2

WHY NOW? A HISTORY OF AI

Rowan Atkinson's comic masterpiece *Mr. Bean* opens in the dead of night on a deserted London street. A spotlight appears, the title character falls from the sky, and a choir sings in Latin, "ecce homo qui est faba"—behold the man who is a bean. Mr. Bean picks himself up, brushes off his suit, and runs awkwardly into the darkness. He is something otherworldly, a thing that literally fell from the sky, defying comprehension.

Given the parade of AI wonder after wonder in recent years, we might be excused for thinking that AI, like Mr. Bean, fell from the sky, fully formed and beyond our comprehension. However, none of this is true; indeed, I'd argue that AI is still in its infancy.

So why are we hearing about AI now? I'll answer that question with a brief (and biased) history of AI, followed by a discussion of the advances in computing that acted as the catalyst for the AI revolution. This chapter provides context for the models we'll explore throughout the remainder of the book.

****

Since its inception, AI has been divided into two main camps: symbolic AI and connectionism. *Symbolic* AI attempts to model intelligence by manipulating symbols and logical statements or associations. *Connectionism*, however, attempts to model intelligence by building networks of simpler components. The human mind embodies both approaches. We use symbols as elements of thought and language, and our minds are constructed from unbelievably complex networks of neurons, each neuron a simple processor. In computer programming terms, the symbolic approach to AI is top-down, while connectionism is bottom-up. Top-down design starts with high-level tasks, then breaks those tasks into smaller and smaller pieces. A bottom-up design begins with smaller pieces and combines them together.

Proponents of symbolic AI believe that intelligence can be achieved in the abstract, without a substrate resembling a brain. Connectionists follow the evolutionary development of brains and argue that there needs to be some foundation, like a massive collection of highly interconnected neurons, from which intelligence (however defined) can emerge.

While the debate between symbolic AI and connectionism was long-lived, with the advent of deep learning it's safe to say that the connectionists have won the day—though perhaps not the war. Recent years have seen a smattering of papers blending the two approaches. I suspect symbolic AI has a cameo or two left in it, if not ultimately starring in a supporting role.

My introduction to AI in the late 1980s was entirely symbolic. Connectionism was mentioned as another approach, but neural networks were thought inferior and likely to be marginally useful, at best.

A complete history of artificial intelligence is beyond our scope. Such a magnum opus awaits a motivated and capable historian. Instead, I'll focus on the development of machine learning while (very unfairly!) ignoring the mountain of effort expended over the decades by those in the symbolic camp. Know, however, that for most of AI's history, people mostly spoke of symbolic AI, not connectionism. For a fairer presentation, I recommend Michael Wooldridge's book *A Brief History of Artificial Intelligence* (Flatiron Books, 2021), or Pamela McCorduck's deeply personal account in *This Could Be Important: My Life and Times with the Artificial Intelligentsia* (Lulu Press, 2019).

With my apparent connectionist bias in mind, let's take a stroll through the history of machine learning.

Pre-1900

The dream of intelligent machines dates back to antiquity. Ancient Greeks related the myth of Talos, a giant robot meant to guard the Phoenician princess, Europa. Throughout the Middle Ages and Renaissance, many automatons—machines that moved and appeared lifelike—were developed. However, I suspect none were believed to be intelligent or capable of thought. Some were even hoaxes, like the infamous Mechanical Turk that wowed the world by playing, and beating, many skilled chess players. In the end, it was discovered that a person hiding within the machine could control the "automaton" by manipulating a mechanical arm to move free-standing chess pieces on the

board while viewing the board configuration from beneath. Still, the mechanical part of the machine was rather impressive for the late 18th century.

Apart from automatons, there were also early attempts to understand thought as a mechanical process and efforts to produce a logical system capable of capturing thought. In the 17th century, Gottfried Leibniz described such a concept abstractly as an "alphabet of thought." In the 1750s, Julien Offray de La Mettrie published *L'Homme Machine* (*Man as Machine*), arguing that thought is a mechanical process.

The idea that human thought might emerge from the physical entity of the brain rather than the spiritual soul marked the beginning of a new chapter on the road to AI. If our minds are biological machines, why can't there be another kind of machine that thinks?

In the 19th century, George Boole attempted to create a calculus of thought, resulting in what we know now as Boolean algebra. Computers depend on Boolean algebra, to the point that it represents their very implementation as collections of digital logic gates. Boole was partially successful, but he didn't achieve his stated goal: "to investigate the fundamental laws of those operations of the mind by which reasoning is performed; to give expression to them in the symbolic language of a Calculus" (*The Laws of Thought*, 1854). That Boole was willing to try represented another step toward the notion that AI might be possible.

