choice other than focusing on bridging a gap secretly undesirable for most of them between formal methods on one side and neural linguistic models on the other.

Besides explicit or implicit motivations, recent explorations of natural language expressions of mathematical statements such as Ferreira and Freitas (2020, 2021) and Welleck et al. (2021), and autoformalization tasks in particular, are of interest in their own right. A noteworthy work concerning the latter is that of Wu et al. (2022), exploiting the "few-shot learning capabilities of LLMs" to produce translations into Isabelle code of natural language mathematical statements and problems contained in the MATH (Hendrycks et al., 2021b) and MiniF2F (Zheng et al., 2022) mathematical competition datasets. The relative success of this approach is nonetheless remarkable considering the negligible amount of aligned pairs of informal-formal expressions expected to be present in the vast training corpus of the LLMs. Autoformalization faces, however, the challenge of articulating the high-level structure of informal proofs with the low-level details required for automated proving. To address this difficulty, Jiang et al. (2023) proposed a method where an initial informal statement is associated with an informal proof *draft* (either through available data or through automatic generation by an LLM), which is, in turn, broken down by an LLM into a formal proof *sketch*, i.e., a high-level structure of formal statements, to be then passed on to an ATP. The experiments on the miniF2F dataset suggested that this strategy is indeed capable of successfully guiding the ATP and boosting its proving capabilities, with comparable results for the cases where the informal drafts were produced by humans or LLMs.

Considering the development of this research orientation from its inception up to the current state in 2023, one thing seems clear: Despite a declared inclination toward end-to-end black-boxed models inherited from AI perspectives and coated with multiple ideals of intelligent machines and dispensable humans, DNN applications to theorem proving have actually become increasingly reliant on existing (non-neural) interactive or automated theorem provers. To the extent that, as of today, it is hard to imagine how the latter could be dispensed with. And yet, against all odds, neural techniques have proved to contribute to the performance of formal theorem proving within those frameworks. So much so that a hybrid strategy, integrating neural techniques and I/ATPs, appears today as a promising research program for the years to come. In this sense, Jiang et al. (2022) proposed a framework for neural models to interact with any hammer-enabled ITPs, showing that neither the neural model nor the ITP alone can match the performance achieved by their careful integration. However modest and far from its promises the contribution of neural nets may be, the latter have at least succeeded in making them through means that were entirely out of the radar of traditional formal methods. From a philosophical standpoint, the leveraging of informal expressions resulting from various mathematical practices, and the reliance on natural language in particular, are distinctive features of this approach, to which we will have the chance to return. For now, let us say that, despite the significant challenges and the ideological noise surrounding AI applications to theorem proving, DNNs have shown that they can do *something* with mathematical proofs.

3.2 Object-Oriented

A second research orientation for applying DNN methods to mathematics shares a similar use of the learning models with the first. However, the focus is put on manipulating mathematical *objects* rather than statements. Admittedly, proofs can also be understood as mathematical objects, in which case proof-oriented approaches could also fall under this category. At any rate, as learning models, DNNs are, in principle, indifferent to this distinction. Yet, both from a philosophical and a formal standpoint, there is arguably a significant difference between, say, manipulating numbers to perform arithmetical operations and proving a general arithmetical property for a possibly infinite set of numbers.

In line with the general neural models' general setting, the goal in object-oriented mathematical applications is to feed the expression of a mathematical object (e.g., equation, terms of a series, etc.) into the model and expect the latter to output some specific result or property associated with it (e.g., solution, recurrence relation, etc.).

The most elementary case of this strategy is given by arithmetical operations and their corresponding results. Early assessments of deep neural models already showed that recurrent architectures (RNNs) could be trained to perform integer addition and multiplication based on binary representations with strong generalization capacities (Łukasz Kaiser and Sutskever, 2016). Those results were extended to include decimal multiplication through a binary encoding of each decimal digit (Freivalds and Liepins, 2017). Notice that, in all cases, from the model's perspective, input numbers are simply *tokens*, so there is no reason a priori that their numerical content is preserved when represented as the usual sequence of digits. Hence the binary representation preferred by these early approaches. Indeed, Trask et al. (2018) showed that other numerical representations exhibit serious difficulties generalizing outside the range of values seen in training and proposed to enhance neural models with a specific architecture designed

Encoding	3.14	$-6.02.10^{23}$	Tokens / coefficient	Size of vocabulary
P10	`[+, 3, 1, 4, E-2]`	`[-, 6, 0, 2, E21]`	5	210
P1000	`[+, 314, E-2]`	`[-, 602, E21]`	3	1100
B1999	`[314, E-2]`	`[-602, E21]`	2	2000
FP15	`[FP314/-2]`	`[FP-602/21]`	1	30000

Table 1: Examples of different encodings for numerical coefficients in Charton (2021).

to incorporate an inductive bias for numerical computation.

More recently, researchers have explored the capabilities of DNNs to perform numerical calculations on more complex mathematical objects. A landmark in this sense is the work by Charton (2021), training transformers to compute solutions to linear algebra problems, including transposition, addition, multiplication, and inversion of matrices, as well as eigenvalues, eigenvectors and singular value decomposition (SVD). His results showed the models can indeed compute approximate solutions with high accuracy, generalizing out of their training distribution under certain conditions and even extrapolating to larger numbers. Unlike most of the work presented in the previous section, models were here trained on synthetic corpora specially tailored for the intended tasks. This work is emblematic of an object-oriented approach; therefore it's worth examining it more closely.

The input and output in Charton (2021) are, then, matrices of real numbers. However, their representation was carefully designed to fit the kind of data on which transformers are known to exhibit high performance, namely, sequential data. Matrices were thus encoded as sequences of coefficients prefixed with the matrix dimensions. Remember that each term in a sequence is simply a token. For this reason, numerical coefficients were also encoded following very precise rules. Based on the general principle that real numbers can be represented by a triplet of a sign, a mantissa, and an exponent, four alternative encodings were proposed for that purpose: base 10 and 1000 encoding of the mantissa, signed base 1000, and floating point (see Table 1 for Charton's (2021) original examples). Datasets were produced by online sampling dense random matrices, from 5×5 up to 15×15 dimensions, with coefficients sampled uniformly within the interval $(-10, 10)$. The data was used to train a transformer for the various tasks studied. For all tasks, an output sequence of tokens was considered a correct solution if it could be decoded as a matrix and represented a correct solution within a predetermined numerical tolerance.

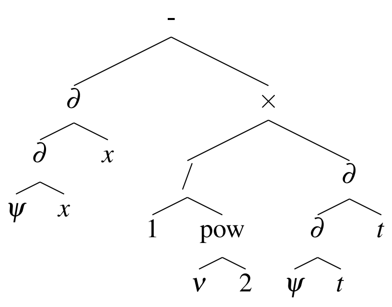
Under those conditions, the study showed that DNNs can perform matrix transposition —which does only involve permutations of tokens in the sequence, without arithmetic operations—with practically perfect accuracy. This is also the case with matrix addition up to 10×10 dimensions and 1% tolerance, with a slight decrease in performance for matrices of larger sizes. The model's accuracy dropped noticeably when trained over matrices of different sizes. Although more complex, matrix multiplication presented very similar results to addition, but only for the first two numerical encodings. The study also exhibited surprisingly high performance in the much more complex task of eigenvalue computation. The correct prediction of eigenvectors remained restricted to matrices of small sizes. However, the accuracy increases with the numerical tolerance, indicating that *errors are not random but somewhat akin to rough approximations*. Matrix inversion proved to be hard for DNNs, with no more than 80% accuracy at 5% tolerance. Finally, while achieving high performance on 4×4 matrices, SVD could not be predicted for matrices of 5×5 or more dimensions.

The fact that the properties of the objects under analysis are well-known contributed to providing many principles of interpretability for these results, both for the success and the failure cases (Charton, 2021, 2022). In this way, Charton could show that, for eigenvectors, even when the predictions were incorrect, the model correctly learned some underlying properties, such as their unit norm and their orthogonality. Also, a fine-grain analysis of the metric used in matrix inversion could show that failures were to be attributed to the computation of matrix inverses rather than the neural model itself. The study also provided evidence against the possibility that the results depend on the model simply memorizing the data and could point to strong predictors of success or failure for some of the most challenging tasks. Overall, this work showed that the success of DNNs in performing numerical calculations on matrix operations is highly dependent on the encoding (the best being P1000 and FP15), with greater difficulty for increasing matrix size and the capacity of generalizing out of training distribution if the latter is carefully chosen.

