HISTORY OF AI

The Statistical & Deep Learning Era

The Statistical Turn · Deep Blue vs Kasparov

ImageNet & AlexNet · AlphaGo · Transformers

The GPT Lineage & the ChatGPT Moment

Philip Osztromok

Generated with Claude

Table of Contents

History of AI — The Statistical & Deep Learning Era — 8 Chapters

📊 The Statistical Turn

Course 2 ended by naming the single root cause behind both AI winters: symbolic AI's absolute dependence on a human manually encoding every fact and rule a system would ever use. This chapter opens Course 3 with the shift that finally addressed that root cause directly — not by narrowing the target, the way Chapter 6's expert systems did, but by changing the method itself. Instead of a person writing the knowledge down, the system would learn it from data.

🔀 A Different Kind of Answer

Statistics, probability theory, and regression had existed for well over a century before this chapter's timeframe — none of the underlying mathematics was new. What changed, gradually through the late 1980s and across the 1990s, was that AI researchers began treating statistical, data-driven methods as the field's **primary** strategy, rather than as a minor supplement bolted onto hand-coded symbolic systems. A statistical system doesn't need someone to write "if a patient has symptom X and Y, suspect diagnosis Z" (MYCIN's approach, Course 2 Chapter 6) — instead, it's shown many examples of patients and diagnoses, and it infers the underlying pattern itself.

This is a genuinely different kind of answer to the knowledge acquisition bottleneck than Chapter 6's expert systems gave. Expert systems made hand-encoding *feasible* by shrinking the domain until a human could realistically write all the rules. Statistical learning removes the human hand-encoding step almost entirely — the bottleneck isn't sidestepped, it's replaced with a different resource requirement: enough data, and enough computing power to learn from it.

🎲 Bayesian Networks — Reasoning Under Uncertainty, Formalized

In 1988, computer scientist **Judea Pearl** published his foundational work on **Bayesian networks** — graphical models that represent a set of variables and their probabilistic dependencies in a principled, mathematically grounded way, and support genuine probabilistic inference: given some evidence, calculate the actual probability of a cause.

A Direct Upgrade Over Certainty Factors

Recall MYCIN's certainty factors from Course 2 Chapter 6 — a pragmatic, hand-tuned numeric confidence score bolted onto each rule, invented largely because rigorous probability theory was considered too impractical to compute with at the time. Pearl's Bayesian networks did the job certainty factors were improvising toward, properly: real probability theory, computationally tractable, with a solid mathematical foundation underneath every number the system produced. Pearl won the Turing Award in 2011 substantially for this contribution.

📈 Support Vector Machines — Finding the Best Boundary

Vladimir Vapnik and **Corinna Cortes** published the modern formulation of the **support vector machine (SVM)** in 1995 — a statistical classification method that, given labeled examples of two categories, finds the optimal boundary (a "hyperplane") that separates them with the widest possible margin. SVMs became one of the most widely used and successful machine learning techniques of the following decade, applied to tasks including handwriting recognition and text classification, and were prized for working well even with comparatively modest amounts of training data — a real practical advantage in an era before the internet-scale datasets Chapter 4 will cover.

🗣️ Machine Translation, Redeemed

Course 2 Chapter 5 covered the ALPAC Report's 1966 verdict: rule-based machine translation had stalled badly, and its failure delivered the first major funding blow to AI-adjacent research. This chapter's statistical turn gives that exact story a genuine second act.

Where Hand-Coded Rules Failed, Statistics Succeeded

Starting in the late 1980s, IBM researchers developed **statistical machine translation** models that learned word-alignment patterns directly from large bilingual text corpora (matched pairs of documents already translated by humans), rather than relying on hand-coded grammar and translation rules. This approach — refined through the 1990s — dramatically outperformed the rule-based systems ALPAC had assessed decades earlier. The very application that delivered AI's first funding collapse became one of the statistical turn's clearest practical vindications.

📐 A New Scientific Culture: Benchmarks Over Demos

The statistical turn brought a cultural shift alongside the technical one. Earlier AI research (SHRDLU, DENDRAL, MYCIN — all covered in Course 2) was largely evaluated through impressive, qualitative demonstrations: watch the system do something remarkable, then judge it by how convincing that looked. The **UCI Machine Learning Repository**, established in 1987, offered researchers a shared collection of standard datasets, enabling something the field had rarely done rigorously before: measuring different methods against each other on identical data, with a genuine numeric score. Empirical, quantitative benchmarking — not demo persuasiveness — became the field's new standard of evidence.

📜 Two Different Fixes for the Same Bottleneck

Expert Systems (Course 2, Ch. 6)	Statistical Learning (This Chapter)
Knowledge source: a human expert's rules, hand-encoded one at a time	Knowledge source: patterns inferred automatically from data
Scales by narrowing the domain until hand-encoding is feasible	Scales by adding more data and more computing power
Evaluated by demonstration — does it look convincing on a chosen case?	Evaluated by benchmark — a numeric score on shared, standard data

🔮 The Bottleneck Moves, It Doesn't Vanish

It's worth resisting the temptation to declare the knowledge acquisition bottleneck fully solved here. Statistical learning trades "a human must hand-write every rule" for "the system needs enough labeled data and enough computing power to learn from" — a genuinely different constraint, and one that will turn out to be far more scalable, but a constraint all the same. This chapter's methods (Bayesian networks, SVMs, statistical MT) worked with the modest data and computing power available in the 1990s. What happens once genuinely large datasets and modern computing power actually arrive — starting with Chapter 4's ImageNet moment — is where this course's real transformation begins.

🤔 Questions to Sit With

Reflection 1

Statistical MT succeeded at the exact task — machine translation — that triggered AI's first funding collapse in Course 2. Does that feel like poetic justice, or does it suggest ALPAC's 1966 verdict was really a verdict on rule-based methods specifically, not on machine translation as a goal?