What these early attempts were lacking was an actual calculating machine. People could dream of artificial minds or beings (like the creature from Mary Shelley's *Frankenstein*) and, assuming their existence, discuss the repercussions. But until there was a machine capable of plausibly mimicking (implementing?) thought, all else was speculation.

It was Englishman Charles Babbage who, in the mid-19th century, first conceived of an implementable general-purpose calculating machine: the Analytical Engine. The Engine was never built in its entirety, but it contained all the essential components of a modern computer and would, in theory, be capable of the same operations. While it's unclear if Babbage appreciated the potential versatility of his machine, his friend, Ada Lovelace, did. She wrote about the machine as a widely applicable, general-purpose device. Still, she did not believe the Engine was capable of thought, as this quote from her *Sketch of the Analytical Engine* (1843) demonstrates:

> The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform. It can follow analysis; but it has no power of anticipating any analytical relations or truths. Its province is to assist us to making available what we are already acquainted with.

This quote may be the first to refer to the possibility of artificial intelligence involving a device potentially capable of achieving it. The phrase "do whatever we know how to order it to perform" implies programming. Indeed, Lovelace wrote a program for the Analytical Engine. Because of this, many people consider her to be the first computer programmer. The fact that her program had a bug in it proves to me that she was; nothing is more

emblematic of programming than bugs, as my 40-plus years of programming experience have demonstrated distressingly often.

1900 to 1950

In 1936, a 24-year-old Englishman named Alan Turing, still a student at the time, wrote a paper that has since become the cornerstone of computer science. In this paper, Turing introduced a generic conceptual machine, what we now call a *Turing machine*, and demonstrated that it could calculate anything representable by an algorithm. He also explained that there are things that cannot be implemented by algorithms and that are, therefore, uncomputable. Since all modern programming languages are equivalent to a Turing machine, modern computers can implement any algorithm and compute anything computable. However, this says nothing about how long the computation might take or the memory required.

If a computer can compute anything that can be implemented as an algorithm, then a computer can perform any mental operation a human can perform. At last, here was the engine that might enable true artificial intelligence. Turing's 1950 paper "Computing Machinery and Intelligence" was an early recognition that digital computers might eventually lead to intelligent machines. In this paper Turing described his "imitation game," known now as the *Turing test*, by which humans might come to believe that a machine is intelligent. Many claims of AI systems that pass the Turing test have appeared, especially in recent years. One of these is OpenAI's ChatGPT. However, few would be inclined to believe that ChatGPT is truly intelligent—in other words, I suspect that this test fails to capture what humans generally understand this term to mean, and a new test will likely be created at some point.

In 1943, Warren McCulloch and Walter Pitts wrote "A Logical Calculus of Ideas Immanent in Nervous Activity," which deserves an award for one of the most opaque yet intriguing paper titles ever. The paper represents "nervous nets" (collections of neurons) as logical statements in mathematics. The logical statements are difficult to parse (at least for me), but the authors' description of "nets without circles" bears a strong resemblance to the neural networks we'll explore in Chapter 4—indeed, one could argue that McCulloch and Pitts's groundbreaking paper led to what we now recognize as a neural network. Frankly, neural networks are far easier to parse and understand, which is good news for us.

The progression from fantastical stories about artificially intelligent machines and beings to a serious investigation of whether mathematics can capture thought and reasoning, combined with the realization that digital computers are capable of computing anything that can be described by an algorithm, set the stage for the advent of artificial intelligence as a legitimate research enterprise.

1950 to 1970

The 1956 Dartmouth Summer Research Project on Artificial Intelligence workshop is generally regarded as the birthplace of AI, and where the phrase "artificial intelligence" was first used consistently. The Dartmouth workshop had fewer than 50 participants, but the list included several well-known names in the worlds of computer science and mathematics: Ray Solomonoff, John McCarthy, Marvin Minsky, Claude Shannon, John Nash, and Warren McCulloch, among others. At the time, computer science was a subfield of mathematics. The workshop was a brainstorming session that set the stage for early AI research.

In 1957, Frank Rosenblatt of Cornell University created the Mark I Perceptron, widely recognized as the first application of neural networks. The Perceptron was remarkable in many respects, including that it was designed for image recognition, the same application where deep learning first proved itself in 2012.

Figure 2-1 shows the conceptual organization as given in the *Perceptron Operators' Manual*. The Perceptron used a 20×20-pixel digitized television image as input, which was then passed through a "random" set of connections to a set of association units that led to response units. This configuration is similar to some approaches to deep learning on images in use today, and resembles a type of neural network known as an *extreme learning machine*.

