"A BRILLIANT TRAVEL GUIDE TO THE COMING WORLD OF AI."
—*Jeanette Winterson*

MARCUS DU SAUTOY

THE CREATIVITY CODE

ART AND INNOVATION IN THE AGE OF AI

First published in the United Kingdom in 2019 as *The Creativity Code: How AI Is Learning to Write, Paint and Think* by
Fourth Estate, an imprint of HarperCollins Publishers, London

First US edition published by Harvard University Press, 2019
All rights reserved
Printed in the United States of America

First printing

Jacket art: Detail of *Mona Lisa*, Leonardo da Vinci, Louvre, Paris,
France / Bridgeman Images

Jacket design: Graciela Galup

9780674240414 (EPUB)

9780674240421 (MOBI)

9780674240407 (PDF)

Library of Congress Cataloging-in-Publication Data on file at https://www.loc.gov/

ISBN 978-0-674-988132

Book design by Chrissy Kurpeski

CONTENTS

/ 1 /

The Lovelace Test

Works of art make rules;
rules do not make works of art.

—CLAUDE DEBUSSY

The machine was a thing of beauty. Towers of spinning brass cogs with numbers on their teeth were pinned to rods driven by a gear train. The seventeen-year-old Ada Byron was transfixed as she cranked the handle of the Difference Engine. Its inventor, Charles Babbage, had invited her to see it in action as it whirred and clicked away, mechanically calculating polynomial sums. Ada had always had a fascination with mathematics and inventions, encouraged by the tutors her mother had been eager to provide.

But it may have been the artistic genes she'd inherited from her father, the poet Lord Byron, that set Ada to imagining what such marvellous machinery might be capable of. A decade later—now married and become Countess of Lovelace—she turned her attention to Babbage's designs for an even more sophisticated calculator. It dawned on her that it would be more than just a number cruncher:

> The Analytical Engine does not occupy common ground with mere "calculating machines." It holds a position wholly its own; and the considerations it suggests are most interesting in their nature.

Ada Lovelace's notes are now recognized as the first inroads into the creation of code, the spark of the idea that has ignited the artificial intelligence revolution sweeping the world today, fueled by the work of pioneers like Alan Turing, Marvin Minsky, and Donald Michie. Yet Lovelace herself was cautious as to how much any machine could achieve: "It is desirable to guard against the possibility of exaggerated ideas that might arise as to the powers of the Analytical Engine," she wrote. "The Analytical Machine has no pretensions whatever to *originate* anything. It can do whatever we *know how to order it* to perform."

Ultimately, she believed, it was limited: you couldn't get more out than you had put in.

This belief has been a mantra of computer science for many years, a shield against the fear that someday programmers will set in motion a computer they cannot control. Some have gone so far as to suggest that to program a machine to be artificially intelligent, we would first have to understand human intelligence. But in the last few years a new way of thinking about code has emerged: a shift from a top-down approach to programming to bottom-up efforts to get the code to chart its own path. It turns out you don't have to solve intelligence first. You can allow algorithms to roam the digital landscape and learn just like children. Today's code is making surprisingly insightful moves, spotting hard-to-detect features in medical images and making shrewd trades on the stock market. This generation of coders believes it can finally prove Ada Lovelace wrong: that you can get more out than you programmed in.

Yet there is still one realm of human endeavor that most people believe the machines will never be able to touch. We have this extraordinary ability to imagine, to innovate and create. Our code, the creativity code, is one we have long felt that no programmer could ever crack. This is a code that we believe depends on being human.

Mozart's *Requiem* allows us to contemplate our own mortality. Witnessing a performance of *Othello* invites us to navigate the landscape of love and jealousy. A Rembrandt portrait conveys so much more than what the sitter looked like. How could a machine ever replace or compete with Mozart, Shakespeare, or Rembrandt? And, of course, human creativity extends beyond the arts. The molecular gastronomy of Michelin star chef Heston Blumenthal, the football trickery of Dutch striker Johan Cruyff, the curvaceous buildings of Zaha Hadid, the invention of the Rubik's Cube by Hungarian Erno Rubik, even the code behind a game like *Minecraft*—all involve great acts of human creativity.

One of the things that drives me to spend hours at my desk conjuring up equations and penning proofs is the thrill of creating something new. My greatest moment of creativity, one I think back to again and again, was the time I conceived of a new symmetrical object. No one knew this object was possible. But after years of hard work and a momentary flash of white-hot inspiration I wrote on my yellow notepad the blueprint for this novel shape. That sheer buzz of excitement is the allure of creativity.

But what do we really mean by this shape-shifting term? Those who have tried to pin it down usually circle around three factors. They speak of creativity as the drive to come up with something that is new, that is surprising, and that has value.

It turns out it's easy to make something new. I can get my computer to churn out endless proposals for new symmetrical objects. Surprise and value are more difficult to produce. In the case of my symmetrical creation, I was legitimately surprised by what I'd cooked up, and so were other mathematicians. No one was expecting the strange new connection I'd discovered between this symmetrical object and the unrelated subject of number theory. My object suggested a new way of understanding an area of mathematics that is full of unsolved problems, and that is what gave it value.

We all get sucked into patterns of thought. We think we see how the story will evolve and then suddenly we are taken in a new direction. This element of surprise makes us take notice. It is probably why we get a rush when we encounter creativity. But what gives something value? Is it a question of price? Does it have to be recognized by others? I might value a poem or a painting I've created, but my conception of its value is unlikely to be shared more widely. A surprising novel with lots of plot twists could be of relatively little value, but a new and surprising approach to storytelling, or architecture, or music—one that changes the way we see or experience things—will generally be recognized as having value. This is what Kant refers to as "exemplary originality," when an original act becomes an inspiration for others. This form of creativity has long been thought to be uniquely human.

At some level, all these expressions of creativity are the products of neuronal and chemical activity. Creativity is a code that evolution across millions of years has honed inside our brains. If we unpick the creative outpourings of the human species, we can start to see that there are rules underlying the creative process. So is our creativity in fact more algorithmic and rule-based than we might want to acknowledge? Can we hope to crack the creativity code?

