

On automating mathematical insight

Asvin G.*

Recent accomplishments in AI and machine learning have come with profound shocks to our perception of the intellectual world, and our place in it. Progress in almost every intellectual field is rapidly increasing, and while automating research in various fields is still mostly for the future, it no longer seems impossible. For various reasons, mathematics seems to be an ideal testing ground in order to push the capabilities of our silicon friends, and in this essay I will explore some plausible strategies that could lead to automating research level work in mathematics.

A short history of recent AI

The modern revolution in AI is closely tied to progress in computer hardware capabilities and is mostly through the paradigm of "deep learning". While these ideas have been around for a while, they were impractical to implement on real world use cases until recently (the last decade), and the resulting capabilities have been utterly surprising to just about everyone. There are two major landmarks that are of importance to our story.

The first is AlphaGo and AlphaZero, Deepmind's AI systems that learn to play Go (and a wider class of games respectively) through *self-play*, to a superhuman level. In short, one starts with a neural network that, on any given game state, computes a "policy function" and a "game state value". The policy function is a probability distribution over the allowed moves in the position, while the "game state value" is a real number proportional to how good the board state is for the player to move. We train these systems by using these functions to play through games (against itself) and then update the functions through gradient descent based on the outcome of each game. I would like to highlight two important aspects of these systems.

First, the "signal" for any given game is incredibly weak. We only get one bit of information at the end of the game that provides aggregated information over how good every single move in that game was (of which there could be dozens). A priori, this is a very insufficient way to learn where all the mistakes and insights during the game are completely correlated, and unlike human learning, there is no attempt to even disentangle them. Second, the learned semantic content is completely implicit in the neural network, specifically in the "game state value". The higher level concepts that humans use to understand and explicitly discuss the game are completely hidden, to the extent that one might doubt whether the network even learns any approximate analogues of the concepts humans are familiar with. Further research on these networks has found that indeed, these networks learn and recover standard human concepts and even concepts unknown to humans that we can probe and understand from the trained neural network.

This feeds into the first point where the lack of an explicit conceptual compositional description of the play strategy makes it difficult to iterate upon and improve the strategy, except through implicit gradient descent. These inefficiencies are overcome through playing through a number of games that would be impossible in any one human lifetime, or even through many lifetimes.

The second major landmark is the introduction of, and astounding success of Large Language Models (LLMs). In this framework, one trains a neural network to predict the analogue of a "policy function", but for text in any given language (usually English). In this paradigm, roughly speaking, one collects a vast corpus of text (such as all the data on the internet, for example), and trains a neural network to predict a probability distribution over the set of all words in the language after we mask a particular word in pieces of this corpus.

*email: gasvinseeker94@gmail.com

Amazingly, this is sufficient to induce a variety of skills in the trained neural network, from a superhuman understanding of grammar, natural language processing, coding ability and even reasoning in various contexts, amongst many other skills. In this context, the learned semantic understanding is even more implicit in the network, and harder to extract, but at the same time it is hard to deny that these networks do learn approximately the same semantics as humans do.

In the rest of the paper, we will see how this strategy might potentially be adapted to mastering mathematics and what implications the above observations might have in that context. But first, I would like to propose a characterization of how any intelligent agent must solve any given problem.

Problem solving as a heuristically guided search

I claim that all intelligent problem solving has two major components to it.

1. An explicit search over strings of discrete tokens in a language adapted to the problem. For example, this could be alpha-beta search as in AlphaGo, explicit reasoning as in the latest AI models or human mathematicians sketching a proof before attempting to make it rigorous.
2. An implicit policy function guiding the above search process. In AlphaGo, this is precisely the deep neural network, in LLMs this corresponds to probability distribution over the tokens resulting from a forward pass while in humans, we simply call it intuition, when we bother to name it at all.

If the problem domain is simple enough, then it would suffice to enumerate all possibilities in order to find the solution. However, this is almost never the case and by the very nature of search, the possibilities grow exponentially in the length of the solution. This is precisely the role of the heuristic guide that lets us cut down the space of possibilities to something manageable and explore longer strings than would be possible otherwise. In other words, the implicit policy function effectively *reshapes the search landscape*, making some potential solutions much more accessible at the cost of making different ones inaccessible.

It is clear then that the success or failure of our problem solver rests entirely on the effectiveness of our policy function, and learning a "good" policy function is then the chief purpose of solving any particular problem. However, there is one final twist to this story: in solving truly difficult problems, we do not merely improve our policy functions but reshape our search landscapes in a more radical fashion by *rewriting the language in which we perform the search in the first place!* We do this at both small and large scales through introducing concepts piecemeal (every new definition in a mathematical paper is an example of this) or by rewriting the entire foundations of the subject (general relativity, evolutionary theory and modern algebraic geometry are just a few such examples).