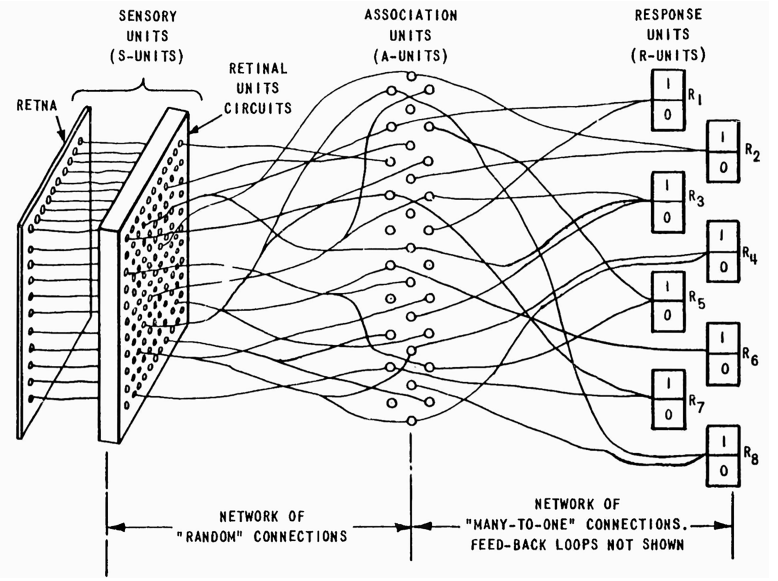

Figure 2-1: The organization of the Mark I Perceptron

If the Perceptron was on the right track, why was it all but forgotten for decades? One reason was Rosenblatt's penchant for hype. At a 1958 conference organized by the US Navy (a sponsor of the Perceptron project), Rosenblatt's comments were so hyperbolic that the *New York Times* reported:

> The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. Later perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech and writing in another language, it was predicted.

The comments ruffled many feathers at the time, though as modern AI systems do allow machines to walk, talk, see, write, recognize people, and translate speech and writing between languages, perhaps we should be more forgiving toward Rosenblatt. He was only some 60 years early.

A few years later, in 1963, Leonard Uhr and Charles Vossler described a program that, like the Perceptron, interpreted a 20×20-pixel image represented as a matrix of 0s and 1s. Unlike the Perceptron, this program could generate the patterns and combinations of image features necessary to learn its inputs. Uhr and Vossler's program was similar to the convolutional neural networks that appeared over 30 years later and are the subject of Chapter 5.

The first of what I call the "classical" machine learning models appeared in 1967, courtesy of Thomas Cover and Peter Hart. Known as *nearest neighbors*, it is the simplest of all machine learning models, almost embarrassingly so. To label an unknown input, it simply finds the known input most like it and uses that input's label as the output. When using more than one nearby known input, the method is called *k-nearest neighbors*, where k is a small number, like 3 or 5. Hart went on to write the first edition of *Pattern Classification*, along with Richard Duda and David Stork, in 1973; this seminal work introduced many computer scientists and software engineers to machine learning, including me.

The success of the Perceptron came to a screeching halt in 1969 when Marvin Minsky and Seymour Papert published their book, *Perceptrons*, which demonstrated that single and two-layer perceptron networks weren't able to model interesting tasks. We'll cover what "single" and "two-layer" mean in time. *Perceptrons*, coupled with the 1973 release of "Artificial Intelligence: A General Survey" by James Lighthill, universally known as "the Lighthill report," ushered in what is now referred to as the first AI winter; funding for AI research dried up in short order.

Minsky and Papert's criticisms of the perceptron model were legitimate; however, many people missed their observation that such limitations were not applicable to more complex perceptron models. Regardless, the damage was done, and connectionism virtually vanished until the early 1980s.

Note the "virtually." In 1979, Kunihiko Fukushima released a paper that was translated to English in 1980 as "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position." The name "Neocognitron" didn't catch on, and this was perhaps one of the last uses of the "-tron" suffix that had been so popular in computer science for the previous three decades. While Uhr and Vossler's 1963 program bore some similarities to a convolutional neural network, the Neocognitron is, to many people, the original. The success of convolutional neural networks led directly to the current AI revolution.

1980 to 1990

In the early 1980s, AI went commercial with the advent of computers specifically designed to run the Lisp programming language, then the lingua franca of AI (today, it's Python). Along with Lisp machines came the rise of *expert systems*—software designed to capture the knowledge of an expert in a narrow domain. The commercialization of AI brought the first AI winter to an end.

The concept behind expert systems is, admittedly, seductive. To build an expert system that, for example, diagnoses a particular kind of cancer, you first interview experts to extract their knowledge and arrange it in a knowledge base. A knowledge base represents knowledge as a combination of rules and facts. Then, you combine the knowledge base with an inference engine, which uses the knowledge base to decide when and how to execute rules based on stored facts or input to the system by a user. Rules fire based on facts, which may lead to placing new facts in the knowledge base that cause additional rules to fire, and so on. A classic example of an expert system is CLIPS, which NASA developed in 1985 and released into the public domain in 1996.

