# Chapter 1: A Brief History of Thinking Machines

*“I propose to consider the question, ‘Can machines think?’”* — Alan Turing, 1950

In the spring of 1950, British mathematician Alan Turing sat down to write a scientific paper that began with a question he immediately decided to abandon. Turing had spent the war breaking German naval codes. He was also the first one to deduce a workable architecture for a modern computer. "Can machines think?[1]" was his opening salvo, followed almost at once by his conclusion that the question was not worth asking in its current form.

Perhaps "think" was too slippery a word for Turing’s mathematical precision. If you ask ten people what “thinking” means, you might receive ten different answers, each shaped by the responder's own particular intuition about what made human minds special. Turing wanted a question with a verifiable answer, so he reframed the problem as a game.

He called it the Imitation Game, a thought experiment now known as the **Turing Test**. He described a setup with three rooms: an interrogator with a keyboard and a screen in one, a human being and a machine in the other two. The interrogator types questions; both the human and the machine type back answers. The interrogator's job is to figure out which is which. Turing proposed that if a machine could fool an interrogator often enough, we would have no reasonable basis for denying that it was doing something worth calling thought, whatever that word turned out to mean.

The proposal was characteristically oblique. Rather than defining intelligence, Turing had described a performance of it. Rather than asking what thinking *is*, he asked what thinking *looks like from the outside*. This was either a stroke of pragmatic genius or a very elegant evasion, depending on your viewpoint. But it established a pattern that would repeat for the rest of the century: each generation would define intelligence by whatever seemed hardest to replicate, and when machines replicated it, the definition would shift.

## Thought as Logic

Six years after Turing's paper, in the summer of 1956, a two-month workshop was convened at Dartmouth College in Hanover, New Hampshire. The proposal[2] for it had been circulated the previous year by four researchers: John McCarthy, Marvin Minsky, Claude Shannon, and Nathaniel Rochester. McCarthy was a mathematician at Dartmouth who would go on to coin the term '**artificial intelligence**.' Minsky would become the field's most influential theorist. Shannon,

---

[1] Turing, A. M. "Computing Machinery and Intelligence." *Mind* 49 (1950): 433–460.
[2] McCarthy, John, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence." August 31, 1955.

at Bell Labs, had essentially founded the modern science of information. Rochester, at IBM, had helped design the company's first mass-produced scientific computer.

The proposal's central claim was that every aspect of learning, and every other feature of intelligence, could in principle be described precisely enough for a machine to simulate it. This was not wishful thinking: in the preceding two decades, mathematicians and logicians working across Europe and America had produced a series of results that suggested the mind might be, at its core, a kind of logical engine. They had shown that all of arithmetic could be derived from a handful of axioms using a few simple rules. Turing himself had proven that any computation that could be precisely described could be carried out by a machine following a handful of simple rules. If thought was something like computation, and computation was something like logic, then a machine that could manipulate logical symbols according to formal rules would, given a sufficiently rich set of rules, think.

Allen Newell, a researcher at the RAND Corporation in Santa Monica, and Herbert Simon, an economist and psychologist at Carnegie Mellon University in Pittsburgh who would later win the Nobel Prize in Economics, took this seriously. In 1956, the same year as the Dartmouth workshop, they demonstrated a program called the Logic Theorist, which could prove theorems in formal logic by searching selectively through the space of possibilities rather than trying every possible proof, choosing moves that seemed to make progress. It proved thirty-eight of the first fifty-two theorems in *Principia Mathematica*, a foundational work of mathematical logic by Alfred North Whitehead and Bertrand Russell that had attempted to derive all of mathematics from a small set of logical axioms. One of its proofs was even more elegant than the original. Simon wrote to a colleague that they had "solved the venerable mind-body problem." The mind-body problem is the philosophical question of how a physical brain gives rise to something as apparently non-physical as thought, and it has resisted resolution for centuries. Simon's answer was to reframe the question entirely: if thinking was symbol manipulation, and symbol manipulation could run on any physical substrate, then the mystery dissolved into engineering.

Two years later, Newell and Simon built something more ambitious, a program they called the General Problem Solver, designed not for any particular domain but for the activity of solving problems in general, whatever form those problems might take. It would represent the current state of affairs, represent a goal, identify the gap between them, and select operations to close it. The design was meant to mirror the structure of human cognition, which Newell and Simon were studying simultaneously through experiments in which people spoke aloud while solving puzzles. The claim was that the program and the person were doing the same thing.

