Trellis Studio

The Brain and the Desk.

Lesson 1: Intro to Large Language Models

This is the first lesson of a nine-lesson course on how AI assistants actually work. It takes about nine minutes, and it assumes nothing: no technical background, no prior reading.

BY THE END OF THIS LESSON, YOU WILL BE ABLE TO

1. Explain what a large language model is.
2. Say what happens when a model is "thinking", and what thinking effort controls.
3. Name the major model providers, the apps they power, and the models each one ships.
4. Describe the two phases of training: pre-training and fine-tuning.
5. Explain what a harness is, and why it decides the results you get.

CONTACT

trellisstudio.co/ai-training · hello@trellisstudio.co

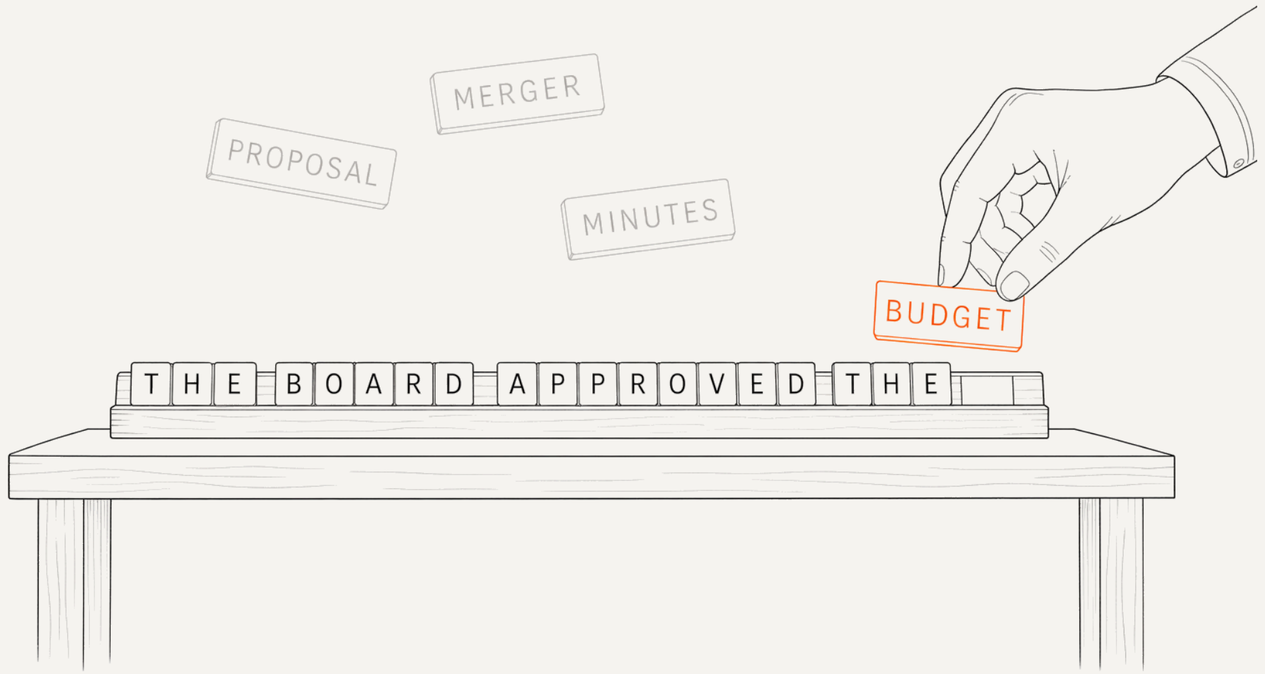
You already use a language model every day. It lives in your keyboard.

The auto-complete on your iMessage or texting app is a tiny language model. An **LLM**, or large language model, works in the same way, just with much better prediction.

That prediction quality is expensive. To train an LLM, you need:

- **Data:** hundreds of thousands of books, petabytes in total (1 PB = 1,000 TB).
- **Hardware:** GPUs, which are the chips that do the maths, and a great deal of RAM to feed them.
- **Time:** the equivalent of hundreds to tens of thousands of hours of training.