The solution of partial differential equations through DNN techniques has also been the object of much research effort. Blechschmidt and Ernst (2021) provide a survey of approaches to numerical solutions. However, Lample and Charton (2020) introduced an original perspective, testing the DNNs' capabilities to solve differential equations and integration *symbolically*. The overall strategy of this seminal paper was to feed a function or a first or second-order differential equation as input and require the model to output its integral or its solution, respectively. Here, too, we find that the training corpora were carefully designed for the intended tasks. In particular, mathematical expressions of functions and differential equations were represented as binary trees, where the internal nodes represent operators, while the leaves can represent either numbers, constants, or variables. The trees were then sequentialized through prefix (i.e., Polish) notation by enumerating nodes in prefix order and then encoded as a vector to be fed into the model (see Fig. 2 for an example). As the authors observe, using trees to represent expressions also provides a principled way to tackle the otherwise thorny problem of sampling from an expression space. Training and test datasets were carefully constructed using Mathematica to produce pairs of equations and solutions, which were then turned into the chosen representation and carefully cleaned up to eliminate any possible source of noise. The model used was an early version of a transformer, considered by the authors as a sequential model aimed at natural language translation.

Relying on all this careful handcrafting of the mathematical object's representations, the authors trained models on five tasks, each one relying on a dedicated synthetic dataset: integration of randomly generated functions solvable by a computer algebra system; integration of derivatives of randomly generated functions; integration of the composition of one random function and the derivative of another random function (i.e., integration by parts); first-order differential equations; and second-order differential equations. Under the precise conditions established by the authors, the corresponding models exhibited surprisingly high scores across those tasks, with close to 100% accuracy for the different kinds of integration and around 80% and 40% for ODEs of orders 1 and 2, respectively. The latter could be brought up to around 94% and 73% when considering if at least one of the top 10 hypotheses output by the model was correct (and 97% and 81% for the top 50). These results outperformed those of computer algebraic systems such as Mathematica (under runtime constraints), Matlab, and Maple. Generalization outside the training distribution proved challenging. However, the authors showed that the reason was a systematic bias toward specific lengths for the expressions of problems and solutions in different datasets. Training on mixed datasets

$$\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{v^2}\frac{\partial^2 \psi}{\partial t^2}$$



$$[\ -\ \partial\ \partial\ \psi\ x\ x\ \times\ /\ 1\ \mathrm{pow}\ v\ 2\ \partial\ \partial\ \psi\ t\ t\ ]$$

Figure 2: Example of an original mathematical expression (top left), its tree representation (top right), and the latter's sequentialized version (bottom) in Lample and Charton (2020).

substantially increased the performance on problems outside the training distribution.

While neural frameworks had been shown to successfully process symbolic properties of mathematical propositions and numerical properties of mathematical objects—at least to some extent—, the relevance of Lample and Charton's (2020) work lies in the capacity of bridging the gap between both orientations by operating on objects through their symbolic expression. The application of DNNs to *symbolic regression* provides another significant confirmation of this potential. Symbolic regression is the task of finding the symbolic expression of a function given some of its values. Before the emergence of current DNN models, symbolic regression methods relied on evolutionary algorithms. Early applications of deep neural techniques to this task include Martius and Lampert (2016); Udrescu and Tegmark (2019); Kim et al. (2019); Petersen and Landajuela (2019), where different kinds of DNNs are used to improve the exploration of the space of possible mathematical expressions, in search of the one that best fits the set of given values. Martius and Lampert (2016), for instance, propose a model where, activation functions correspond to symbolic operators acting on different dimensions of each layer, in such a way that, after training the model through backpropagation, the symbolic expression can be read out of the learned coefficients. Petersen and Landajuela (2019) adopt a different strategy, where syntactic properties of symbolic expressions are encoded as constraints within a generative recurrent architecture, which is then trained through reinforcement learning. In all these cases, training takes place within one set of values (i.e., corresponding to one function) so the learning procedure only concerns the selection of the symbolic expression for those values. More recently, an alternative approach has been proposed, more in line with current trends in AI: The model is trained on pairs of sets of values and corresponding symbolic expressions, learning to predict the latter from the former (Biggio et al., 2021; Valipour et al., 2021; d'Ascoli et al., 2022; Kamienny et al., 2022). From this perspective, *the symbolic expression appears as yet another property of a mathematical object* (e.g., a function) initially given through numerical expressions. Significantly, symbolic regression thus becomes a special case of language modeling, as Valipour et al. (2021) point out: "Symbolic mathematics behaves as a language in its own right, with well-formed mathematical expressions treated as valid 'sentences' in this language. It is natural, therefore, to consider using deep language models to address tasks involving symbolic mathematics" (p. 2).

A notable case of this recent orientation is that of d'Ascoli et al. (2022), where a neural model is trained to discover the recurrence relation holding between the first terms of a numerical sequence. The data is generated by sampling symbolic expressions of functions to produce a recurrence relation and compute the first 25 terms of a sequence. Symbolic expressions were encoded as sequentialized trees, following Lample and Charton (2020) (cf. above), while numbers (integers and floats) were encoded as in Charton (2021). A simple transformer architecture is then trained to predict a symbolic expression. Since many equivalent expressions could correspond to the initial function, a prediction is considered correct if the difference between the n following terms in the sequence computed with the predicted expression and those calculated with the initial expression is smaller than a given tolerance error. The authors showed their model could achieve an in-distribution accuracy of over 90% for integer sequences and over 70% for floats for the prediction of the following 5 terms (falling to 78% and 43%, respectively, for 10 terms), outperforming numeric methods. The model was also evaluated on the Online Encyclopedia of Integer Sequences (OEIS, Sloane (2007)) for out-of-domain generalization, where, even if performance dropped substantially, it showed some significant results and outperformed existing methods for this task in Mathematica.

Overall, despite many limitations and somewhat modest results often concerning relatively simple and well-known mathematical objects, this line of work has shown that the generic approach provided by DNNs can meaningfully manipulate, to a significant extent, both numeric and symbolic properties of mathematical objects of different kinds (arithmetical, algebraic, analytic, etc.). Admittedly, the relative simplicity of the objects under study, whose properties are in most cases well-known, is such that these results cannot be expected to replace or even compete with existing methods and practices (at least not anytime soon). But as Charton (2021) makes it clear, this is not the purpose of this work either. Its significance lies elsewhere, namely in showing that those well-known practices are not entirely orthogonal to the seemingly unrelated principles of neural nets. At any rate, this line of research demonstrates that if DNNs can do something with mathematical proofs, the same can be said for mathematical objects.

3.3 Skill-Oriented

A third group of works applying neural methods to mathematical knowledge follows a different strategy than the previous two. The idea is not so much to solve specific tasks involving mathematical statements (e.g., premise selection) or mathematical objects (e.g., symbolic integration) but to endow the neural model with general mathematical *skills* so that, once trained, it becomes capable of spontaneously solving mathematically related problems.