By 1976, Newell and Simon had formalized this into what they called the Physical Symbol System Hypothesis. A physical symbol system, meaning any device that creates, stores, copies, and manipulates symbols according to rules, has, they argued, everything it needs to produce general intelligent action. The brain, in this view, was one such system. A digital computer was another. They were built from different materials but ran on the same principle.

The systems that followed from this conviction were, within their domains, genuinely impressive. Programs like MYCIN, developed at Stanford in the early 1970s, could diagnose bacterial

infections and recommend antibiotic treatments. The system worked the way a textbook doctor works: if the patient has this symptom, combined with that lab result and this clinical sign, then consider these organisms, in this order of likelihood. The rules were painstakingly extracted from expert physicians through long interviews, then encoded, tested, and refined. MYCIN performed at least as well as many specialists, and better than most general practitioners.

But MYCIN knew only about infections. Move it to a neighboring domain, say antiviral therapy, and it was useless. It would have to be rebuilt from scratch. Every rule it contained had been placed there by a human. It could not add a rule on its own, could not notice patterns in the cases it had seen, could not learn. If the world changed, the program did not. Whoever maintained it had to read the literature, interview new experts, and update the rules by hand. The bottleneck was not computing power. It was the sheer difficulty of translating human knowledge, which is intuitive, embedded in experience, and often impossible to fully articulate, into the kind of explicit logical form a symbol system could use.

The systems worked, but only inside the boundaries their creators had drawn. Step outside those boundaries and the cracks began to show.

## Thought as Search

Chess attracted enormous attention from early researchers for a reason that now looks almost too obvious. The game had the properties researchers needed: complete information, perfect rules, a clear definition of winning. Every possible position could, in principle, be evaluated. The catch was that "in principle" concealed a practical impossibility. The number of possible chess games exceeds the number of atoms in the observable universe by a margin that makes the phrase "vastly more" inadequate. No machine would ever search every branch of every tree. Something had to be cut.

Simon had already argued, from his work on the Logic Theorist, that the human mind did not search everywhere. Instead, it searched selectively, guided by experience about where the promising paths tended to lie. Chess made exactly the same demand. A grandmaster does not evaluate millions of positions; she looks at a handful of candidates, recognizes patterns, prunes the implausible. If a machine could learn to do something similar, using rules of thumb rather than grinding through every possibility, then search might be the key to intelligence.

This insight spread well beyond chess. **Expert systems**, which dominated the practical landscape through the 1970s and 1980s, were essentially searches through a space of rules: given a problem, find the rules that apply, apply them, and see where you end up. The systems worked in domains where someone had taken the trouble to map the space in advance. A medical diagnosis program knew which symptoms pointed to which diseases because its creators had encoded that knowledge. A geological survey system could identify mineral deposits because petroleum engineers had specified the relevant indicators.

The brittleness of these systems had a single cause: the knowledge was static. It had been frozen at the moment of construction. If a disease changed its presentation, if new evidence altered standard practice, if a scenario arose that the original experts had not anticipated, the system had no mechanism to update itself. Someone had to open the hood and modify the rules.

There was a deeper problem, too. The more complex the system became, the harder it was to maintain. Rules began to contradict each other and edge cases multiplied. The engineers who had built the system could not always predict how thousands of rules would interact in novel situations. At some point, the system became too complicated for anyone to fully understand, which meant that debugging it required the kind of intuition the system itself was supposed to be replacing.

The world was too large and too changeable to fit inside a rulebook.

## Thought as Uncertainty

What the symbolic approach had assumed was that knowledge comes in the form of definite facts: the patient has a fever, or does not. The infection is bacterial, or it is not.

Real reasoning, argued Judea Pearl, a computer scientist at UCLA who would later win the Turing Award, the highest honor in computer science for this very work, is not about certainties. It is about degrees of confidence under incomplete information.

A doctor examining a patient does not wait for certainty before forming a view. She considers what she knows, age, symptoms, history, lab results, and asks, given all of this, what is most likely going on? When new information arrives, she updates. A test comes back positive; her estimate of one diagnosis rises, another falls. She does not start over. She adjusts, continually, on the basis of accumulating evidence.