> **One idea to hold from the start: everything an LLM does, it does by predicting the next word. Even, as the next slide shows, its "thinking".**



IT WEIGHS EVERY WORD IT KNOWS, THEN PICKS ONE. THEN IT DOES THAT AGAIN.

If you predict well enough, you can call it thinking.

These models do not actually *think*. When a product shows the model "thinking", it is doing more of the same prediction, privately:

- Before writing the answer you see, the model predicts a hidden *scratchpad* of working-out.
- In the scratchpad, it *tries* an approach in words, *checks* it in words, and *discards* it in words.
- Then it predicts the visible answer with that scratchpad in front of it.

Thinking effort is the setting that controls how much scratchpad the model may write before it must answer:

- High effort means a longer private scratchpad, and usually a *better answer*.
- It also costs more time and money, because every scratchpad word is a predicted word, and each new predicted word is measured and priced as sub-words called **tokens**.

> **This is why we say the model simulates thinking: the mechanism is still next-word prediction, only more of it.**

YOU ASK

Is 17 × 24 bigger than 400?

HIDDEN SCRATCHPAD · PREDICTED FIRST, NEVER SHOWN

17 × 24. Try 17 × 20 = 340. Then 17 × 4 = 68. 340 + 68 = 408. Check: 408 is greater than 400. So the answer is yes.

THE ANSWER YOU SEE · PREDICTED FROM THE SCRATCHPAD

Yes. 17 × 24 = 408, which is just over 400.

EVERY LINE HERE, HIDDEN OR VISIBLE, IS NEXT-WORD PREDICTION. "THINKING" IS THE HIDDEN PART.

THE TOKEN MATHS

- A token is a sub-word. "Predicting" is two tokens: `predict` + `ing`. As a rule of thumb, 100 words is about 133 tokens.
- Providers price per million tokens. Claude Opus 5, for example: about $25 per million tokens it writes, $5 per million it reads.
- For scale: all seven Harry Potter books are about 1.1 million words, or roughly 1.4 million tokens. Writing them would cost about $36. Reading them, about $7.
- One hard question at the highest thinking effort can use around 200,000 tokens. That is roughly one volume of The Lord of the Rings, mostly scratchpad, predicted and thrown away.

All prices in US dollars. Prices: Anthropic's published API rates for Claude Opus 5, 2026 (platform.claude.com). The 200,000-token task: Anthropic, agentic coding performance by effort level, Opus 4.7 at max effort. Word counts for the book series are the commonly cited totals; token counts use the 100-words-to-133-tokens rule of thumb.

Who makes the models, what you type into, and what it costs.

The companies that train models are called **model providers**. The app you type into is a separate thing from the model, and it is called the **harness**.

Notice in the table that some companies make both, and one very large company makes only harnesses.

PROVIDER	HARNESS (THE APP YOU TYPE INTO)	CURRENT AND RECENT MODELS	COST OF THE FLAGSHIP, PER MILLION TOKENS
Cohere	North	Command A, Command R+	Command A: about $10 written / $2.50 read
Moonshot	Kimi	Kimi K2	K2: about $2.50 written / $0.60 read; the weights are open
Alibaba	Qwen app	Qwen 3	Open weights: free to download and run yourself
OpenAI	ChatGPT app	GPT-Luna, Terra, Sol; GPT-5.6, 5.5	Sol: about $20 written / $4 read
Anthropic	Claude app · Claude Code · Claude Cowork	Fable 5; Opus 5, Sonnet 5, Haiku 4.5	Opus 5: $25 written / $5 read
Google	Gemini app · Gemini in Gmail and Docs	Gemini 3.1 Pro, Gemini 3.1 Flash	3.1 Pro: $12 written / $2 read
Microsoft	Microsoft Copilot · GitHub Copilot · Copilot Studio	None of its own. Its harnesses serve OpenAI and Anthropic models	Copilot: $30 per person per month
Amazon	Amazon Q · Alexa+ · Bedrock (for builders)	Nova 2 Lite, Nova 2 Sonic	Nova 2 Lite: $2.50 written / $0.30 read
IBM	watsonx	Granite 4	Granite 4: from about $0.20 per million, sold with monthly minimums

A selection, not a catalogue; other providers include Thinking Machines (US) and MiniMax (China). Prices are the providers' published API rates for one flagship model, rounded, checked 27 August 2026; the Sol rate is promotional. Microsoft Copilot is a subscription, not a per-token rate.