This book aims to explore the limits of the new AI to see whether it can match or even surpass the marvels of our human code. Could a machine paint, compose music, or write a novel? It may not be able to compete with Mozart, Shakespeare, or Picasso, but could it be as creative as a child when asked to write a story or paint a scene?

My daughters are being creative when they build their Lego castles. My son is heralded as a creative midfielder when he leads his football team to victory. We speak of solving everyday problems creatively and running organizations creatively. The creative impulse is a key part of what distinguishes humans from other animals, and yet we often let it stagnate inside us, falling into the trap of becoming slaves to our formulaic lives. Being creative requires a jolt to take us out of the well-carved paths we retrace each day. This is where a machine might come in: perhaps it could give us that jolt, throw up a new suggestion, stop

/ 2 /

Three Types of Creativity

The chief enemy of creativity is good sense.

—PABLO PICASSO

The value placed on creativity in modern times has led to a range of writers and thinkers trying to articulate what it is, how to stimulate it, and why it is important. It was while serving on a committee convened by the Royal Society to assess what impact machine learning would likely have on society that I first encountered the theories of Margaret Boden.

Boden is an original thinker who over the decades has managed to fuse many different disciplines: she is a philosopher, psychologist, physician, AI expert, and cognitive scientist. In her eighties now, with white hair flying like sparks and an ever-active brain, she enjoys engaging with the question of what these "tin cans," as she likes to call computers, might be capable of. To this end, she has identified three different types of human creativity.

Exploratory creativity involves taking what is already there and exploring its outer edges, extending the limits of what is possible while remaining bound by the rules. Bach's music is the culmination of a journey that baroque composers embarked on to explore tonality by weaving together different voices. His preludes and fugues pushed the boundaries of what was possible before breaking the genre open and ushering in the classical era of Mozart and Beethoven. Renoir and Pissarro reconceived how we could visualize nature and the world around us, but it was Claude Monet who really pushed the boundaries, painting his water lilies over and over until his flecks of color dissolved into a new form of abstraction.

Mathematics revels in this type of creativity. The classification of Finite Simple Groups is a tour de force of exploratory creativity. Starting from the simple definition of a group of symmetries—a structure defined by four simple axioms—mathematicians spent 150 years compiling the list of every conceivable element of symmetry. This effort culminated in the discovery of the Monster simple group: it has more symmetries than there are atoms in the Earth and yet fits into no pattern of other groups. This form of mathematical creativity involves pushing limits while adhering strictly to the rules of the game. Those who engage in it are like the geographical explorers who, even as they discover previously unknown territory, are still bound by the limits of our planet.

Boden believes that exploration accounts for 97 percent of human creativity. This is also the sort of creativity at which computers excel. Pushing a pattern or set of rules to an extreme is a perfect exercise for a computational mechanism that can perform many more calculations than the human brain can. But is it enough to yield a truly original creative act? When we hope for that, we generally imagine something more utterly unexpected.

To understand Boden's second type, *combinational creativity*, think of an artist taking two completely different constructs and finding a way to combine them. Often the rules governing one will suggest an interesting new framework for the other. Combination is a very powerful tool in the realm of mathematical creativity. The eventual solution of the Poincaré conjecture, which describes the possible shapes of our universe, was arrived at by applying the very different tools used to understand flow over surfaces. In a leap of creative genius, Grigori Perelman landed at the unexpected realization that by knowing the way a liquid flows over a surface one could classify the possible surfaces that might exist.

My own research takes tools from number theory that have been used to understand primes and applies them to classify possible symmetries. The symmetries of geometric objects don't look at first sight anything like numbers. But applying the language that has helped us to navigate the mysteries of the primes and replacing primes with symmetrical objects has revealed surprising new insights into the theory of symmetry.

The arts have also benefited greatly from this form of cross-fertilization. Philip Glass took ideas he learned from working with Ravi Shankar and used them to create the additive process that is the heart of his minimalist music. Zaha Hadid combined her knowledge of architecture with her love of the pure forms of the Russian painter Kasimir Malevich to create a unique style of curvaceous buildings. In cooking, creative master chefs have fused cuisines from opposite ends of the globe.

There are interesting hints that this sort of creativity might also be perfect for the world of AI. Take an algorithm that plays the blues and combine it with the music of Boulez and you will end up with a strange hybrid composition that might just create a new sound world. Of course, it could also be a dismal cacophony. The coder needs to find two genres that can be fused algorithmically in an interesting way.

It is Boden's third form of creativity that is the more mysterious and elusive. What she calls *transformational creativity* is behind those rare moments that are complete game changers. Every art form has these gear shifts. Think of Picasso and cubism. Schoenberg and atonality. Joyce and modernism. They are phase changes, like when water suddenly goes from liquid to gas or solid. This was the image Goethe hit upon when he sought to describe how he was able to write *The Sorrows of Young Werther*. He devoted two years to wrestling with how to tell the story, only for a startling event, a friend's suicide, to act as a sudden catalyst. "At that instant," he recalled in *Dichtung und Wahrheit*, "the plan of *Werther* was found; the whole shot together from all directions, and became a solid mass, as the water in a vase, which is just at the freezing point, is changed by the slightest concussion into ice."

At first glance it would seem hard to program such a decisive shift, but consider that, quite often, these transformational moments hinge on changing the rules of the game, or dropping a long-held assumption. The square of a number is always positive. All molecules come in long lines, not chains. Music must be written inside a harmonic scale structure. Eyes go on either sides of the nose. There is a meta rule for this type of creativity: start by dropping constraints and see what emerges. The creative act is to choose what to drop—or what new constraint to introduce—such that you end up with a new thing of value.

If I were asked to identify a transformational moment in mathematics, the creation of the square root of minus one, in the mid-sixteenth century, would be a good candidate. This was a number that many mathematicians believed did not exist. It was referred to as an imaginary number (a dismissive term first used by Descartes to indicate that there was no such thing). And yet its creation did not contradict previous mathematics. It turned out it had been a mistake to exclude it. Now consider, if that error had persisted to today: Would a computer come up with the concept of the square root of minus one if it were fed only data telling it that there is no number whose square could be negative? A truly creative act sometimes requires us to step outside the system and create a new reality. Can a complex algorithm do that?