Pearl built mathematical frameworks that allowed computers to do something similar: represent knowledge not as true-or-false facts but as probabilities, and reason about how new observations should shift those probabilities. A system built this way could say not "the patient has infection X" but "the patient has infection X with 73 percent probability given these findings, rising to 89 percent if this additional test comes back positive." It handled uncertainty gracefully because uncertainty was built into its foundations rather than treated as a problem to be eliminated.

The systems that resulted were more honest about what they knew and more resilient when the world failed to match expectations. They could acknowledge ambiguity rather than producing bad results when inputs did not fit the rules.

But the probabilities themselves still had to come from somewhere. The structure of the model, which variables mattered, which caused which, which could be safely ignored, had to be specified in advance by human designers. A probabilistic system reasoned from its model; it did

not build the model. The engineers had moved from encoding rules to encoding probabilistic relationships, which was genuine progress, but the knowledge still arrived from outside. The machine updated what it believed; it did not learn what to believe in the first place.

For that, a different kind of system was required.

## Thought as Learning

The first thing to understand about how a neural network learns is that it learns by being wrong.

Consider a simple version of the problem. You want a system that can recognize handwritten digits. You show it an image of a handwritten "7" and the system produces a guess. Say it guesses "1." That is wrong. The system is then given a signal indicating that it was wrong and, critically, how wrong. Something inside it adjusts, not dramatically, just a small nudge distributed across thousands of tiny internal connections, each becoming fractionally more or less likely to fire in certain combinations. Then you show it another image. Another guess, and another adjustment. You do this millions of times.

What the system is doing, at a mechanical level, is something like tuning a vast number of dials simultaneously, where no single dial controls the answer and the answer emerges only from the combined configuration of all of them together. After enough tuning, on enough examples, the configuration somehow captures something about what makes a 7 look like a 7, what makes a 1 look like a 1. The programmer did not specify any of those features. They emerged through iteration, from the accumulated pressure of millions of corrections. This process of repeated correction is called **training**.

The technique that makes it work is called **backpropagation**. The key insight is that the **error signal**, the measure of how wrong the system was, can be used not just to adjust the final output but to calculate how much each internal connection contributed to the error, working backward through the network layer by layer. Each connection gets nudged in whatever direction would have made the error slightly smaller. Do this enough times and the whole network reshapes itself around the structure of the problem. The technique was popularized by Geoffrey Hinton, a British-Canadian computer scientist who would eventually share the Nobel Prize in Physics for his foundational work on neural networks, together with David Rumelhart and Ronald Williams. Rumelhart and Williams were at UC San Diego; Hinton was at Carnegie Mellon, though he later moved to the University of Toronto, where much of his subsequent work took place.

For two decades, the approach remained more theoretically interesting than practically transformative. Computers were not fast enough and datasets were not large enough. Then, in 2012, a team at the University of Toronto, Krizhevsky, Sutskever, and Hinton, entered a major image classification competition with a network substantially deeper than anything previously attempted. The competition asked systems to correctly classify photographs from more than a million images across a thousand categories: dogs, airplanes, mushrooms, electric guitars.

The Toronto network correctly identified the subject of an image within its top five guesses, the standard measure for the competition, roughly 85 out of every 100 times. That may not sound impressive until you consider that the next best system in the same competition was getting it right, by the same measure, only 74 times out of a hundred. The gap was not the kind of modest improvement the field had grown accustomed to. It was large enough that researchers understood immediately that something had changed, not just in the performance of one system, but in the direction the entire field would take.

Within a year, almost every research group in the field had abandoned what they had been doing and switched to the same approach. Networks trained on images learned to identify tumors in medical scans, read lips, generate realistic photographs. Networks trained on text learned to translate between languages, answer questions about documents they had just been shown, write code. What had changed was not just the technique. It was the relationship between the system and the world: instead of a designer installing knowledge from the outside, the system was acquiring it from the inside, through exposure and accumulated correction.