In an expert system, there's no connectionist network or collection of units from which one might (hopefully) cause intelligent behavior to emerge, making it a good example of symbolic AI. Instead, the knowledge base is an essentially rigid collection of rules, like "if the engine temperature is above this threshold, then this other thing is the likely cause," and facts, like "the engine temperature is below the threshold." Knowledge engineers are the links between the experts and the expert system. Building a knowledge base from the experts' answers to the questions posed by the knowledge engineers is complex, and the resulting knowledge base is hard to modify over time. However, the difficulty in designing expert systems doesn't mean they're useless; they still exist, mainly under the guise of "business rule management systems," but currently have minimal impact on modern AI.

The hype surrounding expert systems, combined with early successes, drove renewed interest in AI in the early 1980s. But when it became clear that expert systems were too brittle to have a general use, the bottom fell out of the industry, and AI's second winter hit in the middle of the decade.

During the 1980s, connectionists occupied the background, but they were not sitting still. In 1982, John Hopfield demonstrated what are now known as Hopfield networks. A *Hopfield network* is a type of neural network that stores information in a distributed way within the weights of the network, and then extracts that information at a later time. Hopfield networks aren't widely used in modern deep learning, but they proved an important demonstration of the utility of the connectionist approach.

In 1986, David Rumelhart, Geoffrey Hinton, and Ronald Williams released their paper "Learning Representations by Back-propagating Errors," which outlined the backpropagation algorithm for training neural networks. Training a neural network involves adjusting the weights between the neurons so that the network operates as desired. The backpropagation algorithm was the key to making this process efficient by calculating how adjusting a particular weight affects the network's overall performance. With this

information, it becomes possible to iteratively train the network by applying known training data, then using the network's errors when classifying to adjust the weights to force the network to perform better on the next iteration (I'll discuss neural network training in more depth in Chapter 4). With backpropagation, neural networks could go well beyond the limited performance of Rosenblatt's Perceptron. However, even with backpropagation, neural networks in the 1980s were little more than toys. While there's contention about who invented backpropagation and when, the 1986 paper is generally understood to be the presentation that influenced neural network researchers the most.

1990 to 2000

The second AI winter extended into the 1990s, but research continued in both the symbolic and connectionist camps. Corinna Cortes and Vladimir Vapnik introduced the machine learning community to *support vector machines (SVMs)* in 1995. In a sense, SVMs represent the high-water mark of classical machine learning. The success of SVMs in the 1990s through the early 2000s held neural networks at bay. Neural networks require large datasets and significant computational power; SVMs, on the other hand, are often less demanding of resources. Neural networks gain their power from the network's ability to represent a function, a mapping from inputs to the desired outputs, while SVMs use clever mathematics to simplify difficult classification problems.

The success of SVMs was noted in the academic community as well as the broader world of software engineering, where applications involving machine learning were increasing. The general public was largely unaware of these advances, though intelligent machines continued appearing frequently in science fiction.

This AI winter ended in 1997 with the victory of IBM's Deep Blue supercomputer against then-world chess champion Gary Kasparov. At the time, few people thought a machine could ever beat the best human chess player. Interestingly, a decade earlier, one of my professors had predicted that an AI would accomplish this feat before the year 2000. Was this professor clairvoyant? Not really. Deep Blue combined fast custom hardware with sophisticated software and applied known AI search algorithms (in particular, the Minimax algorithm). Combined with heuristics and a healthy dose of custom knowledge from other chess grandmasters, Deep Blue was able to out-evaluate its human opponent by searching more possible moves than any human could ever hope to contemplate. Regardless, at its core, Deep Blue implemented what AI experts knew *could* beat a human if the machine had enough resources at its disposal. Deep Blue's victory was inevitable because researchers expected computers to eventually become fast enough to overcome a human's abilities. What was needed was known; all that remained was to put the pieces together.

1998 saw the publication of "Gradient-Based Learning Applied to Document Recognition," a paper by Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner that escaped public notice but was a watershed moment

for AI and the world. While Fukushima's Neocognitron bore strong similarities to the convolutional neural networks that initiated the modern AI revolution, this paper introduced them directly, as well as the (in)famous MNIST dataset we used in Chapter 1. The advent of convolutional neural networks (CNNs) in 1998 begs the question: why did it take another 14 years before the world took notice? We'll return to this question later in the chapter.

2000 to 2012

Leo Breiman introduced *random forests* in 2001 by forming the existing pieces of what would become the random forest algorithm into a coherent whole, much like Darwin did with evolution in the 19th century. Random forests are the last of the classical machine learning algorithms we'll contemplate in Chapter 3. If "random forests" remind you of the decision trees in Chapter 1, there's a reason: a random forest is a forest of decision trees.