Stated in those abstract terms, one could wonder how such a goal could have any chance of success. However, hope in this research orientation is motivated by the turning point operated by transformer architectures in the field of natural language processing (NLP) and AI more generally. The original transformer model introduced by Vaswani et al. (2017) was conceived to improve machine translation performance while significantly reducing the training time. Its superiority compared to previous models had been demonstrated in practically all NLP tasks. Moreover, when trained as a *language model*, that is, as a model producing probability distributions over words or tokens given other tokens,Saxton et al. (2019) showed they could significantly outperform the best recurrent models on a variety of mathematical tasks (including algebra, arithmetic, calculus, and probability among others). To that end, transformers are trained on synthetic corpora where mathematical tasks are expressed in terms of natural language question answering, processed as a sequence of characters, e.g. (p. 2):

> Question: `Let x(g) = 9 * g + 1. Let q(c) = 2 * c + 1. Let f(i) = 3 * i 39. Let w(j) = q(x(j)). Calculate f(w(a))`
>
> Answer: `54 * a - 30`

However, the true power of transformers was revealed with the emergence of what came to be known as *Large Language Models* (LLMs). Taking advantage of the scaling capabilities of transformers, primarily due to their exploitation of parallelization, the distinctive feature of LLMs lies in their training strategy. Popularized thanks to the BERT model (Devlin et al., 2018), the idea is to train, or "*pre-train*" as it's referred to in the field, a transformer architecture in a highly generic self-supervised fashion, namely by learning to predict random masked words in natural language sentences over a gigantic amount of natural language text (typically scraped from all over the Internet). Despite its apparent simplicity, working in an autoregressive generative fashion,[4] such training endows the model with the general capacity to perform a large variety of natural language tasks in the form of a text to be generated as the continuation of a given prompt. Once pre-trained, the model can be further trained or "*fine-tuned*" on a significantly smaller domain-specific dataset, to perform tasks expressible in natural language and normally requiring domain-specific skills. A variant of this "transfer learning" strategy was introduced by the celebrated GPT-3 model (Brown et al., 2020), aiming to circumvent the need for fine-tuning by drastically scaling up both the model size and the training data.[5] This change in scale afforded generative pre-trained transformer models a unique feature, known as "in context-learning", namely, the capacity to perform domain-specific tasks without any further training, just by prompting the model with a few examples or a short description of the intended task. Thus, once the model has been pre-trained, one can input a short text with a few examples of, say, pairs of grammatically incorrect/correct English sentences before yet another incorrect sentence, after which the model will continue the text autoregressively, outputting the correct counterpart. Surprisingly, this highly unexpected behavior holds with reasonable accuracy for a remarkably wide range of tasks.[6]

Within this line of research, it became natural to prompt the generic model with tasks requiring mathematical skills, either after fine-tuning the model on mathematical data or in an in-context fashion. Shen et al. (2021), for instance, evaluate the performance of the original pre-trained BERT model when fine-tuned for three general tasks associated with mathematics education: large-scale knowledge component prediction, open-ended question answer scoring or autograding, and knowledge tracing correctness prediction (see Table 2 for examples). Their work shows that such a model achieves a performance of almost 92%, 89%, and 87% for the three tasks, respectively (the first and third measured as accuracy, the second as area under the curve). Moreover, the

[4] That is, iteratively generating text one word at a time from a sequence of words initially given, recursively augmented by the words produced by the model.

[5] Whereas BERT contained a maximum of 340 million parameters in its original version, and at the time of GPT-3's release, the largest model contained 17 billion, GPT-3 was composed of more than 10 times this number: 175 billion parameters. As for the data, GPT-3 was trained on 500 billion words (tokens) compared to 3,300 million for BERT.

[6] For a comprehensive account of LLMs, see Bommasani et al. (2021). For a precise yet accessible presentation of the mechanisms behind GPT, see, for instance, Wolfram (2023).

Task	Text
Knowledge component prediction	Simplify the expression: (z2)2 Put parentheses around the power if next to coefficient, for example: 3x2=3(x^2),x5=x^5
Question answer scoring	Q: Explain your answer on the box below. A: because it is the same shape, just larger, making it similar
Knowledge tracing correctness prediction	Q: What is 2.6 + (-10.9)? A: -8.3

Table 2: Example texts of the three tasks examined in Shen et al. (2021)

authors could improve this baseline by adding a pre-training step on a dedicated mathematical corpus between the original pre-training and the fine-tuning. The mathematical corpus included paper abstracts from arXiv, college MOOC syllabus, and textbooks and curricula ranging from pre-kindergarten to college graduate level.

As for in-context learning, the original GPT-3 paper by Brown et al. (2020) already showed the arithmetical skills of such a generic model, in particular, by trying to perform up to five-digit elementary arithmetic. The model is thus prompted with a few cases (typically a dozen) of arithmetic operations expressed in natural language, such as:

```
Q: What is 24 times 42?
A: 1008
```

ending with a line that stops at `A:`, after which the model is expected to fill in the subsequent text, hopefully with the correct answer to the last question. GPT-3 exhibited an accuracy of almost 100% on 2-digit addition and subtraction and over 94% and 80% for 3-digit subtraction and addition, respectively. These results are certainly surprising considering the model was never trained to perform such a task, but only to predict the next token given previous tokens. In addition, the authors provided evidence that only a negligible number of arithmetic problems tested could be found in the training data, suggesting that the model's performance cannot be attributed to simple data memorization. Nogueira et al. (2020) later showed that the capacity of LLMs to perform elementary arithmetical operations was highly dependent on the textual representation of numbers. By fine-tuning a pre-trained T5 model (Raffel et al., 2019) on textual expressions like the ones used in GPT-3, the authors demonstrated that, while single-token decimal notation performed very poorly, representing numbers as digits separated by powers of 10, either using scientific notation or not (e.g., `3 10e1 2 10e0` or `3 10 2` for 32), could afford the model almost perfect accuracy for the addition of numbers up to 15 digits. Their study also confirmed the importance of pre-training on generic masked language modeling: A transformer trained from scratch needs 10 times more training examples to reach the same performance on addition as a pre-trained model.

A noteworthy characteristic of this approach is that *any mathematical problem is treated by expressing it in natural language* (see, for instance, Griffith and Kalita, 2021; Meng and Rumshisky, 2019; Cobbe et al., 2021). Yet, as surprising as those elementary results may be from a conceptual standpoint, tasks requiring mathematical skills have proved challenging for LLMs generically trained on natural language data when compared to other (mostly linguistic) tasks, such as machine translation, summarization, or sentiment analysis (see, for instance, Hendrycks et al., 2021a; Davis, 2023).

Question: A line parallel to $y = 4x + 6$ passes through $(5, 10)$. What is the y-coordinate of the point where this line crosses the y-axis?

Target: Since the line is parallel to $y = 4x + 6$, we know that its slope is 4. Thus the equation of the line is $y = 4x + b$ where b is the y-intercept. Plugging $(5, 10)$ into this gives $10 = 4 \cdot 5 + b \Rightarrow b = \boxed{-10}$ which is what we wanted.

Model output: The line is parallel to $y = 4x + 6$, which means that it has the same slope as $y = 4x + 6$. Since the line passes through $(5, 10)$, we can use the point-slope form of a line to find the equation of the line:

$$\begin{aligned} y - 10 &= 4(x - 5) \\ y - 10 &= 4x - 20 \\ y &= 4x - 10 \end{aligned}$$

Now that we have the equation of the line, we can find the y-coordinate of the point where the line crosses the y-axis by substituting $x = 0$ into the equation:

$$y = 4 \cdot 0 - 10 = \boxed{-10}.$$

Figure 3: Example of a problem from the MATH dataset, correctly solved by Minerva (Lewkowycz et al., 2022). The solution is output in LaTeX, token by token.

As suggested above, a possible strategy for the skill-oriented line of research to remain relevant despite the difficulty is to further train pre-trained LLMs on large amounts of mathematical (or mathematically related) texts. This further training remains generic, in the sense of not being oriented toward any specific task: the training continues to take place as language modeling. A good example of this approach is provided by the Minerva model (Lewkowycz et al., 2022), where the PaLM LLM (Chowdhery et al., 2022) is further trained on a vast mathematical and scientific dataset built from the arXiv preprint server and mathematical webpages scraped from the Internet. The objective is to be able to perform both numerical calculations and symbolic manipulation without training on specific tasks or relying on principles foreign to the neural architecture (like a calculator or a compiler). To improve the performance, the model also implements a series of techniques, including prompting methods that encourage detailed answers and majority voting on samples of possible results to a question. A distinctive feature of Minerva is that, unlike typical language modeling training, mathematical expressions are not normalized to keep only the natural language text but include the syntax of mathematical typesetting systems such as LaTeX and MathJax. Thus, an expression like $E = mc^2$ is represented as the string `$E=mc^2$` instead of `Emc2`. In this way, the model input and output can include a more precise representation of mathematical expressions. With all these tweaks and tricks, Minerva was able to significantly outperform previous models on different mathematical benchmarks, including high school, college, and graduate math problems and STEM-focused language understanding problems, with gains of up to 30% in some cases. Figure 3 shows an example from the MATH dataset correctly solved by Minerva. The model exhibited an accuracy of over 50% on this dataset, compared to less than 20% for existing neural models at the moment of publication.