The emergence of the romantic movement in music is in many ways a catalog of rule-breaking. Instead of hewing to close key signatures as earlier composers had done, upstarts like Schubert chose to shift keys in ways that deliberately defied expectations. Schumann left chords unresolved that Haydn or Mozart would have felt compelled to complete. Chopin composed dense moments of chromatic runs and challenged rhythmic expectations with his unusual accented passages and bending of tempos. The move from one musical era to another, from Medieval to Baroque to Classical to Romantic to Impressionist to Expressionist and beyond, is one long story of smashing the rules. It almost goes without saying that historical context plays an important role in allowing us to define something as new. Creativity is not an absolute but a relative activity. We are creative within our culture and frame of reference.

Could a computer initiate the kind of phase change that can move us into a new state? That seems a challenge. Algorithms learn how to act based on the data presented to them. Doesn't this mean that they will always be condemned to producing more of the same?

As the epigraph of this chapter, I chose Picasso's observation that the "chief enemy of creativity is good sense." That sounds, on the face of it, very much against the spirit of the machine. And yet, one can program a system to behave irrationally. One can create a meta rule that will instruct it to change course. As we shall see, this is in fact something machine learning is quite good at.

Can Creativity Be Taught?

Many artists like to fuel their own creation myths by appealing to external forces. In ancient Greece, poets were said to be inspired by the muses, who breathed a kind of creative energy into their minds, sometimes sacrificing the poet's sanity in the process. For Plato, "a poet is holy, and never able to compose until he has become inspired, and is beside himself and reason is no longer in him . . . for no art does he utter but by power divine." The great mathematician Srinivasa Ramanujan likewise attributed his insights to ideas imparted to him in dreams by the goddess Namagiri, his family's deity. Is creativity a form of madness or a gift of the divine?

One of my mathematical heroes, Carl Friedrich Gauss, was known for covering his tracks. Gauss is credited with creating modern number obvious path. This involves deep immersion in what we have created to date. Out of that deep understanding might emerge something never seen before. It is important to impress on students that there isn't very often some big bang that resounds with the act of creation. It is gradual. Van Gogh expressed it well: "Great things are not done by impulse but by small things brought together."

I find Boden's second type, combinational creativity, to be a powerful weapon in stimulating new ideas. I often encourage students to attend seminars and read papers in subjects that don't seem connected with the problems they are tackling. A line of thought from a distant corner of the mathematical universe might resonate with the problem at hand and stimulate a new idea. Some of the most creative bits of science are happening today at the junctions of the disciplines. The more we can stray beyond our narrow lanes to share our ideas and problems, the more creative we are likely to be. This is where a lot of the low-hanging fruit is found.

Boden's third type, transformational creativity, seems hard at first sight to harness as a strategy. But again, the goal is to test the status quo by dropping some of the constraints that have been put in place. Try seeing what happens if you change one of the basic rules you have accepted as part of the fabric of your subject—it's dangerous, because by doing so you can collapse the system. But this brings me to one of the most important ingredients needed to foster creativity, and that is embracing failure.

Unless you are prepared to fail, you will not take the risks that will allow you to break out and create something new. This is why our education system and our business environment, both realms that abhor failure, are terrible environments for fostering creativity. If I want creativity from my students, I have learned, it is important to celebrate their failures as much as their successes. Sure, their failures won't make it into the PhD thesis, but so much can be learned from them. When I meet with my students, I repeat again and again Samuel Beckett's call to "Try. Fail. Fail again. Fail better."

Are these strategies that can be written into code? In the past, the top-down approach to coding meant there was little prospect of creativity in the output. Coders were never very surprised by what their algorithms produced. There was no room for experimentation or failure. But this all changed recently—because an algorithm, built on code that learns from its failures, did something that was new, shocked its creators, and had incredible value. This algorithm won a game that many believed was beyond the abilities of a machine to master. As we will see in Chapter 3, it was a game that required creativity to play.

/ 3 /

Ready Steady Go

We construct and construct,
but intuition is still a good thing.

—PAUL KLEE

People often compare mathematics to playing chess, and certainly there are connections. But when Deep Blue beat the best chess master the human race could offer, in 1997, it did not lead to the closure of mathematics departments. Although chess is a good analogy for the formal effort of constructing a proof, there is another game that mathematicians have regarded as much closer to their work, because it also features a creative and intuitive side. That is the Chinese game of Go.

I first discovered Go when I visited the mathematics department at Cambridge as an undergraduate to explore whether to do my PhD with the amazing group that had helped complete the Classification of Finite Simple Groups, a sort of Periodic Table of Symmetry. As I sat talking about the future of mathematics with John Conway and Simon Norton, two of the architects of this great project, I kept being distracted by students at the next table furiously slamming black and white stones onto a grid carved into a large, wooden board.

Eventually I asked Conway what they were doing. "That's Go," he explained. "It's the oldest game that is still being played to this day." In contrast to chess, with its warlike quality, Go is a game of territory capture. Players take turns placing their white or black pieces (or stones) onto a grid nineteen squares wide and nineteen squares high. If one player manages to surround a collection of the other's stones, those stones are captured. The game is over when all the stones have been placed, and the winner is the player who has captured the most stones. It sounds rather simple. The challenge of the game is that, as you are pursuing efficient captures of your opponent's stones, you must also avoid having your own stones captured.

"It's a bit like mathematics," Conway explained. "Simple rules that give rise to beautiful complexity." In fact, it was while Conway watched a game playing out between two experts, as they drank coffee in that common room, that he formed the germ of the idea he would later call "surreal numbers." As a Go match moves to its end game, it behaves like this new sort of number.

I've always been fascinated by games. When I travel abroad I like to learn whatever game I find the locals playing and bring it back with me. So when I got back from the wilds of Cambridge to the safety of my home in Oxford, I decided to buy a Go set from the local toy shop and see just what appeal it held for these obsessed students. As I began to explore the game with an Oxford classmate, I realized how subtle it was. It seemed impossible to devise a strategy that would lead to a win. In an important respect, a Go game proceeds in a direction opposite to chess. Whereas with chess, one's choices of moves get simplified with every piece removed from the board, with this game the constant addition of stones to the board makes the situation ever more complicated.