Stacked denoising autoencoders are one type of intermediate model, and they were my introduction to deep learning in 2010. An *autoencoder* is a neural network that passes its input through a middle layer before generating output. It aims to reproduce its input from the encoded form of the input in the middle layer.

An autoencoder may seem like a silly thing to fiddle with, but while learning to reproduce its input, the middle layer typically learns something interesting about the inputs that captures their essence without focusing on fine, trivial details. For example, if the inputs are the MNIST digits, then the middle layer of an autoencoder learns about digits as opposed to letters.

A *denoising autoencoder* is similar, but we discard a random fraction of the input values before pushing the input through the middle layer. The autoencoder must still learn to reproduce the entire input, but now it has a more challenging task because the input is incomplete. This process helps the autoencoder's middle layer discover a better encoding of the input.

Finally, a *stacked denoising autoencoder* is a stack of denoising autoencoders, wherein the output of the middle layer of one becomes the input of the next. When arranged this way, the stack learns a new representation of the input, which often helps a classifier appended to the top of the stack to discriminate between classes. For example, in my work at the time, the inputs were small pieces of an image that may have contained a target of interest. Two or three layers of trained stacked denoising autoencoders were used to transform the inputs into a list of numbers that would hopefully represent the input's essence while ignoring the image's minutiae. The outputs were then used with a support vector machine to decide if the input was a target.

2012 to 2021

Deep learning caught the world's attention in 2012 when AlexNet, a particular convolutional neural network architecture, won the ImageNet challenge with an overall error of just over 15 percent, far lower than any competitor. The ImageNet challenge asks models to identify the main subject of color

images, whether a dog, a cat, a lawnmower, and so on. In reality, "dog" isn't a sufficient answer. The ImageNet dataset contains 1,000 classes of objects, including some 120 different dog breeds. So, a correct answer would be "it's a Border Collie" or "it's a Belgian Malinois."

Random guessing means randomly assigning a class label to each image. In that case, we would expect an overall success rate of 1 in 1,000, or an error rate of 99.9 percent. AlexNet's error of 15 percent was truly impressive—and that was in 2012. By 2017, convolutional neural networks had reduced the error to about 3 percent, below the approximate 5 percent achievable by the few humans brave enough to do the challenge manually. Can you discriminate between 120 different dog breeds? I certainly can't.

AlexNet opened the floodgates. The new models broke all previous records and began to accomplish what no one had really expected from them: tasks like reimagining images in the style of another image or painting, generating a text description of the contents of an image along with the activity shown, or playing video games as well as or better than a human, among others.

The field was proliferating so quickly that it became nearly impossible to keep up with each day's deluge of new papers. The only way to stay current was to attend multiple conferences per year and review the new work appearing on websites such as arXiv (*https://www.arxiv.org*), where research in many fields is first published. This led to the creation of sites like *https://www.arxiv-sanity-lite.com*, which ranks machine learning papers according to reader interest in the hope that the "best" might become easier to find.

In 2014, another breakthrough appeared on the scene, courtesy of researcher Ian Goodfellow's insight during an evening's conversation with friends. The result was the birth of *generative adversarial networks (GANs)*, which Yann LeCun called at the time the most significant breakthrough in neural networks in 20 to 30 years (overheard at NeurIPS 2016). GANs, which we'll discuss in Chapter 6, opened a new area of research that lets models "create" output that's related to but different from the data on which it was trained. GANs led to the current explosion of *generative AI*, including systems like ChatGPT and Stable Diffusion.

Reinforcement learning is one of the three main branches of machine learning, the other two being the supervised learning we've been discussing and unsupervised learning, which attempts to train models without labeled datasets. In reinforcement learning, an agent (a model) is taught via a reward function how to accomplish a task. The application to robotics is obvious.

Google's DeepMind group introduced a deep reinforcement learning–based system in 2013 that could successfully learn to play Atari 2600 video games as well as or better than human experts (who counts as an expert in a then-35-year-old game system, I'm not sure). The most impressive part of the system, to me, was that the model's input was precisely the human's input: an image of the screen, nothing more. This meant the system had to learn to

parse the input image and, from that, how to respond by moving the joystick to win the game (virtually—they used emulators).

The gap between beating humans at primitive video games and beating humans at abstract strategy games like Go was, historically, deemed insurmountable. I was explicitly taught in the late 1980s that the Minimax algorithm used by systems like Deep Blue to win at chess did not apply to a game like Go; therefore, no machine would ever beat the best human Go players. My professors were wrong, though they had every reason at the time to believe their statement.

In 2016, Google's AlphaGo system beat Go champion Lee Sedol in a five-game match, winning four to one. The world took notice, further enhancing the growing realization that a paradigm shift had occurred. By this time, machine learning was already a commercial success. However, AlphaGo's victory was utterly impressive for machine learning researchers and practitioners.