The challenges faced by a skill-oriented approach are more than just technical. Among others, the notion of “mathematical skill” is difficult to characterize. The field has progressively converged to an implicit agreement assuming an institution-centered answer to this question: mathematical skills are the ones measured by formal education tests, exams, and competitions. This choice is reflected in the benchmarks established by the field, which are essentially composed of textbooks and curricula from different educational levels, national (which often means: American) admission tests, like the

Scholastic Assessment Test (SAT), and institutional competitions such as the International Mathematics Olympiad (IMO). But such a choice is as uncritical as it is unreflective. Historical, geographical, and social biases remain unassessed. Also, the field's research habits leave no room to address the consequences of reinforcing a school-like conception of mathematics, which is fundamentally focused on performance due in part to the uninterpretable characteristics of the proposed models. Let alone the encouragement of an anthropomorphic image of neural models resulting from the idea that they can acquire skills and " beat" humans on tests designed for specific human purposes. Yet, if we refrain from taking these models at their face value and admit from a critical stance that the notion of mathematical skill is not independent of how educational institutions measure it at a given space and time, the performance exhibited by these models appears no less surprising. Compared to the previous orientations, the lack of dedicated model architectures (like in proof-oriented cases) or datasets (as in object-oriented approaches), as well as the highly unstructured and underspecified goals and means characterizing the skill-oriented perspective, endows those results with enough room for philosophical inquiry.

3.4 Heuristic-Oriented

The fourth and final research orientation differs from the previous ones in that neural models are not conceived as generic devices performing multiple tasks in a way supposedly akin to human agents. Instead, the idea is to use neural nets as *tools* addressing specific aspects of advanced open problems in mathematics in order to guide the intuition of mathematicians at work. From a philosophical perspective, this approach can be deemed one of the most sensible and fruitful, if only because it grants a central place to working mathematicians, whose judgment is arguably indispensable when mathematical practice is not reduced to reproducing existing mathematical knowledge. However, this orientation is the most underdeveloped in the current state of the field. Many factors may explain this situation. Starting with ideological reasons, related to a widespread belief in the field viewing AI systems as agents imitating, competing with, and ultimately replacing human agents. Such a position is not independent of economic conditions in a field that is largely driven by corporations. In its current state, AI research is not separable from profit-seeking through the development of market products, and the market size of advanced mathematical research is negligible compared to the one defined by the automation of known mathematical tasks in business and education. Finally, from a sociological perspective, the development of this orientation requires the close collaboration of two very different research communities, namely professional mathematicians (often in the area of pure mathematics) and applied computer scientists and engineers, which represents an additional challenge to the success of the attempts in this direction.

As a result of their heuristic nature, works belonging to this category are less characterized by unified methods, frameworks, and goals than by the individual adaptation of new machine learning tools to specific and highly technical mathematical problems. Accordingly, this orientation might be better presented through concrete cases rather than general trends. We will present two such cases, which represent two important aspects of mathematical heuristics: forming *conjectures* and searching for *counterexamples*.

Early uses of deep learning for conjecture generation can be found in Hughes (2016); Carifio et al. (2017); Levitt et al. (2019); Jejjala et al. (2019); Heal et al. (2020). Most of this work concerned combinatorial properties in knot theory. But it was undoubtedly the paper by Davies et al. (2021) that attracted renewed attention to this line of research by presenting the use of machine learning in mathematical practice as a general framework for working mathematicians. The authors present deep ‡neural methods as a tool "to guide [the mathematicians'] intuitions concerning complex mathematical objects, verifying their hypotheses about the existence of relationships and helping them understand those relationships" (p. 70). In their view, deep learning techniques can contribute to understanding the relation between two objects $X(z)$ and $Y(z)$ by helping to identify a function $\hat{f}$ such that $\hat{f}(X(z)) \approx Y(z)$ for some well-specified domain of z. Such a function can then be used to verify the existence of structural features in objects and understand them through attribution techniques. Accordingly, the proposed method consists in starting with a hypothesis for a function f, generating synthetic data, training a supervised model, and finding patterns through attribution techniques. The whole procedure can help mathematicians formulate conjecture candidates, informing, in turn, new hypotheses and data generation in an iterative and interactive process.

The authors of this work demonstrated the fruitfulness of their approach in the case of two open problems concerning the structure of knots and the combinatorial invariance of symmetric groups in representation theory. To better understand how neural techniques interact with mathematical objects and practices in this case, it may be helpful to examine certain aspects of this work. The result obtained concerns the connection between two distinct branches of knot theory, which could be addressed by finding relations between their respective characteristic invariants. In particular, the goal was to obtain a connection between the *signature* of knot (a classical algebraic invariant in gauge theory), and typical geometric invariants in hyperbolic knot theory. Such a connection was not yet known in the field. The strategy was, then, to predict the signature of a knot from its hyperbolic invariants.

Using existing libraries in knot theory, the authors produced a sample set of 2,7 million knots with their corresponding invariants, divided into a train and a test set, which served to train a very classic fully-connected feed-forward neural net to predict the signature from a list of hyperbolic invariants. Somewhat surprisingly, the model was able to predict the signature over the test set with very high accuracy, indicating that a connection does indeed exist between both kinds of properties of the knots. However, the high predictive accuracy can only do so much as verify that a structural connection exists. To obtain further insight into the structure of such a connection, the authors made use of gradient-based attribution techniques (saliency maps). The main idea is to measure the sensitivity of the prediction accuracy to small changes in the different dimensions of the input, which correspond, in this case, to the different hyperbolic invariants. In this way, the authors could determine that the prediction of the signature relied primarily on three out of twelve invariants, all three related to a geometric structural feature of the knot called "cusp". This observation allowed them to define a new quantity attached to a knot, which they termed "natural slope" and used to state the conjecture that the signature was equal to the natural slope multiplied by some error. Interestingly, this first conjecture turned out to be false. Further investigation resulted in a refined version of the conjecture, including a fourth invariant also identified by the attribution analysis. This eventually led the authors to prove two theorems connecting the two kinds of invariants.

Our first example shows that the success of neural nets in targeting functions predicting some mathematical features from others can contribute to verifying and understanding structural properties of mathematical objects and formulating and refining conjectures. On the other hand, their capacity to find arguments in the domain for which the prediction fails can provide a powerful tool to discover counterexamples of existing conjectures. Such a practice can be no less informative of the objects' structure. Wagner (2021) provides a notable example of this approach. This paper uses neural models to find counterexamples and constructions to open problems in combinatorics and graph theory. The strategy adopted shows some originality compared to previous approaches. The idea is to train a neural model to find a mathematical object through *reinforcement learning*, following the success of this technique in training automatic systems to play games, such as Go. From this perspective, mathematics appears as a game where counterexample candidates are akin to possible winning moves. The model is highly general in that it requires only minimal prior knowledge, namely the encoding of legitimate moves (i.e., the sort of mathematical construction expected) and a reward function providing a score to be used as feedback for how close the construction is from a winning move. Thus, by trying to maximize the score without knowing anything about the structure of the problem, the learning algorithm will improve the model's performance toward a winning move determined by the conjecture to be refuted.

Wagner (2021) applied this general strategy to a series of conjectures in extremal combinatorics, an area of mathematics interested in the maximum or minimum sizes of collections of finite objects (e.g., graphs) subject to some restriction. Significantly, within this framework, the general way to encode mathematic constructions is as *words*, in such a way that the combinatorial problem is translated into "a problem about generating a word of certain length from a finite alphabet" (p. 3). For instance, n-vertex graphs are represented as 0-1 sequences of length $\frac{n(n-1)}{2}$, which, as we have seen, can be generated by neural language models in an autoregressive fashion. The reward function computes a score once the entire word has been produced. The function typically includes a positive and a negative component reflecting desired and undesired properties given the problem studied. Using the reward function, the model is trained through gradient descent to minimize the cross-entropy loss.