These systems were not discovering facts about the world so much as finding patterns in human-produced data, reliable relationships between inputs and outputs extracted from examples that humans had labeled or text that humans had written. A translation system trained on millions of parallel texts in English and French did not understand either language in any philosophically loaded sense. It had instead learned, with extraordinary precision, that certain arrangements of symbols in one language correspond to certain arrangements in another. The correspondence was real enough to be useful, and for most purposes, useful was enough.

## Thought as Representation

In 2017, a group of researchers at Google published a paper describing a new architecture for processing language. The **Transformer**, as they called it, solved a problem that had hobbled previous approaches. Earlier systems processed text sequentially, word by word, which meant that by the time the system reached the end of a long sentence, its representation of the beginning had faded. This is not how reading works, and it is not how understanding long passages works, and it was producing real limitations in what the systems could do.

The Transformer solved this by computing, in a single operation, the relationship between every word in a passage and every other word. Think of it as the difference between reading a sentence once, from left to right, and spreading the whole sentence out on a table and examining all the connections simultaneously. "Bank" means something different in a sentence about rivers than in a sentence about finance. The Transformer could figure that out by attending to every other word in the passage at once, using each context clue in parallel rather than having to reconstruct them from memory.

This allowed systems to build richer representations of meaning, not just knowing that a word is present, but encoding something about what it means here, in this passage, surrounded by

these particular other words. At a small scale, the improvement was incremental. At large scale, it became transformative.

What researchers discovered, beginning to chart the relationship between scale and performance, was that more data and larger models did not just improve results linearly. The improvements followed surprisingly orderly mathematical patterns across many orders of magnitude: double the training data, get a predictable gain; scale the model by a factor of ten, get another. The gains did not plateau as quickly as anyone had expected. They kept arriving, and as they did, something else happened that had not been planned.

Systems trained purely to predict the next word in a sequence began doing things no one had programmed. A system trained on text, just text, with no explicit instruction to do anything other than predict what word comes next, could, when prompted appropriately, write coherent essays, solve mathematical problems it had never seen, explain scientific concepts, generate working software. No one had taught it mathematics. No one had taught it to code. These capabilities had not been designed; they had shown up uninvited, as apparent side effects of learning the statistical structure of language at sufficient scale. Researchers called them **emergent** capabilities, and the term carried a frank admission: no one had predicted them. These were behaviors that emerged, apparently, from the sheer volume and density of structure in the training data.

Nobody could say with confidence what the system had actually learned. Something had been captured: something about how reasoning is expressed in language, how arguments are structured, how explanations proceed. But whether that amounted to understanding, or an extraordinarily sophisticated form of pattern completion, remained an open question, and it still does.

## The Missing Body

There was, running alongside these developments, a separate line of research that asked a different question entirely. If the mainstream of the field was concerned with how machines think, this alternative tradition was concerned with how machines act, and whether the distinction matters.

In the late 1980s, Rodney Brooks at MIT made an argument that most of his colleagues did not want to hear: the symbolic tradition, the whole project of building intelligence out of internal models and logical rules, was built on a mistake. Researchers had spent decades trying to give machines rich internal representations of the world, elaborate symbolic models that the machine could reason about, plan with, update. This approach had produced systems that could manipulate abstractions inside a computer but that fell apart the moment they had to deal with a physical environment. The problem, Brooks contended, was not that the representations were insufficiently detailed. It was that the representations were the wrong idea altogether.

Brooks pointed to an evolutionary argument. Single-celled organisms appeared on Earth roughly 3.5 billion years ago. It took another three billion years for anything as complex as an insect to

evolve. Insects can navigate complex terrain, avoid predators, find food, build nests, and coordinate with other members of their species, all without anything resembling the kind of central model-of-the-world that researchers were trying to build into their programs. Expert knowledge, language, abstract reasoning: these arrived only in the last sliver of evolutionary history. The hard problem, the one evolution had spent most of its time solving, was not thinking. It was moving around in a dynamic world, sensing what mattered, and reacting in time for it to count.

Brooks built robots to prove the point. Rather than constructing a single complex program that perceived the world, built an internal model, planned a course of action, and then executed it, he decomposed behavior into layers. The lowest layer did nothing but avoid obstacles: if the robot's sensors detected something nearby, it moved away. A second layer, running simultaneously, made the robot wander, injecting a random heading every few seconds. A third layer looked for distant open spaces and steered toward them. No layer knew about the others. There was no central representation, no master plan, no unified picture of the world.