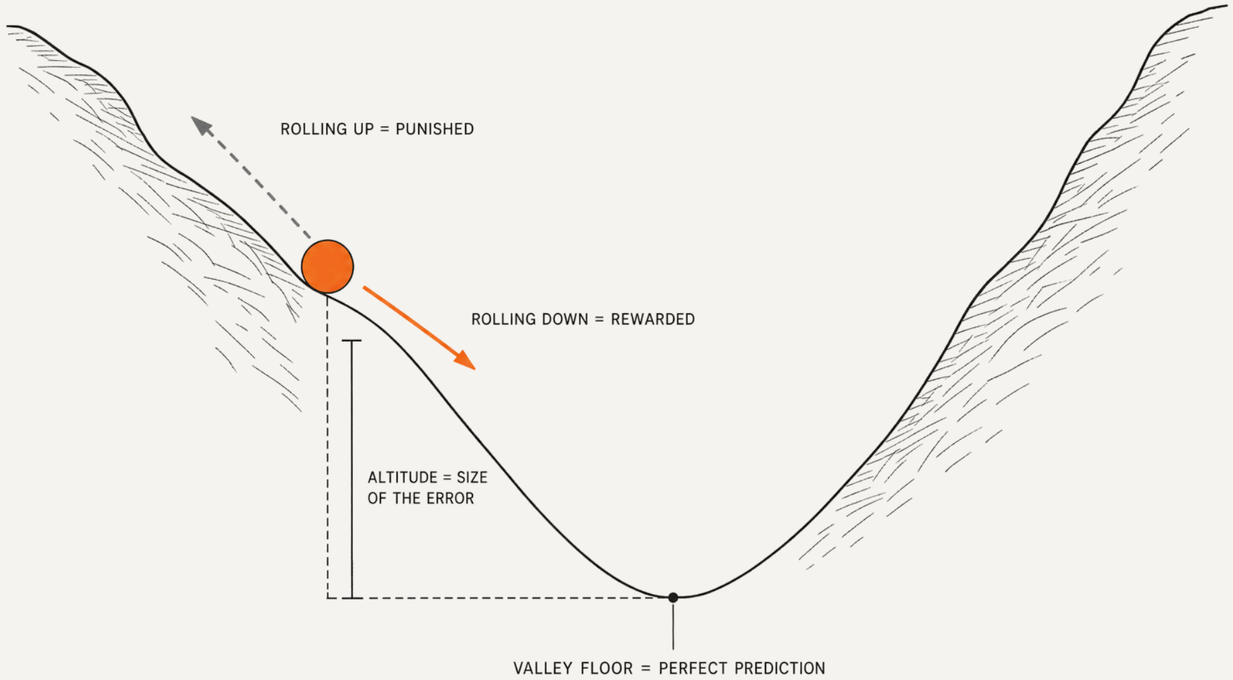
How it learns: a ball in the Grand Canyon.

Picture the Grand Canyon, or the Rockies, and imagine a giant ball rolling around in it:

- The *altitude* of the ball represents the error in the model's latest next-word prediction.
- The *distance* between the ball and the lowest point of the valley represents the size of that error.
- If the ball rolls *down* toward the valley floor, the predictions are getting better, and the model is rewarded.
- If the ball rolls *up* the mountain, the predictions are getting worse, and the model is punished.

This reward-and-punishment signal is called the **reward function**.