Reflection 2

The shift from demo-driven evaluation to benchmark-driven evaluation sounds like straightforward scientific progress. Can you think of a downside to a field optimizing hard for a numeric benchmark score, rather than for a compelling real-world demonstration?

Reflection 3

This chapter argues the knowledge acquisition bottleneck didn't disappear — it moved, from "a human must write the rules" to "the system needs enough data and compute." Is trading one resource constraint for another genuine progress, or just a more scalable version of the same underlying limitation?

🎯 What's Next

Next chapter: **Deep Blue vs Kasparov (1997)** — IBM's chess computer defeats the reigning world champion, and why the brute-force search behind that victory wasn't really the "learning from data" breakthrough this chapter has been building toward.

♟ Deep Blue vs Kasparov

Chapter 1 closed with a promise that this chapter would complicate: a famous, headline-grabbing AI victory that had almost nothing to do with the statistical, learn-from-data turn just covered. Deep Blue's 1997 win over the reigning world chess champion is one of AI's most publicly celebrated moments — and one of the clearest cases in this course of a genuine achievement getting misread as something it wasn't.

🖥️ How Deep Blue Actually Worked

IBM's **Deep Blue** was a massively parallel supercomputer built around custom "chess chips," capable of evaluating roughly **200 million chess positions per second**. Its core technique was **brute-force minimax search with alpha-beta pruning** — systematically looking ahead through the tree of possible future moves, several moves deeper than any human could consciously track, and pruning away branches that couldn't possibly affect the outcome. Grandmaster **Joel Benjamin** helped hand-tune its position-evaluation function (the formula scoring how good a given board looks), and the system was loaded with a large database of opening moves and endgame tablebases covering positions with few pieces remaining.

Notice what's *not* in that description: no learning from a dataset of past games, no statistical inference, none of Chapter 1's machine learning techniques. Deep Blue's chess knowledge was searched and hand-tuned, not learned — in spirit, it's a direct descendant of Course 2 Chapter 3's Logic Theorist and General Problem Solver, doing tree search over a well-defined problem space, just with four decades of computing power behind it instead of 1950s hardware.

May 11, 1997

Deep Blue defeated **Garry Kasparov**, the reigning world chess champion, in a six-game rematch (Kasparov had won their first match in 1996), with a final score of 3.5–2.5. It was the first time a computer had beaten a reigning world champion in a full match under standard tournament time controls — a genuinely historic, widely reported result that made front-page news around the world.

The Controversy

Kasparov later alleged that a mysteriously "too human" move IBM's system played in Game 2 suggested outside human intervention during the match. He formally requested a rematch; IBM declined and dismantled Deep Blue shortly afterward, rather than defending the result or allowing a follow-up match. The dispute was never conclusively resolved, and it left a lingering asterisk over an otherwise landmark achievement.

🧠 Moravec's Paradox

Deep Blue's victory is best understood through an insight roboticist **Hans Moravec** articulated in 1988, now known as **Moravec's Paradox**: tasks that feel intellectually demanding to humans — chess, formal logic, calculus — tend to be computationally cheap for machines, while tasks humans perform effortlessly and unconsciously — recognizing a face, walking across a room, understanding a spoken sentence in a noisy room — tend to be extraordinarily computationally expensive. Chess *felt* like the ultimate test of machine intelligence for decades precisely because it's hard for humans. For a sufficiently fast search algorithm, it turned out to be one of the more tractable problems available.

Why Many AI Researchers Shrugged

Chess has a small, fully-defined rule set, a fully observable board, no hidden information, and a state space that — while enormous — is small enough for search plus raw computing power to conquer. That's exactly the profile of a **closed, well-defined domain** that Course 2 Chapter 4's compare-table already flagged as symbolic search's genuine strength. Many contemporary AI researchers pointed out that Deep Blue's win, however impressive as engineering, demonstrated brute-force search scaling with hardware — not a new kind of machine understanding. It's the Mechanical Turk's question again, from Course 1 Chapter 3: a sufficiently convincing performance gets read as "real" intelligence by the public, regardless of what's actually happening underneath.

📜 Three Ways to Win at a Game

Approach	Knowledge Source	Where It Appears
Symbolic tree search	Hand-tuned evaluation function + brute-force lookahead	Logic Theorist / GPS (Course 2, Ch. 3); Deep Blue (this chapter)
Statistical learning	Patterns inferred from a fixed dataset	Bayesian networks, SVMs, statistical MT (Chapter 1)
Learning through self-play	A system generates its own training data by playing itself	Not yet covered — this is what makes Chapter 5's AlphaGo different from both

🔮 A Game Brute Force Couldn't Solve

Deep Blue's trick — throw enough hardware at exhaustive search — worked for chess because chess's state space, though vast, was small enough for 1997-era computing power to search deeply enough to win. It would not work nearly as well for the ancient game of Go, whose state space dwarfs chess by many orders of magnitude, making pure brute-force search hopeless no matter how much hardware you throw at it. Chapter 5 covers what it actually took to beat a top human at Go in 2016 — and the answer looks far more like Chapter 1's learning-from-data philosophy than this chapter's brute-force search.

🤔 Questions to Sit With

Reflection 1

Moravec's Paradox suggests our intuitions about what's "hard" for a machine are often backwards — chess fell decades before basic vision or common sense. Can you think of another task widely assumed to require "real intelligence" that might turn out to be more tractable than it looks, once you know what's actually happening underneath?

Reflection 2

IBM declined Kasparov's rematch request and dismantled Deep Blue immediately after winning. Does that decision change how you read the result — does refusing a chance to confirm the win under scrutiny make the achievement more or less convincing?

Reflection 3

Many AI researchers considered Deep Blue impressive engineering but not a meaningful step toward general intelligence, while the public largely read it as a landmark "machines beat humans" moment. When expert and public interpretations of an AI milestone diverge this sharply, whose read do you think history tends to remember — and should it?