A concrete example among the ones addressed by Wagner (2021) is a conjecture in spectral graph theory stating that for a connected graph of $n \geq 3$ vertices, with largest eigenvalue λ_1 and matching number μ, we have $\lambda_1 + \mu \geq \sqrt{n-1} + 1$. Although a counterexample to this conjecture already existed for $n = 600$, the method proposed was able to find a counterexample for $n = 19$ by setting n as a hyperparameter and training a neural model to minimize $\lambda_1 + \mu$ (set as the reward or loss function) hoping the loss would drop below $\sqrt{n-1} + 1$ for some construction after enough training. Figure 4 shows a sample of the best construction at different moments of the training process. The last graph constitutes a counterexample for the targeted conjecture. Interestingly, while the trained model remains uninterpretable, the analysis of the evolution of the model's output during training provides some insight into the structural properties of the problem. We can see that the model first restricts the outputs to sparse graphs and then converges to a "balanced double star" structure. As exemplified in the paper, this kind of structural insight can be obtained even in the case where the model fails to provide an explicit counterexample for the problem in question, either indicating how a counterexample could be eventually obtained or providing hints as to why the conjecture holds and how it could be proved.

The relevance of neural methods for heuristic purposes in mathematical research has been called into question, for instance, by Davis's (2021) critique of Davies et al. (2021), for it is not clear what the specific value added by DNNs is compared to more classic statistical methods already in use by mathematicians in their own practice.

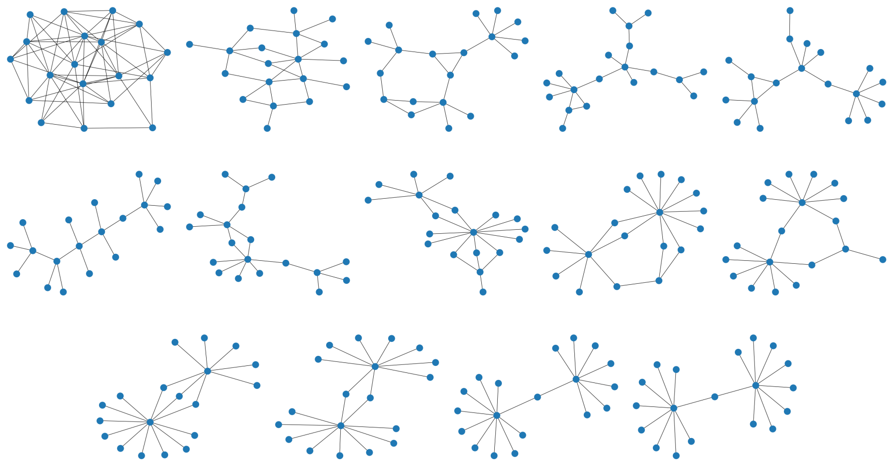


Figure 4: Sample of the best construction over the training progress (from left to right, and from top to bottom) in Wagner (2021). The last graph (bottom right) constitutes a counterexample for the targeted conjecture.

Moreover, as pointed out by Wagner (2021), methods like the ones used in this paper are not adequate to address all problems and, as the author acknowledges, they "did not succeed in refuting any of the most famous conjectures in the field" (p. 1). Nevertheless, the renewed strengths and capabilities of statistical methods brought about by deep learning techniques are such that their most elementary application to the treatment of open problems in advanced mathematics is far from trivial, neither regarding the means deployed nor the kind of results obtained. In particular, neural methods seem to force the development of statistical approaches in fields classically considered foreign to them. Even though a heuristic use of DNNs relies more on a case-by-case approach rather than a solid general framework, it seems that the near future will show a fruitful development of works like the ones presented in this section. Consequently, this line of work could pave the way to more fundamental research on the statistical aspects of various fields in pure mathematics.

4 Philosophical Significance

Given the state of the field at the moment of writing these lines, the survey presented in the previous section is certainly fragmentary, partial, and guaranteed to be outdated by the time the reader encounters these pages. The four categories proposed to understand existing research orientations should be taken as temporary markers on moving soil rather than well-defined boundaries on a stable domain. Many of the papers presented lie at the crossroad of multiple research lines, and one can expect numerous blends of the four orientations in the near evolution of the field. It is nonetheless revealing that, faced with the renewed question of how to apply novel machine learning techniques to mathematics, instead of resorting to well-established distinctions, the research orientations tend to be spontaneously organized according to the AI researchers' implicit assumptions as to *what it is that we do when we do mathematics*. Indeed, as far as lines of research can be identified in this rapidly evolving field, they resist any traditional disciplinary (e.g., algebra, calculus, etc.), epistemological (e.g., pure, applied), or foundational (e.g., logic, set theory, category theory, etc.) categorization. Instead, they consistently explore specific *practices* assumed to characterize what mathematics is about: proving theorems, manipulating objects, acquiring skills, or addressing open problems.

The results in each identified direction exhibit countless limitations, many of which are already revealed in the literature, others yet to discover. We have referred to some of them in our account, including brittleness, poor generalization, restriction to elementary cases, lack of interpretability, absence of conceptual and theoretical foundations, and uncritical attitude towards mathematical corpora (see also Davis, 2019; Welleck et al., 2022). All these limitations are real and could not be stressed enough if neural methods are to become a significant component of future mathematical practices. However, the simple fact that those mathematical practices can be addressed, if only with relative success, from the radically empirical perspective assumed by current machine learning approaches should be enough to arouse the interest of philosophers and historians of mathematics.

There are many reasons for the philosophy of mathematics, and the history and philosophy of mathematical practices in particular, to turn its attention to the recent mathematical applications of machine learning. Even if, or precisely because, those reasons do not coincide with the ones advanced by the researchers in the field. Starting with the fact that, were these approaches to succeed to the point of becoming widely adopted as mathematical tools, the image of mathematics as we know it would certainly change in ways that are far from trivial, neither from a philosophical nor from a historical standpoint. Such a modification of mathematical practices would significantly impact the many dimensions philosophers and historians have become used to identifying in them, such as the role of proofs (Chemla, 2012), the meaning of or even the need for foundations (Wagner, 2019), the nature of explanations (Mancosu, 2008), or the role of experimental procedures (Borwein and Bailey, 2003; Avigad, 2008).

However, in the remaining pages, we would like to focus on a general feature underlying what, in our view, constitutes the philosophical novelty of neural models across their different applications to mathematical practices: *the renewed role of language in our understanding of mathematics*.

4.1 The Return of Language

If there is something our survey of neural applications to mathematics brings to light, it is at least the following: The success—actual or promised—of those methods in all their areas of application is inseparable from a whole new perspective on mathematical language, which goes far beyond the traditional understanding of mathematical expressions as simple notations for pre-existing mathematical contents or a more or less arbitrary syntax for an independently determined semantics. The reason is that language, and more precisely, *writing*, including the wide variety of *textual forms* (statements, expressions, symbols, characters, etc.), *is all these models can rely on* to perform the mathematical tasks they are required to address. Whatever happens within the black box of a neural net happens on the basis of the processing of syntax and *syntax alone*. Granted, DNNs are not, as such, purely syntactic entities but the implementation of a relatively well-specified semantic procedure. However, such a procedure is highly generic, to the point that it can be successfully applied to very different kinds of data (images, language, sound, etc.) to perform significantly different kinds of tasks. The rapid convergence in recent years toward a single architecture (i.e., transformers) for all tasks confirms the resolve to rely on an unspecific analytic procedure. Given their generic character, it follows that if neural models can deal with mathematical content to any extent, such content is nowhere else to be found than in the corpus of (eventually supervised) textual expressions fed as inputs.

It is essential to understand that *this is not a bug but a feature* of this approach. Indeed, multiple ways exist to enhance neural models with hand-made rules and mathematical principles to increase their performance over specific tasks. As soon as accuracy becomes the primary concern, researchers do not hesitate to use all sorts of tweaks and tricks, introducing semantic features by other means. The resulting models progressively become, then, the object of an engineering practice, thus losing much of the philosophical interest concerning their relation to language. Nevertheless, the connectionist credo behind DNNs exhorts to develop learning models as generic concerning tasks as sparing regarding assumptions about the data. Consequently, whether openly acknowledged or unwittingly, current DNN applications to mathematical knowledge grant a critical place to mathematical language and textuality, providing the conditions for an original reflection on its role in the production and circulation of mathematical content.