Most of the general public didn't notice that AlphaGo, trained on thousands of human-played Go games, was replaced in 2017 by AlphaGo Zero, a system trained entirely from scratch by playing against itself, with no human input given. In short order, AlphaGo Zero mastered Go, even beating the original AlphaGo system (scoring a perfect 100 wins and no losses).

However, in 2022, the current state-of-the-art Go system, KataGo, was repeatedly and easily defeated by a system trained not to win but to reveal the brittleness inherent in modern AI systems. The moves the adversarial system used were outside the range encountered by KataGo when it was trained. This is a real-world example of how models are good at interpolating but bad at extrapolating. When the adversarial system was trained not to be better at Go but to exploit and "frustrate" the AI, it was able to win better than three out of four games. I point the reader to the *Star Trek: The Next Generation* episode "Peak Performance," where Data the android "wins" a difficult strategy game against a master not by attempting to win but by attempting to match and frustrate.

Deep learning's penchant for beating humans at video games continues. In place of primitive games like Atari, deep reinforcement learning systems are now achieving grandmaster-level performance at far more difficult games. In 2019, DeepMind's AlphaStar system outperformed 99.8 percent of human players in *StarCraft II*, a strategy game requiring the development of units and a plan of battle.

The 1975 Asilomar Conference on Recombinant DNA was an important milestone in recognizing biotechnology's growth and potential ethical issues. The conference positively impacted future research, and that year its organizers published a summary paper outlining an ethical approach to biotechnology. The potential hazards of a field that was then primarily in its infancy were recognized early, and action was taken to ensure ethical issues were paramount when contemplating future research.

The 2017 Asilomar Conference on Beneficial AI intentionally mirrored the earlier conference to raise awareness of the potential hazards associated with AI. It is now common to encounter conference sessions with titles like

"AI for Good." The 2017 Asilomar conference resulted in the development of a set of principles to guide the growth and application of artificial intelligence. Similarly, as of 2023, the US government—specifically, the White House Office of Science and Technology Policy—has developed a "Blueprint for an AI Bill of Rights" meant to protect the American public from the harmful effects of AI indiscriminately applied. Indeed, White House officials have taken pains to address the AI community directly to encourage proper consideration in developing even more powerful AI systems. All of this is a good sign, but history teaches that human law often lags behind technological development, so the ultimate effectiveness of these necessary attempts at framing the field remains to be seen.

2021 to Now

The COVID-19 pandemic of 2020 brought most of the world to a standstill. However, the AI community was only minimally impacted by the pandemic, likely because remote collaboration and conferences work rather well in this field. In addition, we can access powerful computers over the internet, so physical proximity doesn't limit research as it might in other disciplines where access to the hardware is necessary.

Beginning in 2021 and continuing as I write, an explosion of new models have appeared, each more impressive than the last. Most can accept text input written by humans to produce text, images, or even video output. Often, the input marries text and example imagery to guide the system. Current popular (and concerning) systems include DALL-E 2, Stable Diffusion, Midjourney, and ChatGPT, among others. Most large tech companies have also demonstrated cutting-edge techniques that generate video, audio, and even 3D models. The current buzz generated by these systems is reminiscent of the excitement in 2012 when deep learning proved its worth with AlexNet. It's another paradigm shift; but to be clear, none of these systems are conscious. I'm not alone in this opinion. Even ChatGPT agrees with me. I asked it: "Are you a conscious being?" Here's its reply, verbatim:

> As a machine learning model, I am not a conscious being. I am a computer program that has been trained to generate text based on input from a user. I do not have the ability to think, reason, or experience consciousness in the same way that humans do. I am designed to provide information and assist users, but I do not have any independent thought or consciousness.

I think Ada Lovelace would be satisfied with this answer, though she might be puzzled about how an unthinking machine could generate it. Note, however, that ChatGPT didn't claim not to think, only that it doesn't think like humans do. We'll explore image synthesis in Chapter 6 and large language models like ChatGPT in Chapter 7. Perhaps then we'll find a resolution to her (assumed) confusion.

****

So, why now? The short answer is the fall of symbolic AI and the rise of technological innovations highly favorable to the connectionist approach.

Symbolic AI and connectionism emerged together, with symbolic AI dominating for decades and forcing connectionism into the background. But after two AI winters that left symbolic AI barely breathing, connectionism, assisted by key technological innovations, has risen to fill the void.

I think of the relationship between symbolic AI and connectionism as akin to that between non-avian dinosaurs and mammals. Dinosaurs and mammals emerged at roughly the same time, geologically speaking, but large, terrestrial dinosaurs dominated the world for about 160 million years, forcing mammals to eke out an existence in the shadows. When the asteroid hit 66 million years ago, the large dinosaurs were wiped out, allowing the mammals to evolve and take over.

