THE NEW YORKER

HOW THE ARTIFICIAL-INTELLIGENCE PROGRAM ALPHAZERO MASTERED ITS GAMES

By James Somers December 28, 2018

In 2016, a Google program soundly defeated Lee Sedol, the world's best Go player, in a match viewed by more than a hundred million people.

Photograph by Ahn Young-joon / AP

A few weeks ago, a group of researchers from Google's artificial-intelligence subsidiary, DeepMind, published a paper in the journal *Science* that described an A.I. for playing games. While their system is general-purpose enough to work for many two-person games, the researchers had adapted it specifically for Go, chess, and shogi ("Japanese chess"); it was given no knowledge beyond the rules of each game. At first it made random moves. Then it started learning through self-play. Over the course of nine hours, the chess version of the program played forty-four million games against itself on a massive cluster of specialized Google hardware. After two hours, it began performing better than human players; after four, it was beating the best chess engine in the world.

The program, called AlphaZero, descends from AlphaGo, an A.I. that became known for defeating Lee Sedol, the world's best Go player, in March of 2016. Sedol's defeat was a stunning upset. In "AlphaGo," a documentary released earlier this year on Netflix, the filmmakers follow both the team that developed the A.I. and its human opponents, who have devoted their lives to the game. We watch as these humans experience the stages of a new kind of grief. At first, they don't see how they can lose to a machine: "I believe that human intuition is still too advanced for A.I. to have caught up," Sedol says, the day before his five-game match with AlphaGo. Then, when the machine starts winning, a kind of panic sets in. In one particularly poignant moment, Sedol, under pressure after having lost his first game, gets up from the table and, leaving his clock running, walks outside for a cigarette. He looks out over the rooftops of Seoul. (On the Internet, more than fifty million people were watching the match.) Meanwhile, the A.I., unaware that its opponent has gone anywhere, plays a move that commentators called creative, surprising, and beautiful. In the end, Sedol lost, 1-4. Before there could be acceptance, there was depression. "I want to apologize for being so powerless," he said in a press conference. Eventually, Sedol, along with the rest of the Go community, came to appreciate the machine. "I think this will bring a new paradigm to Go," he said. Fan Hui, the European champion, agreed. "Maybe it can show humans something we've never discovered. Maybe it's beautiful."

AlphaGo was a triumph for its creators, but still unsatisfying, because it depended so much on human Go expertise. The A.I. learned which moves it

should make, in part, by trying to mimic world-class players. It also used a set of hand-coded heuristics to avoid the worst blunders when looking ahead in games. To the researchers building AlphaGo, this knowledge felt like a crutch. They set out to build a new version of the A.I. that learned on its own, as a "tabula rasa."

The result, AlphaGo Zero, detailed in a paper published in October, 2017, was so called because it had zero knowledge of Go beyond the rules. This new program was much less well-known; perhaps you can ask for the world's attention only so many times. But in a way it was the more remarkable achievement, one that no longer had much to do with Go at all. In fact, less than two months later, DeepMind published a preprint of a third paper, showing that the algorithm behind AlphaGo Zero could be generalized to any two-person, zero-sum game of perfect information (that is, a game in which there are no hidden elements, such as face-down cards in poker). DeepMind dropped the "Go" from the name and christened its new system AlphaZero. At its core was an algorithm so powerful that you could give it the rules of humanity's richest and most studied games and, later that day, it would become the best player there has ever been. Perhaps more surprising, this iteration of the system was also by far the simplest.

A typical chess engine is a hodgepodge of tweaks and shims made over decades of trial and error. The best engine in the world, Stockfish, is open source, and it gets better by a kind of Darwinian selection: someone suggests an idea; tens of thousands of games are played between the version with the idea and the version without it; the best version wins. As a result, it is not a particularly elegant program, and it can be hard for coders to understand. Many of the changes programmers make to Stockfish are best formulated in terms of chess, not computer science, and concern how to evaluate a given situation on the board: Should a knight be worth 2.1 points or 2.2? What if it's on the third rank, and the opponent has an opposite-colored bishop? To illustrate this point, David Silver, the head of research at DeepMind, once listed the moving parts in Stockfish. There are more than fifty of them, each requiring a significant amount of code, each a bit of hard-won chess arcana: the Counter Move Heuristic; databases of known endgames; evaluation modules for Doubled Pawns, Trapped Pieces, Rooks on (Semi) Open Files, and so on; strategies for searching the tree of possible moves, like "aspiration windows" and "iterative deepening."