Two primary consequences aspects stand out when assessing the originality of the role played by language in this setting. The first is the effect of the fact that bare syntactic objects are all these models can resort to for processing mathematical knowledge. Without dedicated axiom systems, deductive rules, logical operations, symbolic reasoning, material manipulation, or access to any other kind of context than textual, most of the conceptual focus within this line of research is put on mathematical *representations*. Be it logical statements, formulas, equations, expressions, or symbols,[7] researchers in machine learning have been led to raise the question of mathematical representations in ways that can communicate with current research interests in the philosophy of mathematics. Take, for instance, the proof-oriented approach and its reliance on multiple proof assistants' syntax to represent logical statements. The numerous strategies to process that syntax to improve the neural models' performance on inferential tasks (e.g., character- vs. token-based processing, explicit definition replacement, variable renaming, etc.) could provide novel perspectives for the design of mathematical languages, as addressed, for instance, by Avigad (2015); **?**. Furthermore, the various methods to encode mathematical objects in the object-oriented approaches (sequentialized binary trees for formulas, lists of numbers for matrices, different positional encodings for numbers, etc.) provide the elements for a fruitful dialogue with current investigations on the role of representations in the philosophy and history of mathematical practices, such as Schlimm (2018); Waszek (2018); Kohlhase et al. (2018). In any case, practically no study across the different orientations fails to underline the critical character of the corpus and its representation for the success or failure of neural methods in mathematics. Yet, this centrality of representational aspects is not without novelty compared to existing philosophical perspectives on this problem. For the goal here is not to find the best representation for a given content (a suitable "notation") but the best representation for given texts so that tasks assumed to rely on their unknown content can be correctly performed by a statistical learning model such as neural nets. The careful investigation of this original question may bring unprecedented perspectives and insights into the classical problem of the relation between syntax and semantics, in mathematics as well as in language in general.

The second remarkable consequence concerning mathematical language is related

[7]To the best of our knowledge, there has been no work within this area of research involving the content of mathematical diagrams yet. From a different perspective, Sørensen and Johansen (2020) have used neural techniques to identify diagrams in mathematical texts.

to the connection between mathematics and *natural language*. Indeed, addressed as an artificial language from a modern logical perspective, most philosophical traditions have kept mathematical language apart from natural language for almost one and a half centuries. Moving away from this standard position, neural machine learning models are led to operate novel articulations between both, which are far from philosophically trivial. On a superficial level, one can find that, since many of the corpora used are composed of existing published papers (especially since the emergence of LLMs), the natural language necessarily present in those papers is usually leveraged to contribute to determining the content of mathematical expressions. A conceptual investigation on this topic is likely to provide compelling insight into the interaction between both natural and mathematical expressions in different corpora, be it from a historical perspective, in line with works such as those of Netz (1999) or Herreman (2000), or from a more cognitive viewpoint (Giaquinto, 2008; Toffoli and Giardino, 2013; Kohlhase et al., 2018). However, deep learning approaches bring natural language to the forefront of mathematical practice and knowledge in a more fundamental fashion. For the vast majority of the learning models and techniques used for processing mathematical expressions are none other than those specifically developed for the processing of natural language. The recent convergence toward transformer architectures also confirms this circumstance. This means that, within this setting, *mathematical expressions are treated as being themselves a sort of natural language*. This idea is not only implicit in all uses of neural language modeling architectures in mathematical applications but, as we have seen, is repeatedly advanced by the researchers themselves across all research orientations. At odds with a view making a clear-cut distinction between natural and artificial languages or resolved to reduce the former to the latter through logical means, the success of neural methods in mathematics is inextricably tied to an essential yet unexplored link between mathematical contents and the mechanisms governing the ordinary practice of natural language. It could be, for instance, that mathematical and natural language share specific properties, or that the treatment of natural language constitutes a general framework for studying all kinds of languages, or yet that deep neural models constitute a general framework for the analysis of any type of data, including both natural language and mathematics. Whatever the case, the feats of neural models' applications to mathematics mark the return of natural language to the center of the reflection about what it means to do mathematics, in a philosophically original and compelling way.

4.2 New Challenges for a Linguistic Approach to Mathematical Practice

Despite the assumed divide between artificial and natural languages in modern philosophies of science, linguistic approaches to mathematics have not been altogether absent from the historical and philosophical scene, particularly since the emergence of the philosophy of mathematical practices. The chapters included in the current section of this Handbook provide an excellent account of much of that work. However, the many uses of neural models call for an original linguistic approach to mathematics, presenting new challenges to a language-driven philosophy of mathematics.

As already advanced, the originality lies in that *mathematical content is derived from expressions alone, semantics from syntax, meaning from pure text*. Yet, in modern mathematics, access to content is supposed to be provided through explicit and rigorous symbolic mechanisms, such as definitions within a formal language, logical operations, deductive rules, systems of axioms, or formal models. Those symbolic principles are assumed not only to determine what those expressions mean but also to control how expressions interact and what results from those interactions. Without direct access to explicit symbolic means—let alone cognitive faculties or social conditions—it is difficult to know how the semantics or the content of expressions can be determined.

The difficulty is even greater if one considers the mathematical content DNNs manipulate is not limited to *referential* aspects of expressions. Indeed, the solution of mathematical tasks requires much more than associating, for instance, the expression `406` to a particular number or quantity, or to the same entity as the expression `four hundred and six`; it goes beyond referring the expression $A \wedge B$ to a logical conjunction or $y'' - y = 0$ to a differential equation of second order. The tasks in question require, in addition, that enough *operational* content is involved when determining that `406` added to `326` equals `732`, that $A \wedge B$ is likely or unlikely to be a premise in the proof of some given logical statement, or that $y(x) = c_1 e^x + c_2 e^{-x}$ is the solution to that differential equation.

Resorting to how content is dealt with in natural language, as most models do, might sound promising because the range of expressions does not depend, in this case, on explicit rules and other artificial procedures—typically absent in neural models—but tends to rely on more "natural" mechanisms. The latter could be invoked as being also at work in the relation between mathematical expressions and their content to explain the surprising capabilities of neural nets. However, as natural as it may seem, if we restrict ourselves to pure expressions, the access to content in natural language is far from simple and direct. The meaning of words is usually given through other words, risking an infinite regression, and ostensive definitions are doomed to inescapable ambiguities unless considerable segments of content are already known, as the radical critiques of Wittgenstein (2009) and Quine (2013), among others, have sufficiently shown.

To make things worse, the recourse to natural language imposes significant constraints on the treatment of expressions. Indeed, DNNs can only achieve their results by addressing the problem as a predictive task. Thus, given the expressions `406` and `326` as inputs, the model is required to predict the expression `732`, after being trained over a massive quantity of similar cases (although not this specific one). Or more precisely, it is expected to predict the expression `732` from any expression containing `406`, `326` and some expression of the additive operation, as in `406+326=` or `what is 406 plus 326?` (or some sequential encoding thereof). Now, by approaching the task as if we were dealing with natural language, we force models to perform those predictions by the same means by which they can predict, say, `you` from `she asked` when generating the expression `she asked you`, or `Paris` from `The capital of France is`, however differently structured those expressions and contents may be.

Finally, analyzing those underlying means presents innumerable obstacles. As it is widely acknowledged, DNN models are practically uninterpretable (see for instance Lipton, 2018). In particular, we have no access to formal representations of a model other than its internal state, which can go from hundreds of billions up to over a trillion parameters in the current state of the art. Several methods have been developed in recent years, providing tools for the analysis and principles of interpretability of NLP neural models (see Belinkov and Glass (2019) and Madsen et al. (2021) for a survey). Among them, the method known as *probing* (Conneau et al., 2018) enjoys some success. The idea is to use the encoded vector representations of a model to train a classifier over a specific linguistic property considered relevant for performing linguistic tasks (e.g., grammatical dependencies). If the classifier achieves good performance, the properties in question can be regarded as somewhat encoded in the internal state of the original neural model. However, while several mechanisms have been proposed to show that the information corresponding to the probed property is legitimately attributable to and effectively employed by the model (Hewitt and Liang, 2019; Ravfogel et al., 2021), a fundamental gap remains between neural models and the symbolic feature being assessed. In other words, high—and ultimately superhuman—performance on such predictive tasks, which can indeed be conceived to grasp significant aspects of the content of expressions, might not reach that accuracy through the same means as humans. Neural models might be able to write sensible natural language or solve natural language tasks without ever following grammatical rules for any acceptable sense of grammatical rule-following. Likewise, those models might be able to solve mathematical tasks with reasonable and even surprising accuracy without ever manipulating any of the symbolic procedures by which we have historically controlled our mathematical practices. Converging results do not imply identical theories, representations, knowledge, or capabilities, which, in turn, does not mean that content is not being grasped. Interpretability attempts like the one of Charton (2022) presented above are certainly remarkable and promising but still limited and, all in all, largely insufficient given the scope of the problem.