Of course, analogies ultimately break down. The dinosaurs didn't die out completely—we now call them birds—and they didn't go extinct because they were somehow inferior. In fact, the dinosaurs are one of Earth's greatest success stories. Non-avian dinosaurs died because of plain old bad luck. It was, almost literally, a disaster that did them in ("disaster" from the Italian *disastro*, meaning "ill star").

Might symbolic AI re-emerge? It's likely in some form, but in cooperation with connectionism. Symbolic AI promised that intelligent behavior was possible in the abstract, and it didn't deliver. Connectionism claims that intelligent behavior can emerge from a collection of simpler units. Deep learning's successes support this view, to say nothing of the billions of living brains currently on the planet. But, as ChatGPT pointed out, existing connectionist models "do not think, reason, or experience consciousness in the same way that humans do." Modern neural networks are not minds; they are representation-learning data processors. I'll clarify what that means in Chapter 5.

Though our species, *Homo sapiens*, relies critically on symbolic thought, it isn't a requirement for intelligence. In his book *Understanding Human Evolution* (Cambridge University Press, 2022), anthropologist Ian Tattersall claims it was unlikely that Neanderthals used symbolic thought as we do, nor did they have language as we do, but that they were nonetheless intelligent. Indeed, the Neanderthals were sufficiently human for our ancestors to "make love, not war" with them more than once—the DNA of people of non-African ancestry testifies to this fact.

I expect a synergy between connectionism and symbolic AI in the near future. For example, because a system like ChatGPT is, in the end, only predicting the next output token (word or part of a word), it can't know when it's saying something wrong. An associated symbolic system could detect faulty reasoning in the response and correct it. How such a system might be implemented, I don't know.

****

Hints of what might emerge from connectionism were evident by the early 1960s. So, was it only symbolic AI bias that delayed the revolution for

so many decades? No. Connectionism stalled because of speed, algorithm, and data issues. Let's examine each in turn.

Speed

To understand why speed stalled the growth of connectionism, we need to understand how computers work. Taking great liberties allows us to think of computers as memory, which holds data (numbers) and a processing unit, typically known as the central processing unit (CPU). A microprocessor—like the one in your desktop computer, smartphone, voice-controlled assistant, car, microwave, and virtually everything else you use that isn't a toaster (oh, and in many toasters, too)—is a CPU. Think of a CPU as a traditional computer: data comes into the CPU from memory or input devices like a keyboard or mouse, gets processed, then is sent out of the CPU to memory or an output device like a monitor or hard drive.

Graphics processing units (GPUs), on the other hand, were developed for displays, primarily for the video game industry, to enable fast graphics. GPUs can perform the same operation, such as "multiply by 2," on hundreds or thousands of memory locations (read *pixels*) simultaneously. If a CPU wants to multiply a thousand memory locations by 2, it must multiply the first, second, third, and so on sequentially. As it happens, the primary operation needed to train and implement a neural network is ideally suited to what a GPU can do. GPU makers, like NVIDIA, realized this early and began developing GPUs for deep learning. Think of a GPU as a supercomputer on a card that fits in your PC.

In 1945, the Electronic Numerical Integrator and Computer (ENIAC) was state-of-the-art. ENIAC's speed was estimated to be around 0.00289 million instructions per second (MIPS). In other words, ENIAC could perform just under 3,000 instructions in one second. In 1980, a stock 6502 8-bit microprocessor like the ones in most then-popular personal computers ran at about 0.43 MIPS, or some 500,000 instructions per second. In 2023, the already somewhat outdated Intel i7-4790 CPU in the computer I'm using to write this book runs at about 130,000 MIPS, making my PC some 300,000 times faster than the 6502 from 1980 and about 45 million times faster than ENIAC.

However, NVIDIA's A100 GPU, when used for deep learning, is capable of 312 teraflops (TFLOPS), or 312,000,000 MIPS: 730 million times faster than the 6502 and an unbelievable 110 *billion* times faster than ENIAC. The increase in computational power over the timespan of machine learning boggles the mind. Moreover, training a large neural network on an enormous dataset often requires dozens to hundreds of such GPUs.

Conclusion: Computers were, until the advent of fast GPUs, too slow to train neural networks with the capacity needed to build something like ChatGPT.

Algorithm

As you'll learn in Chapter 4, we construct neural networks from basic units that perform a simple task: collect input values, multiply each by a weight value, sum, add a bias value, and pass the result to an activation function to create an output value. In other words, many input numbers become one output number. The collective behavior emerging from thousands to millions of such units leading to billions of weight values lets deep learning systems do what they do.