> **Training, everywhere in this course, means exactly one thing: letting the ball find its way further down.**



ONE PICTURE TO KEEP. EVERY TECHNIQUE THAT MENTIONS "TRAINING" MOVES THIS BALL.

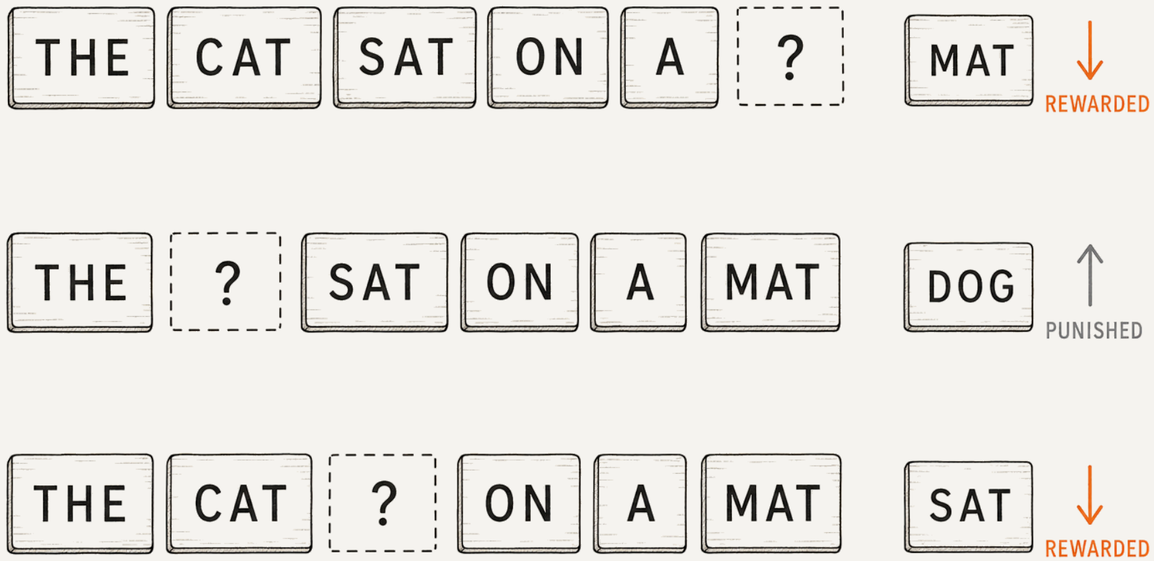
Pre-training: one drill, repeated across the internet.

Here is what the training drill looks like:

- We give the model the sentence "the cat sat on a mat" thousands of times, and each time we *hide one word*.
- The model predicts the hidden word. If it gets the word right, we reward it. If it gets the word wrong, we punish it.
- Now run that same drill, not on one sentence, but across *petabytes* of text. Slowly, the ball settles low in the valley.

This first phase is called **pre-training**, because providers first train their models on a large amount of general information.

> **Pre-training ends with a model that predicts the average of the internet extremely well.**