5 Conclusion: But How?

In the face of all these obstacles, which are by no means unknown to researchers in the field, why should we place any hope at all in contemporary machine learning approaches to mathematical knowledge? Why should we rely on the orientations that a critical assessment could derive from them for a linguistically driven philosophy of mathematical practice? Isn't the entire enterprise simply ill-conceived?

If the proof of the pudding is in the eating, it seems fair to acknowledge that, while the results exhibited so far are certainly limited from a mathematical point of view, they are far from insignificant from a philosophical one. Given the scale of the challenges, the slightest evidence that a seemingly impossible task can be achieved deserves philosophical consideration. Moreover, the models in question have shown surprising capacities in the treatment of natural language, where the richness of content makes the task otherwise challenging. Indeed, despite these models' duly identified limitations, the results exhibited by the application of DNNs to natural language processing in the last decade have been enough to catalyze, if not altogether generate, a revolution in the field (cf. Manning, 2015).

Due to the close relation between mathematical texts and natural language within this framework, it seems legitimate to assume that the hope placed in neural methods for treating mathematical language is grounded on the mechanisms underlying the success of neural language models, whatever they may be. In this sense, the symbolic character of mathematical practices cannot be an objection *a priori* to this approach. On the one hand, it is not excluded that neural models are implicitly manipulating some higher-order representation akin to symbolic properties or structures (cf., for instance, Manning et al., 2020). On the other, symbolic practices in mathematics as we know them are a relatively recent phenomenon. Although mathematical practices across different historical periods and cultures have constantly used expressive means that are irreducible to those of natural language, the recourse to a foundational role of symbolic practices such as axiomatic methods, formal systems of inference, or model-theoretical semantics has only become standard in the course of the past century. Mathematical knowledge has very well been produced and evolved for millennia across different cultures without such foundations (cf. Wagner, 2019) and would arguably continue to do so if alternative principles and practices came to fulfill better the tasks expected from it in existing cultures, even if that requires to re-inspect our existing notion of semantics, content or even understanding, both in natural language and in mathematics.

Accordingly, a philosophical investigation of this phenomenon should focus on the elementary mechanisms responsible for the success of generic learning algorithms in manipulating natural language content when applied to a practically raw corpus of texts, and on the reasons and conditions for those same mechanisms to apply to the treatment of mathematical content. To avoid resorting to dubious, or at any rate unverifiable analogies between artificial neural models and human learning capacities, let alone biological characteristics of the human brain, it is imperative to turn our attention to what those models *actually do*. As we have seen, DNNs are statistical models resulting from a statistical learning procedure in the context of predictive tasks. If such models can grasp linguistic content when operating with pure text, it is natural to ask what the source of that content can be. Since the expressions fed into the model are nothing but a sequence of empty identifiers, and models are highly generic before training, the source of all content, be it linguistic or mathematical, can lie nowhere else but in the training corpus. Which is another way of saying: in the series of collective and historically determined linguistic and mathematical practices, as they are recorded in the many forms textuality can take.

That significant segments of such content can be drawn from the expression of those practices is something the limited yet startling results of mathematical applications of neural language models are displaying before our eyes. The task remains to understand *how* that is even possible and invest this feat with the philosophy it deserves. With it, a whole new program opens up before a philosophy of mathematical practices informed by a renewed theory of language and signs.

Acknowledgements The author wishes to thank Deniz Sarikaya, Bharath Sriraman, David Waszek, Roy Wagner, and John Terilla for their patient support, precious feedback, and constant encouragement.

Funding Information This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 839730

References

Alexander A. Alemi, François Chollet, Niklas Een, Geoffrey Irving, Christian Szegedy, and Josef Urban. Deepmath - deep sequence models for premise selection. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2243–2251, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Jeremy Avigad. *Computers in Mathematical Inquiry*, chapter 11, pages 134–150. Oxford University Press, New York, 2008.

Jeremy Avigad. Mathematics and language, 2015. URL https://arxiv.org/abs/1505.07238.

Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, Christian Szegedy, and Stewart Wilcox. Holist: An environment for machine learning of higher-order theorem proving (extended version). *CoRR*, abs/1904.03241, 2019. URL http://arxiv.org/abs/1904.03241.

Yonatan Belinkov and James Glass. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 7:49–72, 04 2019. ISSN 2307-387X. doi: 10.1162/tacl_a_00254. URL https://doi.org/10.1162/tacl_a_00254.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales, 2021.

Jan Blechschmidt and Oliver G. Ernst. Three ways to solve partial differential equations with neural networks — a review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021. doi: https://doi.org/10.1002/gamm.202100006.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL https://arxiv.org/abs/2108.07258.

Jonathan M. Borwein and David H. Bailey. *Mathematics by experiment - plausible reasoning in the 21st century*. A K Peters, 2003. ISBN 978-1-56881-211-3.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2 edition, 2022. doi: 10.1017/9781009089517.

Jonathan Carifio, James Halverson, Dmitri Krioukov, and Brent D. Nelson. Machine learning in the string landscape. *Journal of High Energy Physics*, 2017(9):157, 2017. ISSN 1029-8479. doi: 10.1007/JHEP09(2017)157. URL https://doi.org/10.1007/JHEP09(2017)157.

François Charton. Linear algebra with transformers. *CoRR*, abs/2112.01898, 2021. URL https://arxiv.org/abs/2112.01898.

François Charton. What is my math transformer doing? – three results on interpretability and generalization, 2022.

Karine Chemla. *The history of mathematical proof in ancient traditions*. Cambridge University Press, Cambridge, 2012.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1198. URL https://aclanthology.org/P18-1198.

Stéphane d'Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, and François Charton. Deep symbolic regression for recurrent sequences. *CoRR*, abs/2201.04600, 2022.

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis, and Pushmeet Kohli. Advancing mathematics by guiding human intuition with AI. *Nature*, 600(7887):70–74, December 2021. doi: 10.1038/s41586-021-04086-x. URL https://doi.org/10.1038/s41586-021-04086-x.

Ernest Davis. The use of deep learning for symbolic integration: A review of (lample and charton, 2019), 2019.

Ernest Davis. Deep learning and mathematical intuition: A review of (davies et al. 2021). *CoRR*, abs/2112.04324, 2021.

Ernest Davis. Mathematics, word problems, common sense, and artificial intelligence, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

Deborah Ferreira and André Freitas. Premise selection in natural language mathematical texts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.657. URL https://aclanthology.org/2020.acl-main.657.

Deborah Ferreira and André Freitas. STAR: Cross-modal [STA]tement [R]epresentation for selecting relevant mathematical premises. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3234–3243, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.282. URL https://aclanthology.org/2021.eacl-main.282.

Karlis Freivalds and Renars Liepins. Improving the neural GPU architecture for algorithm learning. *CoRR*, abs/1702.08727, 2017. URL http://arxiv.org/abs/1702.08727.

Zoubin Ghahramani. Introducing PaLM 2, 2023. URL https://blog.google/technology/ai/google-palm-2-ai-large-language-model/.

Marcus Giaquinto. *Cognition of Structure*, chapter 2, pages 43–64. Oxford University Press, New York, 2008.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge MA, London UK, 2016.

Kaden Griffith and Jugal Kalita. Solving arithmetic word problems with transformers and preprocessing of problem text. *CoRR*, abs/2106.00893, 2021. URL https://arxiv.org/abs/2106.00893.