🎯 What's Next

Next chapter: **The Neural Network Revival** — the backpropagation algorithm, and why deep learning had to wait years after the underlying math was already known for enough data and compute to finally catch up to it.

🧠 The Neural Network Revival

Chapter 1 covered statistical learning's rise, and Chapter 2 showed brute-force search winning at chess without any learning at all. This chapter turns to the technique now most associated with modern AI — artificial neural networks — and tells a story with an unusual shape: the core mathematical breakthrough arrived decades before the technique actually became useful, and almost nobody outside a small, persistent group of researchers thought it was worth the wait.

The Perceptron (1958) — And Its Limits

Psychologist **Frank Rosenblatt** introduced the **Perceptron** in 1958 — a simple, single-layer neural network loosely modeled on biological neurons, capable of learning to classify simple patterns by adjusting internal weights based on examples. The press coverage was, even by this course's standards, remarkably breathless: contemporary newspaper reports suggested the Perceptron would soon walk, talk, see, write, and even reproduce itself.

In 1969, **Marvin Minsky** and **Seymour Papert** — Minsky already familiar from Course 2's Cold War chapter and Course 1's science-fiction chapters — published *Perceptrons*, a rigorous mathematical analysis proving that a single-layer perceptron fundamentally **cannot** learn certain simple patterns, most famously the XOR (exclusive-or) function. This wasn't a training bug or an engineering limitation; it was a hard mathematical ceiling on what the architecture could ever represent, no matter how it was trained.

A Winter Within the Winters

Minsky and Papert's critique is widely credited with sharply reducing research interest and funding specifically for neural networks — sometimes informally called a "neural net winter," distinct from the two broader AI winters covered in Course 2. Notice the timeline: this 1969 pullback predates even the 1973 Lighthill Report (Course 2, Chapter 5). Neural networks fell out of favor a full four years before symbolic AI's first major funding collapse — a smaller, earlier tremor before the field's bigger quakes.

🔁 Backpropagation (1986) — The Answer, Decades Late

Minsky and Papert's proof applied specifically to single-layer networks. Add one or more **hidden layers** between input and output, with non-linear activation at each layer, and a network becomes mathematically capable of representing XOR and far more complex patterns besides. The missing piece was a practical way to *train* such a multi-layer network — and that's exactly what **David Rumelhart, Geoffrey Hinton, and Ronald Williams** supplied in their landmark 1986 paper, popularizing the **backpropagation** algorithm: using calculus's chain rule to efficiently compute how much each individual weight, at every layer, contributed to the network's final error, then adjusting every weight accordingly.

It's worth being precise about what backpropagation actually did: it directly, technically answered Minsky and Papert's 1969 critique — seventeen years later. The mathematical objection was real, and multi-layer networks trained by backpropagation were the real, working answer to it.

🏦 LeNet and Reading Checks

Backpropagation's promise wasn't purely theoretical. Through the late 1980s and 1990s, **Yann LeCun** developed **convolutional neural networks (CNNs)** — a specialized architecture well-suited to image data — culminating in **LeNet**, a handwritten digit recognizer trained with backpropagation. LeNet was good enough to be deployed commercially: banks in the United States used it to automatically read handwritten numbers on checks, a genuine, unglamorous, real-world deep learning success running quietly in production years before "deep learning" was a phrase anyone used.

😴 The Quiet Years

Working, But Not Winning

Despite backpropagation's mathematical soundness and LeNet's real deployment, neural networks spent most of the 1990s and early 2000s as a minority interest. On most benchmarks of the era, Chapter 1's support vector machines and other statistical methods matched or outperformed neural networks, while being easier to train and better understood theoretically. Deep networks — those with many hidden layers — were also notoriously difficult to train in practice, suffering from what's now called the **vanishing gradient problem**: the error signal backpropagation relies on tends to shrink to almost nothing by the time it reaches a network's earliest layers, making those layers barely learn at all.

🔓 2006 — Hinton Reignites Interest

In 2006, Geoffrey Hinton published work on **deep belief networks**, introducing a layer-by-layer unsupervised pretraining technique that made genuinely deep networks practical to train for the first time. This is generally cited as the moment the term **"deep learning"** entered wide use, and as the point where Hinton, LeCun, and **Yoshua Bengio** — later jointly dubbed the **"Godfathers of Deep Learning"** and awarded the 2018 Turing Award for this body of work — began to look less like holdouts on an unfashionable technique and more like researchers about to be proven right.

📜 What 1986 Had, and What It Didn't

Available in 1986	Still Missing
A mathematically sound training algorithm (backpropagation)	Enough labeled training data to make deep networks worth training
Proof that multi-layer networks could represent complex patterns	Enough raw computing power to train large networks in practical time
Working, deployed examples at small scale (LeNet)	Techniques to reliably train very deep networks (vanishing gradients)

🔮 The Math Was Ready. The World Wasn't.

This chapter's central irony deserves to be stated plainly: the algorithm that powers essentially every headline AI system built since has existed, in mathematically correct form, since 1986. What neural networks were actually waiting for, for roughly two more decades, wasn't a better idea — it was enough data and enough computing power to let that idea show what it could really do. Chapter 1 already flagged this exact pattern: the knowledge acquisition bottleneck didn't vanish with statistical learning, it just changed shape into a data-and-compute requirement. This chapter is that requirement's origin story. The next chapter is what happens the moment it's finally satisfied.

🤔 Questions to Sit With

Reflection 1

Backpropagation existed as working, correct math for two decades before it became the dominant AI technique. What does that gap tell you about how much "having the right idea" actually matters, versus having the resources to exploit it?

Reflection 2

Minsky and Papert's 1969 critique was mathematically correct, and it's credited with suppressing an entire research direction for years, even though the fix (hidden layers) was conceptually available the whole time. Is it fair to hold a single accurate critique responsible for a field-wide pullback, or does that responsibility belong more to how the field reacted to it?