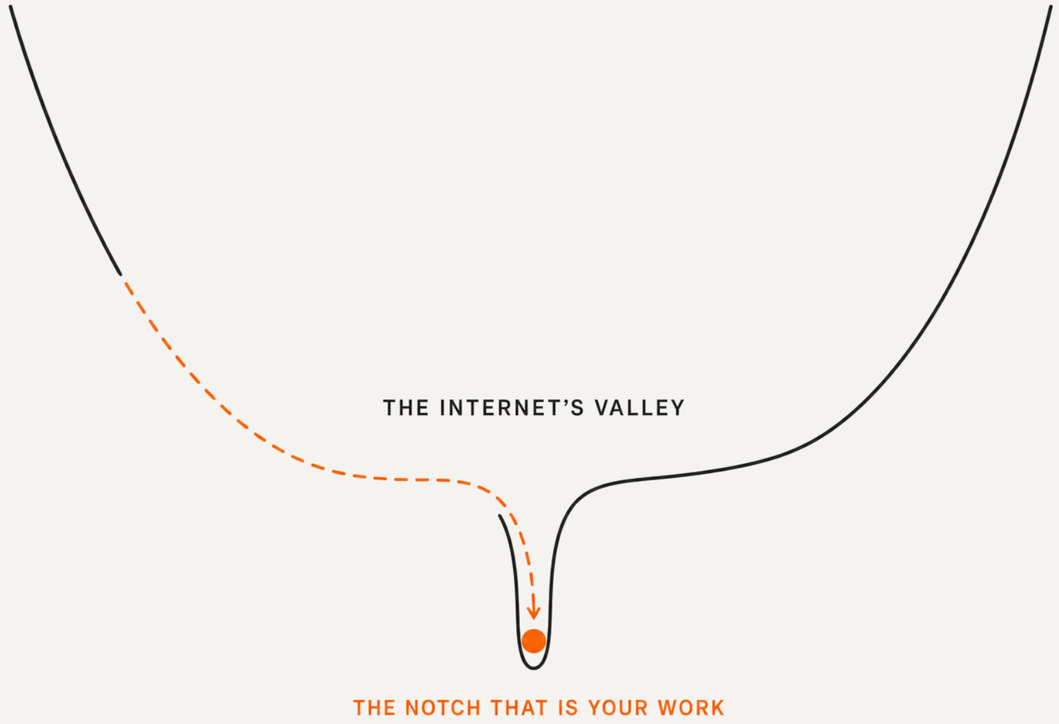
THREE ROUNDS OF THE THOUSANDS RUN ON THIS ONE SENTENCE ALONE.

Fine-tuning: the same ball, your valley.

After pre-training, the model still needs to predict what *your work* requires, not the average of the internet:

- We continue the training with examples from your own domain, and we adjust the reward function to encourage exactly that behaviour.
- The drill stays the same; only the sentences change. "The client signed the **[?]**" should now predict *"SOW"*, because that is what it is called in your work.
- The average of the internet would predict something vaguer, and now that costs the model a punishment.

This second phase is called post-training, or fine-tuning.



FINE-TUNING SETTLES THE BALL INTO THE NARROW NOTCH THAT IS YOUR WORK.

Fine-tuning has named techniques, such as PEFT, SFT, DPO, PPO, RLHF, and RLVR. We will cover them much later in the course; every one of them moves the same ball.

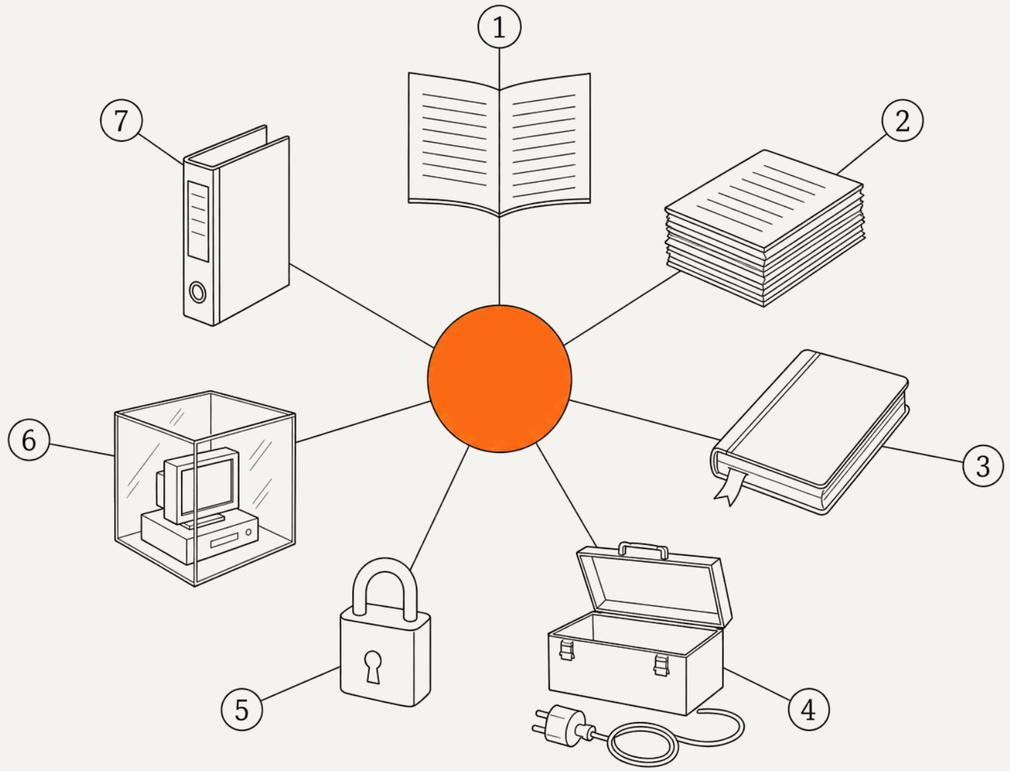
The model is the auto-complete. The app around it makes it useful.