AlphaZero, by contrast, has only two parts: a neural network and an algorithm called Monte Carlo Tree Search. (In a nod to the gaming mecca, mathematicians refer to approaches that involve some randomness as "Monte Carlo methods.") The idea behind M.C.T.S., as it's often known, is that a game like chess is really a tree of possibilities. If I move my rook to d8, you could capture it or let it be, at which point I could push a pawn or move my bishop or protect my queen. . . . The trouble is that this tree gets incredibly large incredibly quickly. No amount of computing power would be enough to search it exhaustively. An expert human player is an expert precisely because her mind automatically identifies the essential parts of the tree and focusses its attention there. Computers, if they are to compete, must somehow do the same.

Chess commentators have praised AlphaZero, declaring that the engine "plays like a human on fire."
Photograph Courtesy DeepMind Technologies

This is where the neural network comes in. AlphaZero's neural network receives, as input, the layout of the board for the last few moves of the game. As output, it estimates how likely the current player is to win and predicts which of the currently available moves are likely to work best. The M.C.T.S. algorithm uses these predictions to decide where to focus in the tree. If the network guesses that 'knight-takes-bishop' is likely to be a good move, for example, then the M.C.T.S. will devote more of its time to exploring the consequences of that move. But it balances this "exploitation" of promising moves with a little "exploration": it sometimes picks moves it thinks are unlikely to bear fruit, just in case they do.

At first, the neural network guiding this search is fairly stupid: it makes its predictions more or less at random. As a result, the Monte Carlo Tree Search starts out doing a pretty bad job of focussing on the important parts of the tree. But the genius of AlphaZero is in how it learns. It takes these two half-working parts and has them hone each other. Even when a dumb neural network does a bad job of predicting which moves will work, it's still useful to look ahead in the game tree: toward the end of the game, for instance, the M.C.T.S. can still learn which positions actually lead to victory, at least some of the time. This knowledge can then be used to improve the neural network. When a game is done, and you know the outcome, you look at what the neural network predicted for each position (say, that there's an 80.2 per cent chance that castling is the best move) and compare that to what actually happened (say, that the percentage is more like 60.5); you can then "correct" your neural network by tuning its synaptic connections until it prefers winning moves. In essence, all of the M.C.T.S.'s searching is distilled into new weights for the neural network.

With a slightly better network, of course, the search gets slightly less misguided—and this allows it to search better, thereby extracting better information for training the network. On and on it goes, in a feedback loop that ratchets up, very quickly, toward the plateau of known ability.

When the AlphaGo Zero and AlphaZero papers were published, a small army of enthusiasts began describing the systems in blog posts and YouTube videos and building their own copycat versions. Most of this work was explanatory—it flowed from the amateur urge to learn and share that gave rise to the Web in the first place. But a couple of efforts also sprung up to replicate the work at a large scale. The DeepMind papers, after all, had merely described the greatest Go- and chess-playing programs in the world—they hadn't contained the source code, and the company hadn't made the programs themselves available to players. Having declared victory, its engineers had departed the field.

Gian-Carlo Pascutto, a computer programmer who works at the Mozilla Corporation, had a track record of building competitive game engines, first in

chess, then in Go. He followed the latest research. As the combination of Monte Carlo Tree Search and a neural network became the state of the art in Go A.I.s, Pascutto built the world's most successful open-source Go engines—first Leela, then LeelaZero—which mirrored the advances made by DeepMind. The trouble was that DeepMind had access to Google's vast cloud and Pascutto didn't. To train its Go engine, DeepMind used five thousand of Google's "Tensor Processing Units"—chips specifically designed for neural-network calculations—for thirteen days. To do the same work on his desktop system, Pascutto would have to run it for seventeen hundred years.

To compensate for his lack of computing power, Pascutto distributed the effort. LeelaZero is a federated system: anyone who wants to participate can download the latest version, donate whatever computing power he has to it, and upload the data he generates so that the system can be slightly improved. The distributed LeelaZero community has had their system play more than ten million games against itself—a little more than AlphaGo Zero. It is now one of the strongest existing Go engines.

MORE FROM

ANNALS OF TECHNOLOGY

"Reverse Innovation" Could Save Lives. Why Aren't We Embracing It?

By Tom Vanderbilt

A Study on Driverless-Car Ethics Offers a Troubling Look Into Our Values

By Caroline Lester

How Voting-Machine Lobbyists Undermine the Democratic Process

By Sue Halpern

It wasn't long before the idea was extended to chess. In December of last year, when the AlphaZero preprint was published, "it was like a bomb hit the community," Gary Linscott said. Linscott, a computer scientist who had worked on Stockfish, used the existing LeelaZero code base, and the new ideas in the AlphaZero paper, to create Leela Chess Zero. (For Stockfish, he had developed a testing framework so that new ideas for the engine could be distributed to a fleet of volunteers, and thus vetted more quickly; distributing the training for a neural network was a natural next step.) There were kinks to sort out, and educated guesses to make about details that the DeepMind team had left out of their papers, but within a few months the neural network began improving. The chess world was already obsessed with AlphaZero: posts on chess.com celebrated the engine; commentators and grandmasters pored over the handful of AlphaZero games that DeepMind had released with their paper, declaring that this was "how chess ought to be played," that the engine "plays like a human on fire." Quickly, Lc0, as Leela Chess Zero became known, attracted hundreds of volunteers. As they contributed their computer power and improvements to the source code, the engine got even better. Today, one core contributor suspects that it is just a few months away from overtaking Stockfish. Not long after, it may become better than AlphaZero itself.