And yet the robot navigated crowded offices, dodged people walking by, and explored its environment for hours on end. The behavior that emerged from these simple interacting layers looked, to an observer, purposeful and adaptive. It looked, one might say, intelligent. Brooks called his design the **subsumption architecture**[3], and distilled its philosophy into a single line: "The world is its own best model." There was no need to build an internal replica of reality when reality was right there, available through the robot's sensors at every moment.

The implication challenged the entire symbolic tradition. If an insect-level creature could navigate the world without representations, perhaps the elaborate model-building that researchers had been pursuing was not the foundation of intelligence but a late and optional addition to it. Perhaps the researchers had spent decades working on the penthouse while ignoring the foundation.

Brooks was not alone in this line of thinking, though he was its most provocative voice. Across robotics, researchers were arriving at a similar conclusion: the problems the field had treated as peripheral were in fact the hard ones. Perception, motor control, real-time response to a changing environment: these were what evolution had spent billions of years solving. Logical reasoning, planning, symbol manipulation, the things the field had placed at the center, might turn out to be the easy part, or at least the part that only became possible once the harder problems had been solved.

## Intelligence on Wheels

The most visible test of this idea came not in a laboratory but in the Mojave Desert.

In 2004, the Defense Advanced Research Projects Agency (DARPA) offered a million-dollar prize to any team that could build a vehicle capable of driving itself 142 miles through the desert

[3] Brooks, Rodney A. “Intelligence without Representation.” *Artificial Intelligence* **47**, nos. 1–3 (1991): 139–159.

without human intervention. A hundred and seven teams registered, but only fifteen raced. The best-performing vehicle traveled less than eight miles before getting stuck on an embankment and catching fire.

DARPA doubled the prize and ran the race again in 2005. This time, five of twenty-three vehicles finished the course. The winner was Stanley, a modified Volkswagen Touareg built by a team at Stanford led by Sebastian Thrun, a computer scientist who had spent his career at the intersection of robotics and machine learning.

Stanley was not programmed with a map of every rock and rut in the desert. Its designers could not have provided one; the specific course was kept secret until two hours before the race. Instead, the vehicle carried laser range finders and a camera, and its software had to make sense of what the sensors reported in real time. Dirt roads look different in morning light than in afternoon shadow. A dip in the terrain can make the sensors report phantom obstacles where there are none. Sand, rocks, brush, and fence posts all have to be classified correctly, at speed, without second chances.

What made Stanley distinctive was that it learned. Rather than hand-coding rules for every kind of obstacle, the team built systems that learned to distinguish drivable terrain from hazards by observing examples. A human drove the vehicle through representative terrain, and the system recorded what the road looked like through its sensors. The machine learned, statistically, what "road" looked like and what "not road" looked like, and it kept learning as conditions changed. When the surface shifted from pavement to gravel, the system adapted within seconds. When the lighting changed, it adjusted.

Stanley finished the course in just under seven hours. It was not flawless, but it recovered from every glitch, and the race data confirmed that a vehicle following GPS waypoints blindly, without perceiving and responding to the actual terrain, would almost certainly have failed in the narrow mountain passes where the road deviated from the coordinates. Stanley's competence did not reside in a fixed set of rules or a pre-built model of the desert. It emerged from continuous interaction between the vehicle's sensors and the terrain passing beneath its wheels. The system was adjusting to the world as it encountered it, not executing a plan laid down in advance.

## The Moving Target

Looking back, the progression through these eras can seem like a series of false starts. Viewed altogether, it reveals something more interesting. Each time a new approach succeeded, it did so not because of a better theory of the mind but because the system had been given more contact with the actual texture of things rather than an engineer's abstraction of them.