Reflection 3

Hinton, LeCun, and Bengio kept working on neural networks through their least fashionable years and were eventually vindicated. Can you think of a similar case — in any field — where a small group's unfashionable, unrewarded persistence turned out to be right decades later?

🎯 What's Next

Next chapter: **ImageNet & AlexNet (2012)** — the moment a genuinely large labeled dataset and genuinely powerful hardware finally arrived at the same time, and deep learning went from a quiet, decades-long holdout position to the field's dominant approach almost overnight.

🖼️ ImageNet & AlexNet

Chapter 3 ended on a cliffhanger: backpropagation and convolutional networks were mathematically ready by the late 1980s, but the field spent roughly two more decades waiting for enough data and enough computing power to actually exploit them. This chapter is the moment that wait ended — abruptly, publicly, and by a margin nobody in the field was prepared for.

📚 Building ImageNet

Starting around 2007, **Fei-Fei Li** and collaborators at Princeton (later Stanford) set out to build something the field hadn't really had before: a genuinely massive, carefully labeled image dataset, organized according to the WordNet lexical hierarchy. The finished **ImageNet** dataset eventually contained more than **14 million labeled images** across roughly 20,000 categories — orders of magnitude larger than anything computer vision researchers had previously trained on.

Named After a Hoax, Powering a Revolution

Labeling 14 million images by hand was only feasible because Li's team used **Amazon Mechanical Turk** — the crowdsourcing platform Amazon deliberately named after Course 1 Chapter 3's famous 18th-century chess-playing hoax, precisely because both involve genuine human intelligence doing the real work behind an interface that makes it look automated. The historical Turk fooled audiences for eighty years by hiding a human operator inside a "mechanical" illusion. Its modern namesake openly hires human operators to do exactly the kind of labeling work computers of the era still couldn't do reliably themselves — and in doing so, it quietly built the dataset that would trigger deep learning's breakthrough.

🏆 The ILSVRC Challenge

Starting in 2010, an annual competition — the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** — used a 1.2-million-image, 1,000-category subset of ImageNet to benchmark image classification algorithms, continuing the benchmark-driven culture Chapter 1 traced back to the 1987 UCI Machine Learning Repository. Through 2010 and 2011, the best-performing systems relied on the era's standard computer vision toolkit: hand-crafted features like SIFT, fed into statistical classifiers such as Chapter 1's support vector machines. Progress was real but incremental — top entries hovered around a 25–26% top-5 error rate, improving by a percentage point or two each year.

AlexNet's 2012 Landslide

In 2012, a deep convolutional neural network built by **Alex Krizhevsky**, **Ilya Sutskever**, and **Geoffrey Hinton** — the same Hinton from Chapter 3's backpropagation paper and 2006 deep belief network revival — entered ILSVRC under the name **AlexNet**. It achieved a top-5 error rate of roughly **15.3%**, against a second-place entry's 26.2%. That's not an incremental improvement; it's a margin so large that most of the field's typical year-over-year gains for the entire history of the competition combined wouldn't have closed it.

AlexNet's architecture built directly on Yann LeCun's CNN work from Chapter 3, but three things made it practical at this new scale: **ReLU activation functions** (much faster to train than the traditional sigmoid/tanh functions), **dropout** (a technique that randomly disables parts of the network during training to prevent overfitting on so much data), and — critically — training on **two NVIDIA GTX 580 GPUs**, repurposing graphics hardware originally built for rendering video games into a massively parallel engine for the matrix math neural network training actually requires.

📜 Before AlexNet, After AlexNet

Before 2012	After 2012
Hand-crafted features (SIFT) + statistical classifiers (SVMs)	Features learned automatically inside a deep convolutional network
Training on standard CPUs	Training on GPUs, repurposed from graphics rendering
Incremental, percentage-point-a-year progress	A single ~11-point leap, unmatched by any prior year's combined gains

🧠 The Cliffhanger, Resolved

Chapter 3 closed by naming exactly what neural networks were still waiting for: enough data, and enough compute. This chapter shows both arriving at once — ImageNet supplied the data, GPUs supplied the compute, and the algorithm (backpropagation, 1986) and architecture (convolutional networks, LeCun, late 1980s–90s) had already been sitting ready for decades. AlexNet isn't really a new idea. It's Chapter 3's old idea, finally given the resources it needed to prove itself.

A Field Transformed Almost Overnight

How Fast the Pivot Actually Was

Within roughly two to three years of AlexNet's win, hand-crafted feature engineering — the dominant computer vision approach for over a decade — was largely abandoned across the research community in favor of deep convolutional networks. Hinton himself was hired by Google shortly afterward. Companies including Google, Facebook, and Microsoft moved quickly and aggressively into deep learning research and hiring. Few technical results in this course's history — including Deep Blue's 1997 chess win — triggered as fast and as complete a shift in what the entire field considered the obviously correct approach.

🤔 Questions to Sit With

Reflection 1

AlexNet's win came from combining old ideas (backpropagation, CNNs) with new resources (ImageNet, GPUs), not from a new theoretical breakthrough. Does that change how "revolutionary" the moment feels to you, compared to if the core algorithm itself had been newly invented in 2012?

Reflection 2

Course 2 Chapter 8 named a recurring hype-cycle pattern: bold promise, real but narrower delivery, a widening gap, then collapse. The deep learning boom that started here in 2012 is now over a decade old and still going. Does that make it fundamentally different from the expert systems boom, or is it simply a boom that hasn't reached its collapse phase yet?

Reflection 3

Modern Mechanical Turk workers did the invisible human labor that made ImageNet — and therefore AlexNet — possible, echoing the original Turk's hidden human operator. How much of today's AI progress do you think still depends on similarly invisible human labor, and how often does that labor get properly credited?