Kathryn Heal, Avinash Kulkarni, and Emre Can Sertöz. Deep learning gauss-manin connections. *CoRR*, abs/2007.13786, 2020. URL https://arxiv.org/abs/2007.13786.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Alain Herreman. *La topologie et ses signes: éléments pour une histoire sémiotique des mathématiques*. L'Harmattan, Paris, 2000.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks, 2019.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8.

Mark C. Hughes. A neural network approach to predicting and computing knot invariants, 2016.

Vishnu Jejjala, Arjun Kar, and Onkar Parrikar. Deep learning the hyperbolic volume of a knot. *Physics Letters B*, 799:135033, 2019. ISSN 0370-2693. doi: https://doi.org/10.1016/j.physletb.2019.135033. URL https://www.sciencedirect.com/science/article/pii/S0370269319307555.

Albert Qiaochu Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr Mił oś, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding hammers to integrate language models and automated theorem provers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 8360–8373. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/377c25312668e48f2e531e2f2c422483-Paper-Conference.pdf.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SMa9EAovKMC.

Cezary Kaliszyk, François Chollet, and Christian Szegedy. Holstep: A machine learning dataset for higher-order logic theorem proving. *CoRR*, abs/1703.00426, 2017. URL http://arxiv.org/abs/1703.00426.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers, 2022.

Samuel Kim, Peter Y. Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Ceperic, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 32: 4166–4177, 2019.

Andrea Kohlhase, Michael Kohlhase, and Taweechai Ouypornkochagorn. Discourse phenomena in mathematical documents. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *Intelligent Computer Mathematics*, pages 147–163. Springer International Publishing, 2018.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1eZYeHFDS.

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. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 26337–26349. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a8901c5e85fb8e1823bbf0f755053672-Paper-Conference.pdf.

Dennis Lee, Christian Szegedy, Markus Rabe, Sarah Loos, and Kshitij Bansal. Mathematical reasoning in latent space. In *International Conference on Learning Representations*, 2020.

Jesse S F Levitt, Mustafa Hajij, and Radmila Sazdanovic. Big data approaches to knot theory: Understanding the structure of the jones polynomial, 2019.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 3843–3857. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

Zachary C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, jun 2018. ISSN 1542-7730. doi: 10.1145/3236386.3241340. URL https://doi.org/10.1145/3236386.3241340.

Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. *CoRR*, abs/1701.06972, 2017. URL http://arxiv.org/abs/1701.06972.

Andreas Madsen, Siva Reddy, and Sarath Chandar. Post-hoc interpretability for neural nlp: A survey, 2021. URL https://arxiv.org/abs/2108.04840.

Paolo Mancosu. *Mathematical Explanation: Why it Matters*, chapter 5, pages 134–150. Oxford University Press, New York, 2008.

Christopher D. Manning. Computational Linguistics and Deep Learning. *Computational Linguistics*, 41(4):701–707, 12 2015. ISSN 0891-2017. doi: 10.1162/COLI_a_00239. URL https://doi.org/10.1162/COLI_a_00239.

Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907367117.

Georg Martius and Christoph H. Lampert. Extrapolation and learning equations. *CoRR*, abs/1610.02995, 2016. URL http://arxiv.org/abs/1610.02995.

John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Magazine*, 27(4):12, Dec. 2006.

Yuanliang Meng and Anna Rumshisky. Solving math word problems with double-decoder transformer. *CoRR*, abs/1908.10924, 2019. URL http://arxiv.org/abs/1908.10924.

Reviel Netz. *The Shaping of Deduction in Greek Mathematics*. Cambridge University Press, 1999.

Allen Newell and Herbert A Simon. Plans for the Dartmouth Summer research project on artificial intelligence. Typescript, March 1956. Supplement to McCarthy, et al. (2006).

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. *CoRR*, abs/2003.06713, 2020.

Aditya Sanjay Paliwal, Sarah M. Loos, Markus N. Rabe, Kshitij Bansal, and Christian Szegedy. Graph representations for higher-order logic and theorem proving. In *AAAI Conference on Artificial Intelligence*, 2019.

Brenden K. Petersen and Mikel Landajuela. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv: Learning*, 2019.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *CoRR*, abs/2009.03393, 2020. URL https://arxiv.org/abs/2009.03393.

Willard Van Orman Quine. *Word and Object, new edition (The MIT Press)*. The MIT Press, paperback edition, 2013. ISBN 978-0262518314.

Markus N. Rabe and Christian Szegedy. Towards the automatic mathematician. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction – CADE 28*, pages 25–37, Cham, 2021. Springer International Publishing. ISBN 978-3-030-79876-5.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Shauli Ravfogel, Grusha Prasad, Tal Linzen, and Yoav Goldberg. Counterfactual interventions reveal the causal effect of relative clause representations on agreement prediction. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 194–209, Online, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.conll-1.15. URL https://aclanthology.org/2021.conll-1.15.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models, 2019.

Dirk Schlimm. Numbers through numerals. the constitutive role of external representations. In Sorin Bangu, editor, *Naturalizing Logico-Mathematical Knowledge: Approaches from Psychology and Cognitive Science*, pages 195–217. 2018.

Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil T. Heffernan, Xintao Wu, and Dongwon Lee. Mathbert: A pre-trained language model for general NLP tasks in mathematics education. *CoRR*, abs/2106.07340, 2021.

Neil J. A. Sloane. The on-line encyclopedia of integer sequences. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, pages 130–130, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-73086-6.

Christian Szegedy. A promising path towards autoformalization and general artificial intelligence. In Christoph Benzmüller and Bruce Miller, editors, *Intelligent Computer Mathematics*, pages 3–20, Cham, 2020. Springer International Publishing. ISBN 978-3-030-53518-6.

Henrik Kragh Sørensen and Mikkel Willum Johansen. Counting mathematical diagrams with machine learning. In Ahti-Veikko Pietarinen, Peter Chapman, Leonie Bosveld-de Smet, Valeria Giardino, James Corter, and Sven Linker, editors, *Diagrammatic Representation and Inference*, pages 26–33. Springer International Publishing, 2020.

Silvia De Toffoli and Valeria Giardino. Forms and roles of diagrams in knot theory. *Erkenntnis*, 79(4):829–842, November 2013. doi: 10.1007/s10670-013-9568-7. URL https://doi.org/10.1007/s10670-013-9568-7.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/0e64a7b00c83e3d22ce6b3acf2c582b6-Paper.pdf.

Alan Turing. Intelligent Machinery (1948). In *The Essential Turing*. Oxford University Press, 09 2004. ISBN 9780198250791. doi: 10.1093/oso/9780198250791.003.0016.

Silviu-Marian Udrescu and Max Tegmark. Ai feynman: a physics-inspired method for symbolic regression, 2019.

Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. Symbolicgpt: A generative transformer model for symbolic regression. *ArXiv*, abs/2106.14131, 2021.

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

Adam Zsolt Wagner. Constructions in combinatorics via neural networks, 2021.

Roy Wagner. *Does Mathematics Need Foundations?*, pages 381–396. Springer International Publishing, Cham, 2019. ISBN 978-3-030-15655-8. doi: 10.1007/978-3-030-15655-8_17.

Mingzhe Wang and Jia Deng. Learning to prove theorems by learning to generate theorems. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18146–18157. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d2a27e83d429f0dcae6b937cf440aeb1-Paper.pdf.

David Waszek. *Les représentations en mathématiques*. PhD thesis, 2018. URL http://www.theses.fr/2018PA01H231. Thèse de doctorat dirigée par Panza, Marco Philosophie Paris 1 2018.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. Naturalproofs: Mathematical theorem proving in natural language. *CoRR*, abs/2104.01112, 2021. URL https://arxiv.org/abs/2104.01112.

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

Geordie Williamson. Is deep learning a useful tool for the pure mathematician?, 2023.

Ludwig Wittgenstein. *Philosophical investigations*. Wiley-Blackwell, Chichester, West Sussex, U.K. ;, 4th ed. edition, 2009. ISBN 1-282-55045-4.

Stephen Wolfram. What Is ChatGPT Doing... and Why Does It Work? `https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/`, 2023. Accessed: 2023-07-09.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Norman Rabe, Charles E Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=IUikebJ1Bf0`.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=9ZPegFuFTFv`.

Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms, 2016.