The structure of a neural network is one thing; conditioning the neural network to the desired task is another. Think of the network's structure, known as its *architecture*, as anatomy. In anatomy, we're interested in what constitutes the body: this is the heart, that's the liver, and so on. Training a network is more like physiology: how does one part work with another? The anatomy (architecture) was there, but the physiology (training process) was incompletely understood. That changed over the decades, courtesy of key algorithmic innovations: backpropagation, network initialization, activation functions, dropout and normalization, and advanced gradient descent algorithms. It's not essential to understand the terms in detail, only to know that improvements in what these terms represent—along with the already mentioned improvements in processing speed, combined with improved datasets (discussion coming up)—were primary enablers of the deep learning revolution.

While it was long known that the right weight and bias values would adapt a network to the desired task, what was missing for decades was an efficient way to *find* those values. The 1980s' introduction of the backpropagation algorithm, combined with stochastic gradient descent, began to change this.

Training iteratively locates the final set of weight and bias values according to the model's errors on the training data. Iterative processes repeat from an initial state, some initial set of weights and biases. However, what should those initial weights and biases be? For a long time, it was assumed that these initial weights and biases didn't matter much; just select small numbers at random over some range. This approach often worked, but many times it didn't, causing the network not to learn well, if at all. A more principled approach to initializing networks was required.

Modern networks are still initialized randomly, but the random values depend on the network's architecture and the type of activation function used. Paying attention to these details allowed networks to learn better. Initialization matters.

We arrange neural networks in layers, where the output of one layer becomes the input of the next. The activation function assigned to each node in the network determines the node's output value. Historically, the activation function was either a sigmoid or a hyperbolic tangent, both of which produce an *S*-shaped curve when graphed. These functions are, in most cases, inappropriate, and were eventually replaced by a function with a long name that belies its simplicity: the *rectified linear unit (ReLU)*. A ReLU asks a simple question: is the input less than zero? If so, the output is zero; other-

wise, the output is the input value. Not only are ReLU activation functions better than the older functions, but computers can ask and answer that question virtually instantaneously. Switching to ReLUs was, therefore, a double win: improved network performance and speed.

Dropout and batch normalization are advanced training approaches that are somewhat difficult to describe at the level we care to know about them. Introduced in 2012, *dropout* randomly sets parts of the output of a layer of nodes to zero when training. The effect is like training thousands of models simultaneously, each independent but also linked. Dropout, when appropriate, has a dramatic impact on network learning. As a prominent computer scientist told me at the time, "If we had had dropout in the 1980s, this would be a different world now."

Batch normalization adjusts the data moving between layers as it flows through the network. Inputs appear on one side of the network and flow through layers to get to the output. Schematically, this is usually presented as a left-to-right motion. Normalization is inserted between the layers to change the values to keep them within a meaningful range. Batch normalization was the first learnable normalization technique, meaning it learned what it should do as the network learned. An entire suite of normalization approaches evolved from batch normalization.

The last critical algorithmic innovation enabling the deep learning revolution involves gradient descent, which works with backpropagation to facilitate learning the weights and biases. The idea behind gradient descent is far older than machine learning, but the versions developed in the last decade or so have contributed much to deep learning's success. We'll learn more about this subject in Chapter 4.

Conclusion: The first approaches to training neural networks were primitive and unable to take advantage of their true potential. Algorithmic innovations changed that.

Data

Neural networks require lots of training data. When people ask me how much data is necessary to train a particular model for a specific task, my answer is always the same: all of it. Models learn from data; the more, the better because more data means an improved representation of what the model will encounter when used.

Before the World Wide Web, collecting, labeling, and processing datasets of the magnitude necessary to train a deep neural network proved difficult. This changed in the late 1990s and the early 2000s with the tremendous growth of the web and the explosion of data it represented.

For example, Statista (*https://www.statista.com*) claims that in 2022, 500 hours of new video were uploaded to YouTube *every minute*. It's also estimated that approximately 16 million people were using the web in December 1995, representing 0.4 percent of the world's population. By July 2022, that number had grown to nearly 5.5 billion, or 69 percent. Social media use, e-commerce, and simply moving from place to place while carrying a

smartphone are enough to generate staggering amounts of data—all of which is captured and used for AI. Social media is free because we, and the data we generate, are the product.

A phrase I often hear in my work is "we used to be data-starved, but now we're drowning in data." Without large datasets and enough labels to go with them, deep learning cannot learn. But on the other hand, with large datasets, awe-inspiring things can happen.

Conclusion: In machine learning, data is everything.

****

The main takeaways from this chapter are:

- The symbolic AI versus connectionist feud appeared early and led to decades of symbolic AI dominance.
- Connectionism suffered for a long time because of speed, algorithm, and data issues.
- With the deep learning revolution of 2012, the connectionists have won, for now.
- The direct causes of the deep learning revolution were faster computers, the advent of graphics processing units, improved algorithms, and huge datasets.

With this historical background complete enough for our purposes, let's return to machine learning, starting with the classical algorithms.