🎯 What's Next

Next chapter: **AlphaGo & Reinforcement Learning (2016)** — how DeepMind combined deep neural networks with a learning method that generates its own training data, to conquer a game whose state space made Chapter 2's brute-force chess approach hopeless.

AlphaGo & Reinforcement Learning

Chapter 2 flagged Go as a game Deep Blue's brute-force approach could never handle. Chapter 4 closed with deep learning's tools finally sharpened by real data and real compute. This chapter is where those two threads meet: a system that beats a world champion at the one board game search alone genuinely couldn't conquer, using a learning method fundamentally different from anything else covered so far in this course.

🌌 A Game Brute Force Couldn't Touch

Go is played on a 19×19 grid — compared to chess's 8×8 — and its number of legal board positions has been estimated at roughly $\mathbf{2{\times}10^{170}}$, a number that dwarfs chess's game-tree complexity (roughly 10^{120}) so completely that the comparison barely conveys the gap. Deep Blue's winning strategy — evaluate roughly 200 million positions per second and search deep enough to see the win coming — simply doesn't scale to a space this size, no matter how much hardware you throw at it. For decades, many AI researchers considered Go a benchmark that brute-force methods would never crack, and one that genuine machine intelligence — not just faster search — would be required to solve.

🏢 DeepMind

DeepMind, a UK AI research company founded in 2010 by **Demis Hassabis** and colleagues (acquired by Google in 2014), took on Go as a flagship research challenge. Their system, **AlphaGo**, combined three distinct ingredients that, together, sidestepped the brute-force wall entirely rather than trying to power through it.

🧩 Three Ingredients

AlphaGo paired two deep neural networks — descendants of Chapter 4's convolutional architectures — with a smarter search strategy. A **policy network** learned to predict which moves a strong player would consider, narrowing the search to plausible options instead of every legal move. A **value network** learned to estimate how good a given board position was, without needing to play the game out to the end. Both networks then guided **Monte Carlo Tree Search** — a search algorithm that explores a relatively small number of promising simulated game lines rather than exhaustively evaluating everything, the way Deep Blue's minimax search did.

🎮 What Is Reinforcement Learning?

AlphaGo's networks were trained using **reinforcement learning** — a third major learning paradigm, distinct from both approaches covered earlier in this course. Chapter 1's statistical methods and Chapter 4's ImageNet-trained networks learn from a fixed set of labeled examples (this image is a cat; this is the correct translation). Reinforcement learning instead has an agent take actions in an environment and receive rewards or penalties based on the outcome, gradually adjusting its behavior to maximize reward over time — no human needs to label the "correct" move for every possible board position, because the system discovers what works by playing.

AlphaGo was first trained on a database of human expert games (a supervised warm start), then improved dramatically through **self-play** — playing millions of games against itself, with each game's win or loss used as the reward signal to refine both networks further.

🏆 March 2016 — Lee Sedol

In March 2016, AlphaGo faced **Lee Sedol**, one of the world's top-ranked professional Go players, in a five-game match in Seoul, South Korea, broadcast live to a global audience. AlphaGo won **4 games to 1**. Many AI researchers had publicly predicted, only a year or two earlier, that a computer beating a top human at Go was still a decade or more away — making the timing itself, not just the result, a genuine shock to the field's own expectations.

Move 37

In Game 2, AlphaGo played a move so unconventional that professional commentators initially assumed it was an error — a move no human player of Lee Sedol's caliber would seriously consider. It turned out to be a brilliant, winning move. "Move 37" is now widely cited as a moment where an AI system appeared to demonstrate something closer to creative insight than raw calculation — a striking contrast to Chapter 2's dismissal of Deep Blue as "just brute force, not real intelligence."

🔁 AlphaGo Zero and AlphaZero

DeepMind pushed the underlying idea even further in 2017 with **AlphaGo Zero**, which learned to play Go entirely through self-play, starting from nothing but the rules of the game — no human game data at all — and surpassed the original Lee-Sedol-beating AlphaGo within days of training. A generalized version, **AlphaZero**, applied the same self-play approach to chess and shogi, and decisively defeated the world's strongest existing chess engines. That's a genuine full-circle moment for this course: the same game Deep Blue conquered through raw brute-force search in 1997 fell again, twenty years later, to a system that had never been taught a single human chess strategy — it simply played against itself until it discovered better ones.

📜 Two Ways to Beat a World Champion

Deep Blue (Chapter 2, 1997)	AlphaGo (This Chapter, 2016)
Exhaustive brute-force search over the full game tree	Narrow, guided search over only the most promising lines
Hand-tuned evaluation function (grandmaster-assisted)	Learned evaluation function (value network, trained by self-play)
Works because chess's state space is small enough to search	Works despite Go's state space being far too large to search exhaustively

🔮 A New Kind of Learning, Not Just a Bigger Network

It's worth being precise about what actually changed here. AlexNet (Chapter 4) proved deep networks could learn extraordinarily well from a large fixed dataset. AlphaGo proved something different: a system can learn to master a task with no complete "correct answer" dataset available at all, by generating its own experience and improving from it. That distinction — learning from a fixed dataset versus learning by interacting with an environment — turns out to matter enormously for the next architecture this course covers, which didn't come from reinforcement learning at all, but from a different rethink of how neural networks should process sequences of information.

🤔 Questions to Sit With

Reflection 1

"Move 37" is often described as looking creative rather than merely calculated. Having read Chapter 2's skepticism about calling Deep Blue's win "real" intelligence, do you find AlphaGo's case genuinely different, or is this the same convincing-performance question in a new costume?

Reflection 2

AlphaGo Zero surpassed the human-trained AlphaGo by learning purely from self-play, with no human game data at all. What does it suggest about human expertise — in Go or elsewhere — that a system improved faster once it stopped learning from us?

Reflection 3

Go fell to AI roughly a decade earlier than most experts predicted. Thinking back to Course 2's recurring theme of overconfident timelines (the Dartmouth Workshop's two-month promise, in particular) — does an expert prediction being wrong in the *optimistic* direction feel like a different kind of mistake than being wrong in the overconfident direction, or is it the same misjudgment either way?