On its own, the trained model is still just a very good auto-complete. The app around it is where the usefulness comes from:

- The app you actually use, whether that is Copilot, the ChatGPT app, Claude chat, or Claude Code, is *wrapped around* the model.
- The wrapper works by giving the model the right task-relevant information at the right moment.
- It is like handing a new concept to someone in the middle of a speech: they have to work it into what they are saying. The model has to incorporate whatever the wrapper puts in front of it.

This wrapper is called the **harness**.

> **The same model inside a different harness will give you very different results.**



THE SEVEN PARTS OF A HARNESS.

Lessons 3 to 6 take the harness apart piece by piece: what the model can see, how it reaches your systems, how it runs code, and what keeps all of that safe.

Check yourself before the answer.

In the middle of a conversation, you correct your assistant. It apologizes, and it gets things right for the rest of the chat. What just happened?

A Your correction retrained the model. The ball rolled.

B The ball did not move. The harness placed your correction in front of the model, and the prediction now takes it into account.

C The provider fine-tuned the model on your conversation overnight.

ANSWER

The answer is **B**. Training only happens during pre-training and fine-tuning, and those run on the provider's hardware over enormous sets of examples. Your correction never touched the ball. Instead, it became task-relevant information that the harness holds in front of the model, which is exactly the person-in-mid-speech situation from the last slide. That is also why the correction is gone tomorrow: a fresh chat starts empty.

Auto-complete, a ball in a valley, and a wrapper.

Five sentences to keep:

- An LLM is the auto-complete from your keyboard, running at an extraordinary scale.
- "Thinking" is more of the same prediction, done privately on a scratchpad, and every predicted word is priced in *tokens*.
- A handful of providers make the models. The app you type into, the *harness*, is a separate thing, and sometimes a separate company.
- Training is a ball finding the floor of an error valley: first the internet's valley, which is *pre-training*, then the notch that is your work, which is *fine-tuning*.
- The same model inside a different harness will give you very different results.

TRY IT IN YOUR OWN TOOL, TODAY

Ask your assistant a question that needs a long answer, and watch how the reply arrives. It appears word by word, because it is being predicted word by word. If your tool shows a "thinking" indicator first, you now know what is happening behind it: a private scratchpad, also arriving word by word.

Taught live, for your team.

This lesson is the first of nine in The Brain and the Desk, Trellis Studio's course on how AI assistants actually work.

We also teach it live: corporate training for whole departments, not only the technical ones, from these foundations through to building the harness for your team's actual work, on Copilot or on Claude. Cohorts run 8 to 20 people, as a one-day intensive or four weekly sessions, on site in the Toronto area or remote.

> **If that would help, we are one conversation away.**

trellisstudio.co/ai-training · hello@trellisstudio.co

Next, Lesson 2: Small brain or big brain, quick or slow. Providers ship each model in several sizes, and the thinking-effort setting decides how long one may work before it answers. You will learn how to pick the right size and setting for a real task, from the charts the providers publish.