The American Go Association estimates that it would take a number with three hundred digits to count the number of games of Go that are legally possible. For chess, the computer scientist Claude

Shannon estimated that a 120-digit number (now called the Shannon number) would suffice. These are not small numbers in either case, but they give you a sense of how big the difference is in possible permutations.

As a kid I played a lot of chess and enjoyed working through the logical consequences of a proposed move. It appealed to the growing mathematician in me. Because the moves in chess branch out in a controlled manner, it is a manageable task for a computer, or even a human, to comprehend the tree of possibilities and analyze the implications of going down different branches. In contrast, the complexity of Go makes it impossible to analyze the tree of possibilities in any reasonable timeframe. This is not to say that Go players don't work to anticipate the logical consequences of their moves, but it does imply that they also rely on a more intuitive feel for the pattern of play.

The human brain is acutely attuned to discerning whatever structure and pattern there is to be found in a visual image. An experienced Go player looks at the lay of the stones and, by tapping into the brain's strength at pattern recognition, is able to spot a valuable next move. For computers, mimicking this very basic human skill has traditionally been a struggle. Machine vision is a challenge that engineers have wrestled with for decades.

The human brain's highly developed sense of visual structure has been honed over millions of years and has been key to our survival. Any animal's ability to survive depends in part on its ability to pick out structure from the visual mess that nature offers up. A pattern in the chaos of the jungle likely indicates the presence of another animal—and if you fail to take notice, that animal might eat you (or at least you will miss your chance to eat it). The human code is extremely good at reading patterns, interpreting how they might develop, and responding appropriately. It is one of our key assets, and it plays into our appreciation for the patterns in music and art.

It turns out that pattern recognition is precisely what I do as a mathematician when I venture into the remoter reaches of the mathematical jungle. I can't rely on a simple, step-by-step logical analysis of the

Hassabis soon graduated to a Commodore Amiga, which could be programmed to play the games he enjoyed. Chess was still too complicated, but he managed to program the Commodore to play Othello, a game that looks rather similar to Go. Its stones, which are black on one side and white on the other, get flipped when they are trapped between stones of an opponent's color. It's not a game that merits grandmasters, so he tried his program out on his younger brother. It beat him every time.

This was classic if-then programming: he needed to code in by hand the response to each of his opponent's moves. It was "if your opponent plays that move, then reply with this move." The machine's capability all came from Hassabis and his ability to see what the right responses were to win the game. It still felt a bit like magic, though. Code up the right spell and then, rather like *The Sorcerer's Apprentice,* the Commodore would go through the work of winning the game.

Hassabis raced through his schooling, which culminated at the age of sixteen with an offer to study computer science at Cambridge. He'd set his heart on Cambridge after seeing Jeff Goldblum in the film *The Race for the Double Helix*. "I thought, is this what goes on at Cambridge? You go there and you invent DNA in the pub? Wow."

Cambridge wouldn't let him start his degree at the age of sixteen, so he had to defer for a year. To fill his time, he won a place working for a game developer, having come in second in a competition run by *Amiga Power* magazine. While there, he created his own game, *Theme Park,* in which players build and run their own theme park. The game was hugely successful, selling several million copies and winning a Golden Joystick Award. With enough funds to finance his time at university, Hassabis set off for Cambridge.

His course introduced him to the greats of the AI revolution: Alan Turing and his test for intelligence; Arthur Samuel and his program to play checkers; John McCarthy, who coined the term artificial intelligence; Frank Rosenblat and his first experiments with neural networks. These were the shoulders on which Hassabis aspired to stand.

It was while sitting in lectures at Cambridge that he heard a professor repeat the mantra that a computer could never play Go because of the game's great reliance on creativity and intuition. This was like a red rag waved in front of the young Hassabis. He left Cambridge determined to prove that professor wrong.

His idea was that, rather than try to write the program himself that could play Go, he would write the meta-program that could write the program to play Go. It sounded crazy, but the concept was that this meta-program could, as the Go-playing program played more and more games, learn from its mistakes.

Hassabis had learned about a similar idea implemented by artificial intelligence researcher Donald Michie in the 1960s. Michie had written an algorithm called MENACE that learned from scratch the best strategy to play tic-tac-toe or, as it is called in the UK, noughts and crosses. (MENACE stood for Machine Educable Noughts And Crosses Engine.) To demonstrate the algorithm Michie had rigged up 304 matchboxes representing all the possible layouts of X's and O's encountered while playing. Each matchbox was filled with different colored balls to represent possible moves. Balls were removed or added to the boxes to punish losses or reward wins. As the algorithm played more and more games, the reassignment of the balls eventually led to an almost perfect strategy for playing. It was this idea of learning from mistakes that Hassabis wanted to use to train an algorithm to play Go.

Hassabis could base his strategy on another good model. A newborn baby does not have a brain that is preprogrammed with knowledge of how to make its way through life. It is programmed instead to learn as it interacts with its environment.

If Hassabis was going to emulate the way the brain learns to solve problems, then knowing how the brain works was clearly going to help. So he decided to do a PhD in neuroscience at University College London. It was during coffee breaks from lab work that Hassabis started discussing with neuroscientist Shane Legg his plans to create a company to try out his ideas. It shows the low status of AI as recently as a decade ago that they never admitted to their professors their dream to dedicate their lives to AI. But they felt they were onto something big. In September 2010, the two scientists decided to create a company with Mustafa Suleyman, who had been Hassabis's friend since childhood. And thus DeepMind was incorporated.

The company needed money, but initially Hassabis couldn't raise any capital. Pitching on a promise that they were going to solve intelligence by playing games did not sound serious to most investors. A few, however, did see the potential. Among those who put money in right at the outset were Elon Musk and Peter Thiel. Thiel had never invested outside Silicon Valley and tried to persuade Hassabis to relocate there. A born and bred Londoner, Hassabis held his ground, insisting that there was more untapped talent available in London. Hassabis remembers a bizarre conversation he had with Thiel's lawyer, who asked in all earnestness: "Does London have law on IP?" He shakes his head: "I think they thought we were coming from Timbuktu!" The founders had to give up a huge amount of stock to the investors, but they got the money to start trying to crack AI.