🎯 What's Next

Next chapter: **Transformers** — the 2017 "Attention Is All You Need" paper, and the architecture shift that would go on to power every major language model that followed.

🔍 Transformers

Chapter 5 closed by pointing away from reinforcement learning toward a different rethink — this time of how neural networks should process sequences of information at all, rather than isolated images or board positions. This chapter covers the single architectural idea that underlies essentially every major language model built since, introduced with one of the boldest paper titles in this course's history.

🔄 The RNN/LSTM Bottleneck

Before this chapter, sequence tasks like machine translation or language modeling were dominated by **recurrent neural networks (RNNs)** and their more capable successor, the **LSTM (Long Short-Term Memory)** network, introduced by Hochreiter and Schmidhuber back in 1997. Both process a sequence one element at a time — reading a sentence word by word, carrying forward a running internal state — which creates two real problems. First, that step-by-step processing can't be parallelized across the sequence during training, making it inherently slow on the GPU hardware Chapter 4 showed off. Second, information from early in a long sequence tends to fade by the time the network reaches the end, a variant of Chapter 3's vanishing gradient problem showing up again in a new form.

👀 Attention, Before Transformers

In 2014, researchers including Dzmitry Bahdanau introduced an **attention mechanism** as an add-on to RNN-based translation models — instead of forcing the entire input sentence into one fixed-length summary vector, the model learned to "attend" to the most relevant input words while producing each output word. It was a genuine improvement to translation quality, but still bolted onto the same slow, sequential RNN backbone.

⚡ "Attention Is All You Need" (2017)

A team of researchers at Google — Ashish Vaswani, Noam Shazeer, and colleagues — took attention's role a step further and published a paper with a title that stated its own thesis outright: **"Attention Is All You Need."** Their proposed architecture, the **Transformer**, discarded recurrence entirely. No RNN, no LSTM, no step-by-step processing at all — just **self-attention** layers and feedforward layers, applied to the whole sequence at once.

Removing recurrence had an immediate practical payoff: because every position in the sequence could now be processed simultaneously rather than one after another, Transformers could be trained far more efficiently on the parallel GPU hardware that had already proven decisive in Chapter 4. Bigger models, trained on bigger datasets, in less time, became practical almost immediately.

🧩 How Self-Attention Works

At a high level, **self-attention** lets every element in a sequence directly compute how relevant every other element is to it, regardless of distance — a word at the start of a paragraph can influence a word at the end just as directly as its immediate neighbor, solving the long-range dependency problem RNNs struggled with. **Multi-head attention** runs several of these relevance computations in parallel, each free to specialize in a different kind of relationship (grammatical structure, topic relevance, and so on). And because dropping recurrence also means the model has no inherent sense of word order, Transformers add explicit **positional encoding** — extra information injected into each input telling the model where in the sequence it sits.

Proven First on the Old Battleground

The Transformer's debut benchmark was, fittingly, machine translation — the same task whose 1966 rule-based failure (Course 2, Chapter 5's ALPAC Report) triggered AI's first funding winter, and whose 1990s statistical revival (this course's Chapter 1) gave that story its redemption arc. Transformers set new state-of-the-art results on standard English-German and English-French translation benchmarks, while training substantially faster than the RNN-based systems they replaced — a third chapter, decades apart, in the same task's ongoing story.

📜 Sequence Models, Before and After

RNN / LSTM (Before)	Transformer (After)
Processes a sequence step by step, in order	Processes the entire sequence simultaneously
Training is inherently sequential — hard to parallelize	Highly parallelizable — trains efficiently on GPU hardware
Long-range relationships fade over distance	Self-attention connects any two positions directly, regardless of distance

🔮 A Uniquely General Architecture

Almost every technique covered so far in this course was built for one kind of problem — SVMs and Bayesian networks for statistical classification, convolutional networks for images, Monte Carlo Tree Search for board games. The Transformer turned out to be startlingly general-purpose: the same core architecture went on to power vision models (Vision Transformers, applied to images rather than text), DeepMind's AlphaFold (protein structure prediction — the same research lab from Chapter 5's AlphaGo), audio processing, and, most consequentially for the next chapter, large language models. Very few ideas in this course's history generalized this far beyond the problem they were originally built to solve.

🤔 Questions to Sit With

Reflection 1

"Attention Is All You Need" is a notably confident title for a research paper. Having tracked this course's recurring pattern of bold claims and their consequences (the Dartmouth proposal, expert systems' commercial promises), does a title this bold make you more skeptical going in, or did the Transformer's actual impact simply earn it?

Reflection 2

The Transformer solved RNNs' long-range dependency problem by letting every position attend directly to every other position, rather than passing information step by step. Can you think of a human process or organization that struggles with the same "information fades over distance" problem RNNs had?

Reflection 3

Unlike most breakthroughs in this course, the Transformer turned out to generalize far beyond its original task. Do you think that generality was a foreseeable consequence of the architecture's design, or a genuine surprise even to the researchers who built it?

🎯 What's Next

Next chapter: **The GPT Lineage & the ChatGPT Moment** — how this chapter's architecture became the foundation for large language models, and the specific product moment in 2022 that put them in front of hundreds of millions of ordinary people almost overnight.

The GPT Lineage & the ChatGPT Moment

Chapter 6 ended by pointing to language models as the Transformer's most consequential application. This chapter follows that thread from a research paper to a product nearly a billion people would eventually use directly — and pays close attention to the gap between when the underlying capability actually existed and when the wider world actually noticed.

🏢 OpenAI

OpenAI, founded in 2015, built the model lineage that would eventually become ChatGPT directly on top of Chapter 6's Transformer architecture — specifically, a stack of Transformer decoder layers trained to predict the next word in a sequence, given everything that came before it.