While it is possible in principle to capture the changes of such conceptual novelty through merely improving the implicit policy function, there are several advantages to finding a new language:

1. Defining a new concept explicitly allows for iterative refinement over time through the advantages of compositionality, counteracting the inefficiencies of gradient descent (or other updating procedures) in this direction.
2. It allows others (both across time and space) to learn the concept through seeing it used. This is the key step in transitioning from individual intelligence to collective intelligence. For instance, while our chess engines play at a superhuman level, extracting insights from them is a slow and difficult process exactly due to this lack of explicitness.
3. It allows us the flexibility to use different languages towards different goals - for instance, different levels of abstraction or rigor and to leverage *translation* to switch between different modes of thinking.

Automating mathematical research

How might one adapt this strategy to mathematical research? A first attempt might be the following. One might train a neural network on "self-play" where each game consists of trying to find either a proof or

disproof of a given mathematical statement. This strategy comes with some obvious obstacles:

1. Checking that a proposed proof is indeed correct is difficult to automate. This problem can be circumvented through implementing the strategy in a formal theorem proving language such as Lean.
2. Due to the vast number of possible steps at each point and the very narrow path that a correct proof has to walk through (as opposed to the many possible "valid" games of Go, say), even moderately short proofs might take impossibly long to find if our neural network is not already trained.

This is a serious obstacle, but taking inspiration from LLMs, we might want to first train our neural net on an existing corpus of proofs. In this case, training in a natural language as opposed to a formal language is favorable because mathematicians mostly write proofs in English. However, this is quickly changing and one might even first try to train a network to auto-formalize english language proofs first.

3. Solving any specific difficult problem will need knowledge and intuition specific to that area that can only be gained through exploration, and not through general "pre-training". An attempt at implementing this would be train, simultaneously, a deep learning system that finds "plausible variations" of a given problem statement on which the prover can be further fine tuned through the self-play procedure discussed above.

This exact strategy, including the auto-formalization, has been implemented by Deepmind through AlphaProof and the resulting system performs impressively on IMO problems. Will this strategy, scaled up, also reach the frontier of mathematical research? I suspect that the answer is *no* precisely because this does not attempt to leverage the crucial advantages of conceptual rewriting discussed above.

The other strategy one might attempt is to simply scale up existing LLMs, perhaps fine tuned on natural language proofs, and combine it with a reasoning model. This is the strategy taken by OpenAI, for example, and also has seen impressive success with the latest models sometimes able to answer non-trivial mathematical questions at a graduate level (see the Frontier Math benchmark), or even on completely novel questions. There is some non-trivial hope that merely scaling up these systems will introduce within the ability to rewrite the language it searches in, craft new definitions and find new foundations for the problem domain.

But the question remains: *Can we train an AI system to create novel concepts well?*

Training the revolutionary spirit within AI

I propose that the act of writing new foundations is nothing more and nothing less than an act of *compositional compression*, where both words are important. Language itself, in the first place, is functional precisely through these two features. Our language is an efficient (compressed) way of describing the phenomenon we interact with regularly and care about but more than that, it is *compositional*. That is to say, each individual word represents a specific operator (acting on other words) in such a way that the dynamics of these language operators together capture the dynamics of the phenomenon we observe.

In rewriting the foundations of a discipline (or inventing a new concept), what we are identifying is a compression of the existing corpus of knowledge in that discipline in a way that clarifies and simplifies the aspects that *we care about*, potentially at the cost of making certain other aspects more complicated or obtuse. As such, it is very much a subjective task but since the importance of any particular idea in a science is relative to all the other concepts it impinges upon, it might be possible to formalize this.

If we do buy this framework, the question remains as to how one might train an AI system to do this conceptual rewriting. I believe this is an important, difficult open problem but an important lead is provided by the following paper [EJBL24] which provides a way of measuring the "compositional complexity" of a "compositional compression" in our terms. I suspect that it should be possible to train a neural network to "play the game" of minimizing this compositional complexity through self-play and pre-training as discussed above but this remains to be tried.

References

- [EJBL24] Eric Elmoznino, Thomas Jiralerspong, Yoshua Bengio, and Guillaume Lajoie. A complexity-based theory of compositionality. *arXiv preprint arXiv:2410.14817*, 2024.