The challenge of creating a machine that could learn to play Go still felt like a distant dream. They set their sights at first on a less cerebral goal: playing 1980s Atari games. Atari is probably responsible for a lot of students flunking courses in the late '70s and early '80s. I certainly remember wasting a huge amount of time playing the likes of *Pong, Space Invaders,* and *Asteroids* on a friend's Atari 2600 console. The console was one of the first pieces of hardware that could play multiple games, which were loaded via cartridges. This allowed for more games to be developed over time. With previous consoles you could play only the games that had come preprogrammed into them.

One of my favorite Atari games was called *Breakout*. A wall of colored bricks appeared on the screen and your job was to destroy it completely. Your only weapons were a series of brick-destroying balls that you could send toward the bricks by using a paddle at the bottom. Moving this paddle left or right with a joystick, you attempted to intercept the balls as they bounced off the bricks and propel them back toward the wall. As you cleared the bricks—assuming you didn't lose all the balls by letting them fly past your paddle—the yellow ones in the bottom layers each scored you one point. The higher layers of red bricks got you seven points. Meanwhile, as your score rose, the balls would speed up, making the game-play progressively harder.

My friend and I were particularly pleased one afternoon when we found a clever trick to vastly improve our scores. If you dug a tunnel up through the bricks on one edge of the screen, you could get the ball up above the wall, where it would bounce rapidly up and down in that tight crawl space. You could sit back and watch one well-batted ball destroy many upper-level, high-scoring bricks before it eventually ricocheted back down through the wall. You just had to be ready with the paddle to bat the ball back up again. It was a very satisfying strategy!

Hassabis and the team he assembled also spent a lot of time playing computer games in their youth. Their parents may be happy to know that the time and effort they put into those games did not go to waste. It turned out that *Breakout* was a perfect test case to see if the team at DeepMind could program a computer to learn how to play games. It would have been a relatively straightforward job to write a program for each individual game. Hassabis and his team set themselves a much greater challenge.

They wanted to write a program that would have constant awareness of the state of the pixels on the screen and the current score and, with only those two inputs provided, could figure out how to play. The program was not told the rules of the game—only that its objective was to maximize the score. It had to experiment randomly with different ways of moving the paddle in *Breakout* or firing the laser cannon at the descending aliens of *Space Invaders*. Each time it made a move, it could assess whether that move had helped increase the score or not.

The code implements an idea dating from the 1990s called reinforcement learning, which aims to update the probability of actions based on the effect on a reward function, or score. For example, in *Breakout,* the only decision is whether to move the paddle left or right. Initially, the choice will be 50:50. But if moving the paddle randomly results in its hitting the ball, and a short time later the score goes up, the code then recalibrates based on this new information the probability of whether left or right is a better way to go. This increases the chance of heading toward where the ball is heading. The new feature was to combine this learning with neural networks that would assess the state of the pixels to determine what features were correlating to the increase in score.

At the outset, because the computer was just trying random moves, it was terrible. It barely scored anything. But each time it made a random move that bumped up the score, it would remember that move and reinforce the use of such a move in future. Gradually the random moves disappeared and a more informed set of moves began to emerge, moves that the program had learned through experiment seemed to boost its score.

It's worth watching the video the DeepMind team appended to the paper it eventually published. It shows the program learning to play *Breakout*. At first you see it randomly moving the paddle back and forth to see what will happen. Then, when a ball finally hits the paddle and bounces back and hits a brick and the score goes up, the program starts to rewrite itself. If the pixels of the ball and the pixels of the paddle connect, that seems to be a good thing. After four hundred games, it's doing really well, getting the paddle to bat balls back to the wall repeatedly.

The shock for me came with what it discovered after six hundred games. It found the same trick my friend and I had! I'm not sure how many games it took us as kids to discover it, but judging by the amount of time we wasted together, it could well have been more. But there it is. The program manipulated the paddle to ding the ball several times to the right, where it tunneled its way up the side and got trapped in the gap at the top of the screen. But while I remember my friend and I high-fiving when we discovered this trick, the machine felt nothing.

By 2014, four years after the creation of DeepMind, the program had learned how to outperform humans on twenty-nine of the forty-nine Atari games it had been exposed to. The paper the team submitted to *Nature* detailing this achievement came out in early 2015. To be published in *Nature* is one of the highlights of a scientist's career. But press and the whispers of curious bystanders. It could assume a Zen-like state of concentration wherever it was placed.

Sedol wasn't fazed by the knowledge that the machine he was up against had beaten Fan Hui. A few weeks before the match, he made this prediction to reporters: "Based on its level seen in the match [against Fan], I think I will win the game by a near landslide—at least this time." Although he was aware that AlphaGo was constantly learning and evolving, this did not concern him.

Most people still felt that, despite great inroads into programming, an AI Go champion was still a distant goal. Rémi Coulom, the creator of the only software capable of playing Go at any high standard—a program called *Crazy Stone*—was predicting that computers would not beat the best humans at the game for at least another decade.

As the date for the match approached, the team at DeepMind felt it needed someone to really stretch AlphaGo and to test it for any weaknesses. So Fan Hui was invited back to play the machine going into the last few weeks. Despite having suffered a 5–0 defeat and being mocked by the press back in China, he was keen to help out. Perhaps a bit of him felt that if he could help make AlphaGo good enough to beat Sedol, it would make his defeat less humiliating.

As Fan Hui played, he could see that AlphaGo was extremely strong in some areas. But he managed to expose a weakness that the team was not aware of. Faced with certain configurations, it seemed incapable of assessing who had control of the game, often seeming to suffer from the delusion that it was winning when the opposite was true. If Sedol exploited this weakness, AlphaGo wouldn't just lose, it would appear extremely stupid.

Members of the DeepMind team worked around the clock trying to fix this blind spot. Eventually they just had to lock down the code as it was. It was time to ship the hardware they were using to Seoul. The stage was set for a fascinating duel as the players, or at least one player, sat down on March 9 to play the first of the five games.

Beautiful, Beautiful, Beautiful

It was with a sense of existential anxiety that I fired up the YouTube channel broadcasting Sedol's matches against AlphaGo and joined 280 million other viewers to see humanity take on the machines. Having for years compared creating mathematics to playing the game of Go, I had a lot on the line.