When we spoke over the phone, Linscott marvelled that a project like his, which would once have taken a talented doctoral student several years, could now be done by an interested amateur in a couple of months. Software libraries for neural networks allow for the replication of a world-beating design using only a few dozen lines of code; the tools already exist for distributing computation among a set of volunteers, and chipmakers such as Nvidia have put cheap and powerful G.P.U.s—graphics-processing chips, which are perfect for training neural networks—into the hands of millions of ordinary computer users. An algorithm like M.C.T.S. is simple enough to be implemented in an afternoon or two. You don't even need to be an expert in the game for which you're building an engine. When he built LeelaZero, Pascutto hadn't played Go for about twenty years.

David Silver, the head of research at DeepMind, has pointed out a seeming paradox at the heart of his company's recent work with games: the simpler its programs got—from AlphaGo to AlphaGo Zero to AlphaZero—the better they performed. "Maybe one of the principles that we're after," he said, in a talk in December of 2017, "is this idea that by doing less, by removing complexity from the algorithm, it enables us to become more general." By removing the Go knowledge from their Go engine, they made a better Go engine—and, at the same time, an engine that could play shogi and chess.

It was never obvious that things would turn out this way. In 1953, Alan Turing, who helped create modern computing, wrote a short paper titled, "Digital Computers Applied to Games." In it, he developed a chess program "based on an introspective analysis of my thought processes while playing." The program was simple, but in its case simplicity was no virtue: like Turing, who wasn't a gifted chess player, it missed much of the depth of the game and didn't play very well. Even so, Turing conjectured that the idea that "one cannot programme a machine to play a better game than one plays oneself" was a "rather glib view." Although it sounds right to say that "no animal can swallow an animal heavier than itself," plenty of animals can. Similarly, Turing suggested, there might be no contradiction in a bad chess player making a chess program that plays brilliantly. One tantalizing way to do it would be to have the program learn for itself.

The success of AlphaZero seems to bear this out. It has a simple structure, but it's capable of learning surprisingly deep features of the games it plays. In one section of the AlphaGo Zero paper, the DeepMind team illustrates how their A.I., after a certain number of training cycles, discovers strategies well-known to master players, only to discard them just a few cycles later. It is odd and a little unsettling to see humanity's best ideas trundled over on the way to something better; it hits close to home in a way that seeing a physical machine exceed us—a bulldozer shifting a load of earth, say—doesn't. In a recent editorial in *Science*, Garry Kasparov, the former chess champion who lost to I.B.M.'s Deep Blue in 1997, argues that AlphaZero doesn't play chess in a way that reflects the presumably systematic "priorities and prejudices of programmers"; instead—even though it searches far fewer positions per move than a traditional engine—it plays in an open, aggressive style and seems to think in terms of strategy rather than tactics, like a human with uncanny vision. "Because AlphaZero programs itself," Kasparov writes, "I would say that its style reflects the truth."

Playing chess like a human, of course, isn't the same thing as thinking about chess like a human, or learning like one. There is an old saying that game-playing is the *Drosophila* of A.I.: as the fruit fly is to biologists, so games like Go and chess are to computer scientists studying the mechanisms of intelligence. It's an evocative analogy. And yet it could be that the task of playing chess, once it's converted into the task of searching tens of thousands of nodes per second in a game tree, exercises a different kind of intelligence than the one we care about most. Played in this way, chess might be more like earth-moving than we thought: an activity that, in the end, isn't our forté, and so shouldn't be all that dear to our souls. To learn, AlphaZero needs to play millions more games than a human does— but, when it's done, it plays like a genius. It relies on churning faster than a person ever could through a deep search tree, then uses a neural network to process what it finds into something that resembles intuition. Surely the program teaches us something new about intelligence. But its success also underscores just how much the world's best human players can see by means of a very different process—one based on reading, talking, and feeling, in addition to playing. What may be most surprising is that we humans have done as well as we have in games that seem, now, to have been made for machines.

James Somers is a writer and a programmer based in New York. *Read more »*

Video

The Girls Who Slay at Chess
The girls of competitive chess are vastly outnumbered by boys, but these young players won't be intimidated.