📈 GPT-1 to GPT-3 — Scaling the Same Idea

GPT-1 (2018) demonstrated that unsupervised pre-training on a large body of text, followed by task-specific fine-tuning, worked remarkably well across a range of language tasks — a relatively modest 117-million-parameter model by later standards. **GPT-2 (2019)** scaled that same core idea up to 1.5 billion parameters and showed a noticeably more fluent, coherent writing style.

GPT-2's Withheld Release

OpenAI initially declined to release GPT-2's full model weights, citing concern that the model was fluent enough to generate convincing misinformation or spam at scale — a genuinely unusual moment of a research lab publicly withholding its own work over anticipated misuse, rather than releasing it and letting the field sort out consequences afterward. The full model was eventually released in stages over the following months, but the decision itself sparked real debate about how AI labs should weigh openness against potential harm — a debate this course's final chapter picks back up.

GPT-3 (2020) represented a genuine scale leap: 175 billion parameters, more than a hundred times larger than GPT-2. It demonstrated **few-shot learning** — the ability to perform a new task correctly from just a handful of examples given directly in the prompt, with no additional training required. Access was limited to an API rather than a public release or open weights, reflecting both the model's scale and OpenAI's shift toward a commercial product.

📐 Scaling Laws

In 2020, OpenAI researchers including Jared Kaplan published empirical **scaling laws**: model performance improved smoothly and predictably as model size, dataset size, and compute were all increased together, with no sign of the improvements running out at the scales tested. This finding is, in effect, the formal, quantified version of the "data and compute" throughline running through this entire course since Chapter 3 — and it gave the industry a genuine, evidence-based justification for investing enormous sums into training ever-larger models.

🎯 RLHF — Teaching a Model to Be Helpful

A raw GPT-3-style model is trained to predict plausible next words — it isn't inherently trained to be helpful, honest, or safe to talk to. **InstructGPT**, published in early 2022, closed that gap using **Reinforcement Learning from Human Feedback (RLHF)**: human raters ranked different model outputs by quality, and those rankings were used as a reward signal to fine-tune the model's behavior toward what people actually found useful. This is Chapter 5's reinforcement learning showing up again, in a very different role — not learning to win a board game through self-play, but learning to align with human preferences through human judgment.

🚀 November 30, 2022

OpenAI released **ChatGPT** — a free, public, conversational web interface built on a GPT-3.5 model refined with RLHF along the InstructGPT lineage. It reached **one million users within five days** and roughly **100 million monthly active users within about two months**, making it, at the time, the fastest-growing consumer application in history.

A Product Moment as Much as a Research Moment

Here's the detail worth sitting with: GPT-3, a model with broadly comparable underlying capability, had already existed for **two full years** before ChatGPT's release. What actually changed on November 30, 2022 wasn't primarily a research breakthrough — it was RLHF's conversational polish combined with a free, simple, chat-style interface that required no technical knowledge or API access to use. The capability was largely already there. What ChatGPT actually delivered was *access* — and the gap between "a lab has a capable model" and "hundreds of millions of people can feel what that capability is like" turned out to matter enormously, echoing this course's recurring theme (the Mechanical Turk, Deep Blue) about how much public perception depends on a convincing, accessible performance rather than on the underlying capability alone.

📜 Two Years, Same Core Capability

GPT-3 (2020)	ChatGPT (2022)
Comparable core language capability	Same lineage, refined with RLHF for conversational behavior
Developer-only access via a paid API	Free, public, web-based chat interface
Relatively little mainstream public awareness	~100 million monthly users within roughly two months

🔮 The Race That Followed

ChatGPT's public visibility triggered an industry-wide scramble that this course has seen the shape of before: Google rushed out its own competing chatbot, Microsoft invested billions of dollars into OpenAI and rebuilt Bing around the same underlying technology, and Anthropic — founded in 2021 by former OpenAI researchers — released its own model, Claude. It's a genuinely bigger, faster, more capital-intensive race than Course 2's expert-systems boom or Japan's Fifth Generation Project ever were. Whether this round follows the same recurring hype-cycle shape Course 2 named — and if so, what a correction would even look like at this scale — is exactly where this course's final chapter picks up.

🤔 Questions to Sit With

Reflection 1

GPT-3's capability existed two years before ChatGPT made it famous. If a comparably capable model appeared today but only through an obscure developer API, do you think it could stay largely unnoticed by the public the way GPT-3 initially did — or has that gap between "capable" and "known" gotten permanently smaller?

Reflection 2

OpenAI initially withheld GPT-2 over misuse concerns, then released later, larger, more capable models far more openly. What do you make of that shift — does it suggest the initial caution was overblown, or that competitive and commercial pressure can erode safety caution over time regardless of whether the underlying concern was valid?

Reflection 3

RLHF uses human preference judgments to shape model behavior — a very different role for reinforcement learning than Chapter 5's self-play, where the "reward" was an objective win or loss. What are the risks of optimizing a system toward what human raters *prefer*, as opposed to what's objectively correct or true?

🎯 What's Next

Next chapter — the final chapter of this course: **Where We Are Now** — diffusion models, multimodal AI, and a light-touch look at the AGI and alignment debate, covering ground that, unlike everything before it in this course, is still actively unfolding.

🌅 Where We Are Now

Every chapter before this one had the benefit of hindsight — a settled result, a clear winner, a verdict history had already reached. This final chapter doesn't have that luxury. It covers ground that's still actively moving, so treat everything here as a snapshot of a genuinely open moment, not a closed case the way Deep Blue's match or AlexNet's benchmark score were.

🎨 Diffusion Models

Image generation took its own path to the same "learn from data" destination this course has been tracing. A **diffusion model** is trained by taking real images and gradually adding random noise until nothing recognizable remains, then learning to reverse that process, step by step, turning noise back into a coherent image. To generate something new, the model starts from pure random noise and iteratively denoises it — often guided by a text prompt, using cross-attention layers borrowed directly from Chapter 6's Transformer architecture to connect the words of a description to the visual content being formed.