Sedol picked up a black stone, placed it on the board, and waited for the response. Aja Huang, a member of the DeepMind team, would make the physical moves for AlphaGo. This, after all, was not a test of robotics but of artificial intelligence, so AlphaGo would still be relying on human anatomy to place the stones on the board. Huang stared at AlphaGo's screen, waiting for its response to Sedol's first stone. But nothing came.

We all stared at our screens wondering if the program had crashed. The DeepMind team was also beginning to wonder what was up. The opening moves are generally something of a formality. No human would think so long over move 2. There is nothing really to go on yet. What was happening? And then a white stone appeared on the computer screen. It had made its move. The DeepMind team breathed a huge sigh of relief. We were off! Over the next couple of hours the stones began to build up across the board.

As I watched the game it was hard for me at many points to assess who was winning. It turns out that this isn't just because I'm not a very experienced Go player. It is a characteristic of the game. And this is one of the main reasons that programming a computer to play Go is so hard. There isn't an easy way to turn the current state of the game into a robust scoring system of who leads by how much.

Chess, by contrast, is much easier to score as you play. Each piece has a different numerical value which gives you a simple first approximation of who is winning. Chess is destructive. One by one, pieces are removed so the state of the board simplifies as the game proceeds. But Go increases in complexity as you play. It is constructive. Even the commentators, although they kept up a steady stream of observations, struggled to say if anyone was in the lead right up until the final moments of the game.

What they were able to pick up quite quickly was Sedol's opening strategy. Because AlphaGo had learned to play on games that had been played in the past, Sedol was working on the principle that it would put him at an advantage if he disrupted the expectations it had built up. He was making moves that were not in the conventional repertoire. The trouble was, this required Sedol to play an unconventional game—one that was not his own.

It was a good idea, but it didn't work. Any conventional machine programmed on a database of familiar openings wouldn't have known how to respond and would likely have made a move that would have serious consequences in the grand arc of the game. But AlphaGo was not a conventional machine. As David Silver, its lead programmer, explained in the run-up to the match, "AlphaGo learned to discover new strategies for itself, by playing millions of games between its neural networks, against themselves, and gradually improving." If anything, Sedol had put himself at a disadvantage.

As I watched I couldn't help feeling for Sedol. You could see his confidence draining out of him as it gradually dawned on him that he was losing. He kept looking over at Huang, the DeepMind representative who was executing AlphaGo's moves, but there was nothing he could glean from Huang's face. By move 186, Sedol realized there was no way to overturn the advantage AlphaGo had built up on the board. He placed a stone on the side of the board to indicate his resignation.

By the end of day one it was AlphaGo 1, Humans 0. Sedol made an admission at the press conference that day: "I was very surprised, because I didn't think I would lose."

But it was Game Two that would truly shock not just Sedol but every human player of Go. In the first game, experts could follow the logic and appreciate why AlphaGo was playing the moves it was. They were moves a human champion would play. But in Game Two, something rather strange happened. Sedol had just played move 36 and then retired to the roof of the hotel for a cigarette break. While he was away, AlphaGo instructed Huang, its human representative, to execute move 37, placing one of its black stones on an intersection five lines in from the edge of the board. Everyone was shocked.

The conventional wisdom is that during the early part of the game you focus on the outer four lines. Stones placed on the third line build up short-term territory strength at the edge of the board while playing on the fourth line contributes to your strength later in the game as you move into the center of the board. Players have always found that there is a fine balance between playing on the third and fourth lines. Playing on the fifth line has always been regarded as suboptimal, giving your opponent the chance to build up territory that has both short- and long-term influence.

AlphaGo had defied this orthodoxy built up over centuries of competing. Some commentators declared it a clear mistake. Others were more cautious. Everyone was intrigued to see what Sedol would make of the move when he returned from his cigarette break. Even watching on my laptop at home, I could see him flinch as he took in the new stone on the board. He was certainly as shocked as all of the rest of us by the move. He sat there thinking for over twelve minutes. As in chess competitions, the game was being played under time constraints. Using twelve minutes of his time was very costly. It is a mark of how surprising this move was that it took Sedol so long to respond. He could not understand what AlphaGo was doing. Why had the program abandoned the region of stones they were competing over?

Was this a mistake by AlphaGo? Or did it see something deep inside the game that humans were missing? Fan Hui, who was serving as one of the referees, looked down on the board. His initial reaction matched everyone else's: shock. But then he paused to appreciate it. *Wired* magazine reports how he would describe the moment later:

> "It's not a human move. I've never seen a human play this move," says spectator and Go champion Fan Hui. "So beautiful." It's a word he keeps repeating. Beautiful. Beautiful. Beautiful.

Beautiful and deadly it turned out to be. Not a mistake but an extraordinarily insightful move. Some fifty moves later, as the black and white stones fought over territory in the lower left corner of the board, they found themselves creeping toward the black stone of move 37. It was joining up with this stone that gave AlphaGo the edge, allowing it to clock up its second win. AlphaGo 2, Humans 0.

Sedol's mood in the press conference that followed was notably different. "Yesterday I was surprised. But today I am speechless . . . I am in shock. I can admit that . . . the third game is not going to be easy for me." The match was being played over five games. This was a game that Sedol needed to win to have any hope of keeping AlphaGo from claiming the match.

The Human Fights Back

Sedol had a day off to recover. The third game would be played on Saturday, March 12. He needed the rest, unlike the machine. The first game had required more than three hours of intense concentration. The second lasted over four hours. Spectators could see the emotional toll that losing two games in a row was having on him.

Rather than resting, though, Sedol stayed up till six o'clock the next morning analyzing the games he'd lost so far with a group of fellow professional Go players. Did AlphaGo have a weakness they could exploit? The machine wasn't the only one who could learn and evolve. Sedol felt he might learn something from his losses.

Sedol played a very strong opening to Game Three, forcing AlphaGo to manage a weak group of stones within his sphere of influence on the board. Commentators began to get excited. Some said Sedol had found AlphaGo's weakness. But then, as one commentator, David Ormerod, posted, "things began to get scary. . . . As I watched the game unfold and the realization of what was happening dawned on me, I felt physically unwell."

Sedol pushed AlphaGo to its limits, but in doing so he seemed to invoke some hidden powers the program possessed. As the game pro-

This apparently is another characteristic behavior of Go algorithms. Once they see that they are losing they go rather crazy.