The researchers who built the Logic Theorist were not wrong to think reasoning looked like symbol manipulation. For a narrow class of problems, it does. But each generation inherited a conviction about what intelligence fundamentally was, built systems around that conviction, and eventually ran into the same wall: the world was messier than the theory. In the 1950s, the

conviction was that thought was logic, which made sense given that the most rigorous thing humans did with their minds was mathematics. By the 1960s and 1970s, the focus had shifted to search, because when researchers actually watched people solve problems, what they saw looked less like formal proof and more like navigation: trying one path, abandoning it, trying another. In the 1980s, that picture gave way to probability, the recognition that real reasoning rarely operates on certainties but on degrees of confidence that shift as evidence comes in. Then statistical pattern recognition, because it turned out that exposure to enough examples could produce something that looked remarkably like understanding, without anyone having to specify the rules. By the 2010s, scale had become the dominant idea: given enough data and a large enough model, structure in the world would somehow become structure in the system. And running beneath it all, the embodied tradition had been arguing that the hardest part of intelligence was not thinking at all but acting, that a robot adjusting to the world through its sensors was doing something no amount of abstract reasoning could replicate.

Each shift, in its moment, looked like a solution. A new theory displaced an old one, funding followed, and the promises seemed within reach until they weren't. It happened twice, badly enough that researchers named the collapses: AI winters, arriving each time after overconfidence met the complexity of the actual world.

What the progression reveals is less a story of converging on the truth than a story of the truth proving more complicated than anyone had anticipated. The symbolic systems worked best when their designers could transfer, through laborious engineering, the accumulated experience of experts into explicit rules. Probabilistic systems worked better when they could use data to estimate relationships rather than requiring experts to hand-specify them. Learning systems worked because they could process more examples, in more varied forms, than any designer could encode. And the embodied systems demonstrated that some knowledge cannot be computed in advance and then applied; it exists in the continuous back-and-forth between a body and a world that does not stop to wait.

Looked at this way, the history of building a thinking machine is not a sequence of competing theories about the nature of intelligence. It is a gradual, reluctant recognition that intelligence is less like a faculty residing in a system and more like something that accretes through exposure, emerging from contact with a world that is variable, constraint-rich, and full of the consequences of being wrong.

## The Limit of the Mirror

The most capable systems built today have been trained on the full corpus of human-created data: text, images, code, sound. They have absorbed patterns at every scale, from the way a word shifts meaning depending on its neighbors to the way an argument unfolds across pages. The fluency these systems produce would have seemed, a decade ago, like science fiction.

The question is what they have not absorbed.

Written language is a record of conclusions, not of the process that produced them. It captures what people thought, but not what it cost them to find out. The frustration of a plan failing in real time, the adjustment forced by a world that won't cooperate, the particular knowledge that comes only from having been wrong when it mattered: none of that survives the journey onto the page. A system trained on that record inherits everything humans managed to articulate, and nothing they never needed to.

Researchers at DeepMind argued in a 2021 paper ("Reward is Enough") that maximizing a single measure of success in a sufficiently complex environment may be enough to generate general intelligence. The claim was that an agent trying to maximize a reward in a complex enough world would, as a side effect of that pursuit, develop perception, language, planning, and every other ability associated with intelligence. The evidence was substantial, drawn from systems that had mastered Go, chess, and dozens of video games by playing millions of rounds against themselves. But the argument was framed around closed environments, rule-governed and defined by a single optimizable outcome. What it left open was the question of how intelligence develops in environments that are not closed, not fully rule-governed, and where the consequences of being wrong are not scores but changes in the world that cannot be undone.

A 2025 paper from the same research team ("Welcome to the Era of Experience") shifts the argument from how intelligence is driven to what it is built from. Intelligence, it proposes, is shaped not by any single signal but by the totality of experience: exploration, error, feedback, and sustained interaction with an environment that pushes back. In this framing, the current era of training on human-produced data is a transitional phase. What comes next is an era in which systems learn not from the record of human experience but from experience of their own.

Whether that transition is imminent, distant, or even possible remains an open question. But the argument illuminates something about the history traced in this chapter. Each time the field advanced, it did so by exposing systems to more of the world, in more varied and consequential forms. The symbolic systems improved when they incorporated more expert knowledge. The probabilistic systems improved when they could draw on data. The learning systems improved when the data grew to billions of examples. And the embodied systems, the ones that had to act rather than merely process, demonstrated that there is a kind of knowledge that exists only in the interaction between an agent and an environment that does not forgive mistakes.

The systems available today represent the furthest expression of this recognition, and they may also represent its current limit. They have absorbed nearly everything humanity has written, and what they produce is remarkable. But a record of conclusions is not the same thing as the experience of arriving at them. What that difference consists of, and what it would take to close it, is the question the researchers at DeepMind set out to answer.