DALL-E 2 (OpenAI, April 2022), **Stable Diffusion** (Stability AI, August 2022), and **Midjourney** each brought text-to-image generation to a wide audience within months of each other.

Image Generation Had Its Own Moment First

DALL-E 2 launched roughly seven months before ChatGPT, and Stable Diffusion's release was notably **open-source** — a sharp contrast to GPT-3's API-only access (Chapter 7). Both generated real public excitement. But neither matched ChatGPT's scale or speed of adoption, reinforcing Chapter 7's core lesson: a free, simple, conversational text interface turned out to resonate more broadly and immediately than an equally impressive image-generation capability did, even though the underlying "learn to reverse a data-generating process" idea was just as genuine an advance.

🧩 Multimodal AI

By 2023, models including **GPT-4** added the ability to process images alongside text within the same system, and subsequent models moved toward handling text, images, audio, and video together natively, rather than stitching together separate specialized systems for each. This is Chapter 6's closing observation about the Transformer's unusual generality finally paying off at full scale: one underlying architecture, extended to essentially any kind of sequential data, rather than a different bespoke technique for every modality the way Course 2's expert systems each required their own hand-built knowledge base.

🤔 The AGI Question

Every system covered across this entire course — Deep Blue, AlexNet, AlphaGo, GPT-3 — is **narrow**: built for, and good at, a specific class of task. **Artificial General Intelligence (AGI)** refers to a system with human-level or greater capability across essentially any cognitive task, not just one domain. It's a genuinely contested, poorly-defined term — researchers disagree sharply about what would actually count as AGI, whether current architectures (Transformers, scaled up further per Chapter 7's scaling laws) are a real path toward it, or whether a fundamentally different approach will be required, and whether it's decades away, imminent, or effectively unreachable in any meaningful sense.

⚖️ The Alignment Debate

Alignment research asks a question this entire trilogy has actually been building toward since its very first chapter: how do you ensure an increasingly capable AI system actually pursues the goals its creators intended, and behaves safely as its capability scales? Course 1's fiction — HAL 9000's misspecified objective, Skynet's instrumental resistance to shutdown, the Three Laws of Robotics failing the moment anyone tried to stress-test them as real engineering — wasn't just decoration. It was this exact question, worked out in narrative form, decades before today's systems made it a live engineering concern rather than a thought experiment.

A Genuine, Unsettled Disagreement

In 2023, a short public statement warning that AI could pose an extinction-level risk comparable to pandemics or nuclear war was signed by a substantial number of prominent AI researchers and lab leaders — including figures whose companies were simultaneously racing to build more capable systems. Other researchers, equally credentialed, consider existential-risk framing overblown, view current systems' actual capabilities and failure modes as far more mundane, and note — echoing Course 2 Chapter 8's theme about incentives shaping claims — that dramatic risk framing can itself function as a form of marketing, implying a company's technology is powerful enough to be dangerous. Both positions are held by serious researchers acting in apparent good faith. This course won't resolve that disagreement, because it isn't resolved.

📜 This Course, at a Glance

Chapter	Event	Idea That Carried Forward
1	The Statistical Turn	Learning from data, not hand-coded rules — the bottleneck moves, doesn't vanish
2	Deep Blue vs Kasparov	Brute-force search wins closed domains; it doesn't generalize
3	The Neural Network Revival	Correct math (1986) waiting decades for data and compute to catch up
4	ImageNet & AlexNet	Data + compute + algorithm, finally converging — the field pivots almost overnight
5	AlphaGo & Reinforcement Learning	Learning by self-generated experience, not just a fixed dataset
6	Transformers	One architecture, unusually general across tasks and modalities
7	The GPT Lineage & the ChatGPT Moment	Capability and public awareness of it can diverge for years

🧿 Three Courses, One Long Argument

Zoom all the way out and this entire project has been making one sustained argument in three parts. Course 1 showed that centuries of myth and fiction anticipated AI's real problems — deceptive performance, creators losing control, rules that sound complete but aren't — long before anyone could build a system to actually have them. Course 2 showed the real field repeatedly overpromising, hitting the same hand-encoded-knowledge wall in two different disguises, and collapsing both times. Course 3 showed a genuinely different method — learning from data and experience instead of hand-coding it — finally break through that wall, at a speed and scale nothing earlier in this course's history approached. Whether that breakthrough eventually repeats Course 2's hype-cycle pattern at a larger scale, whether Course 1's fictional warnings turn out to be genuinely prophetic or just resonant stories, and whether AGI is a coherent near-term goal or a moving target — none of that is settled. It's the actual, live version of every question this course has been asking in historical form.

🤔 Questions to Sit With

Reflection 1

Course 1 framed decades of myth and fiction as anticipating real AI safety problems. Having now covered the real technical history, do you think that fiction actually prepared researchers and the public to think clearly about alignment — or did it mostly supply dramatic imagery (HAL, Skynet) that makes calm, precise discussion of real risks harder?

Reflection 2

Serious researchers disagree sharply about whether current AI poses existential risk, and Course 2 taught you to notice how incentives shape confident claims in both directions. Whose incentives do you trust more when evaluating this specific debate — and why?

Reflection 3

Looking back across all three courses: which single moment do you think most changed the actual trajectory of AI — not the most famous or dramatic, but the one that, if it had gone differently, would have changed everything that came after it?

🎯 The End of This Story, For Now

That completes **History of AI** — three courses, twenty-four chapters, from Talos and the Golem through Turing, two winters, and the ChatGPT moment. Course 3 (The Statistical & Deep Learning Era) is now complete. No Course 4 is currently planned — this last chapter's subject matter is, quite literally, still being written. If the field's next chapter turns out to be worth its own course later, it'll be waiting here.