Silver winced as he saw the next move AlphaGo was opting for: "I think they're going to laugh." Sure enough, the Korean commentators collapsed into fits of giggles at the moves AlphaGo was now making. Its moves were failing the Turing Test. No human with a shred of strategic sense would make such moves. The game dragged on for a total of 180 moves, at which point AlphaGo put up a message on the screen that it had resigned.

The human race had got one back. AlphaGo 3, Humans 1. The smile on Lee Sedol's face at the press conference that evening said it all. "This win is so valuable that I wouldn't exchange it for anything in the world." The press room erupted with joyful applause. "It's because of the cheers and the encouragement that you all have shown me."

Gu Li, who was commentating the game in China, declared Sedol's move 78 as the "hand of God." It was a move that broke the conventional way to play the game and that was ultimately the key to its shocking impact. Yet this is characteristic of true human creativity. It is a good example of Boden's transformational creativity, where people break out of the system to find new insights.

At the press conference, Hassabis and Silver, his chief programmer, could not explain why AlphaGo had lost. They would need to go back and analyze why it had made such a lousy move in response to Sedol's move 78. It turned out that AlphaGo's experience in playing humans had led it to totally dismiss such a move as something not worth thinking about. It had assessed that this was a move that had only a one-in-ten-thousand chance of being played. It seems as if it had just not bothered to learn a response to such a move because it had prioritized other moves as more likely and therefore more worthy of response.

Perhaps Sedol just needed to get to know his opponent. Perhaps over a longer match he would have turned the tables on AlphaGo. Could he maintain the momentum into the fifth and final game? Losing three games to two would be very different from losing four to one. The last win was still worth fighting for. If he could win a second game, it would sow seeds of doubt about whether AlphaGo could sustain its superiority.

But AlphaGo had learned something valuable from its loss. The next person who plays Sedol's one-in-ten-thousand move against the algorithm won't get away with it. That's the power of this sort of algorithm. It never forgets what it learns from its mistakes.

That's not to say it can't make new mistakes. As Game Five proceeded, there was a moment quite early in the game when AlphaGo seemed to completely miss a standard set of moves in response to a particular configuration that was building. As Hassabis tweeted from backstage, "#AlphaGo made a bad mistake early in the game (it didn't know a known tesuji) but now it is trying hard to claw it back . . . nail-biting."

Sedol was in the lead at this stage. It was game-on. Gradually AlphaGo did claw back. But right up to the end the DeepMind team was not exactly sure whether it was winning. Finally, on move 281—after five hours of play—Sedol resigned. This time there were cheers backstage. Hassabis punched the air. Team members hugged and high-fived. The win that Sedol had pulled off in Game Four had evidently reengaged their competitive spirit. It was important for them not to lose this last game.

Looking back at the match, many recognize what an extraordinary moment this was. Some immediately commented on its being an inflection point for AI. Sure, all this machine could do was play a board game—and yet for those looking on, its capability to learn and adapt was something quite new. Hassabis's tweet after winning the first game summed up the achievement: "#AlphaGo WINS!!!! We landed it on the moon."

It was a good comparison. Landing on the moon did not yield extraordinary new insights about the universe, but the technology that humanity developed to achieve such a feat did. Following the last game, AlphaGo was awarded an honorary nine-dan professional ranking by the Korean Go Association, the highest accolade for a Go player.

From Hilltop to Mountain Peak

Move 37 of Game Two was a truly creative act. It was novel, certainly, it caused surprise, and as the game evolved it proved its value. This was exploratory creativity, pushing the limits of the game to the extreme.

One of the important points about the game of Go is that there is an objective way to test whether a novel move has value. Anyone can come up with a new move that appears creative. The art and challenge is making a novel move that has some sort of value. How should we assess value? It can be very subjective and time-dependent. Something that is panned critically at the time of its release can be recognized generations later as a transformative creative act. Nineteenth-century audiences didn't know what to make of Beethoven's *Symphony No. 5*, and yet it is central repertoire now. During his lifetime, Van Gogh could barely sell his paintings—he traded them for food or painting materials—but now they go for millions. In Go, there is a more tangible and immediate test of value: Does it help you win the game? Move 37 won Game Two for AlphaGo. There was an objective measure that we could use to value the novelty of this move.

AlphaGo had taught the world a new way to play an ancient game. Analysis since the match has resulted in new tactics. The fifth line is now played early on, as we have come to understand that it can have big implications for the end game. AlphaGo has gone on to discover still more innovative strategies. DeepMind revealed at the beginning of 2017 that its latest iteration had played online anonymously against a range of top-ranking professionals under two pseudonyms: Master and Magister. Human players were unaware that they were playing a machine. Over a few weeks it had played a total of sixty complete games. It won all sixty.

But it was the analysis of the games that was truly eye-opening. Those games are now regarded as a treasure trove of new ideas. In several games AlphaGo played moves that any beginners, if they made them, would have their wrists slapped for by their Go masters. Traditionally, for example, you do not play a stone at the intersection of the third column and third row. And yet AlphaGo showed how to use such a move to great advantage.

Hassabis describes how the game of Go had got stuck on what mathematicians like to call a local maximum. Look at the landscape illustrated below and imagine you are at the top of the peak to the left. From this height there is nowhere higher to go. This is called a local maximum. If there were fog all around you, you'd think you were at the highest point in the land. But across the valley is a higher peak. To know this, you need the fog to clear. Then you need to descend from your peak, cross the valley, and climb to the top of it.

The trouble with modern Go is that conventions had built up about ways to play that had ensured players hit Peak A. But by breaking those conventions AlphaGo had cleared the fog and revealed an even higher Peak B. It's even possible to measure the difference. In Go, a player using the conventions of Peak A will in general lose by two stones to the player using the new strategies discovered by AlphaGo.

This rewriting of the conventions of how to play Go has happened at two previous points in history. The most recent was the innovative game-play introduced by the legendary player Go Seigen in the 1930s. His experimentation with ways of playing the opening moves revolutionized the way the game is played. But Go players now recognize that AlphaGo might well have launched an even greater revolution.

Chinese Go champion Ke Jie recognizes that we are in a new era: "Humanity has played Go for thousands of years, and yet, as AI has shown us, we have not yet even scratched the surface. The union of human and computer players will usher in a new era."

Ke Jie's compatriot Gu Li, winner of the most Go world titles, added: "Together, humans and AI will soon uncover the deeper mysteries of Go." For Hassabis, the algorithm is like the Hubble telescope of Go. This illustrates the way many view this new AI. It is a tool for exploring deeper, further, wider than ever before. It is not meant to replace human creativity but to augment it.

And yet there is something that I find quite depressing about this moment. It feels almost pointless to want to aspire to be the world champion at Go when you know there is a machine that you will never be able to beat. Professional Go players have tried to put a brave face on it, talking about the extra creativity that it has unleashed in their own play, but there is something quite soul-destroying about knowing that we are now second-best to the machine. Sure, the machine was programmed by humans, but that doesn't really make it feel better.

AlphaGo has since retired from competitive play. The Go team at DeepMind has been disbanded. Hassabis proved his Cambridge lecturer wrong. DeepMind has now set its heights on other goals: health care, climate change, energy efficiency, speech recognition and generation, computer vision. It's all getting very serious.

Given that Go had always been my shield against computers doing mathematics, was my own subject next in DeepMind's cross-hairs? To truly judge the potential of this new AI, we'll have to look more closely at how it works and dig around inside. The ironic thing is that the tools DeepMind is using to create the programs that might put me out of a job are precisely the ones that mathematicians have created over the centuries. Is this mathematical Frankenstein's monster about to turn on its creator?

If M = 36 and N = 15, then dividing M by N gives you 2 with a remainder (N_1) of 6. Dividing N by N_1, we get 2 with a remainder (N_2) of 3. But now, dividing N_1 by N_2, we get 2 with no remainder at all, so we know that 3 is the largest number that can divide both 36 and 15.

You see that there are lots of "if . . . then . . ." clauses in this process. That is typical of an algorithm and is what makes algorithms so perfect for coding and computers. Euclid's ancient recipe exhibits the four key characteristics any algorithm should ideally possess:

It consists of a precisely stated and unambiguous set of instructions.
Its procedure always comes to a finish, regardless of the numbers inserted. (It does not enter an infinite loop!)
It produces the answer for any values input.
It is fast.

In the case of Euclid's algorithm, there is no ambiguity at any stage. Because the remainder grows smaller at every step, after a finite number of steps it must hit zero, at which point the algorithm stops and spits out the answer. The bigger the numbers, the longer the algorithm will take, but it's still relatively fast. (The number of steps is five times the number of digits in the smaller of the two numbers, for those who are curious.)

If the invention of the algorithm happened over two thousand years ago, why does it owe its name to a ninth-century Persian mathematician? *Algorithmi* is the Latinized form of a surname—that of Muḥammad ibn Mūsā al-Khwārizmī. One of the first directors of the great "House of Wisdom" in Baghdad, al-Khwārizmī was responsible for many of the translations of the ancient Greek mathematical texts into Arabic. Although all the instructions for Euclid's algorithm are there in the *Elements,* the language Euclid used was very clumsy. The ancient Greeks thought about mathematic problems geometrically, so numbers were presented as lines of different lengths and proofs consisted of pictures—a bit like our example of tiling the floor. But pictures aren't sufficient for doing mathematics with much rigor. For that, you need the language of algebra, which uses letters to stand for variable numbers. This was the invention of al-Khwārizmī.

To be able to articulate the workings of an algorithm, we need language that allows us to talk about numbers without specifying what those numbers are. We already saw this at work in Euclid's algorithm, when we gave names to the numbers we were trying to analyze: N and M. These letters can represent any numbers. The power of this new, linguistic take on mathematics was that it allowed mathematicians to understand the grammar underlying how numbers work. Rather than being limited to showing particular examples of a method working, this new language of algebra provided a way to explain the general patterns behind the behavior of numbers. Today's easy analogy is to think of the code behind a running software program. No matter what numbers are plugged in as inputs, it works to yield an output—the third criterion in our conditions for a good algorithm.

Indeed, algorithms have gained enormous currency in our era precisely because they are perfect fodder for computers. Wherever there is a discernible pattern underlying the way we solve a problem to guide us to a solution, an algorithm can exploit that discovery. It is not required of the computer that it think. It need only execute the steps encoded in the algorithm and, again and again, as if by magic, out pop the answers we seek.

Desert Island Algorithm

One of the most extraordinary algorithms of the modern age is the one that helps millions of us navigate the internet every day. If I were exiled to a desert island and could take only one algorithm with me, I'd probably choose the one that drives Google (although perhaps I would check first whether in exile I would still have an internet connection).

In the early days of the World Wide Web (we're talking the early 1990s) there was a directory of all the existing websites. In 1994 there were only three thousand of them. The list was small enough that you could pretty easily thumb through it and find a site that someone had mentioned to you. Things have changed quite a bit since then. When I started writing this paragraph there were 1,267,084,131 websites live on the internet. A few sentences later, that number has gone up to 1,267,085,440. (You can check the current status at www.internetlive stats.com.)

How does Google's search engine figure out exactly which ones of these billion-plus websites to recommend? Most users have no idea. Mary Ashwood, for example, an eighty-six-year-old granny from Wigan, in northeast England, was careful to add a courteous "please" and "thank you" to each query, perhaps imagining an industrious group of interns on the other end sifting through the endless requests. When her grandson Ben opened her laptop and found "Please translate these roman numerals mcmxcviii thank you," he couldn't resist posting a snapshot on Twitter to share his nan's misconception with the world. He got a shock when someone at Google UK tweeted back:

> Dearest Ben's Nan.
> Hope you're well.
> In a world of billions of Searches, yours made us smile.
> Oh, and it's 1998.
> Thank YOU.

Ben's Nan brought out the human in Google on this occasion, but there is no way any company could respond personally to the million searches Google receives every fifteen seconds. So if it isn't magic Google elves scouring the internet, how does Google succeed in so spectacularly locating the information you want?

It all comes down to the power and beauty of the algorithm Larry Page and Sergey Brin cooked up in their dorm rooms at Stanford in 1996. They originally named their new search engine "BackRub" (referring to its reliance on the web's "back links") but by 1997 switched to "Google," inspired by the name a mathematician in the 1930s gave