Artificial intelligence

What are we talking to when we talk to ChatGPT & Co.?

Kaspar Etter · 26 April 2026 ·

Outline

- Agents and intelligence
- Learning as compression
- Artificial neural networks
- Large language models
- Tips for using LLMs

Acknowledgment

These slides were created with help from GPT-5.5.
The images were generated by ChatGPT Images 2.0.

Scope

This talk is mostly about large language models (LLMs).

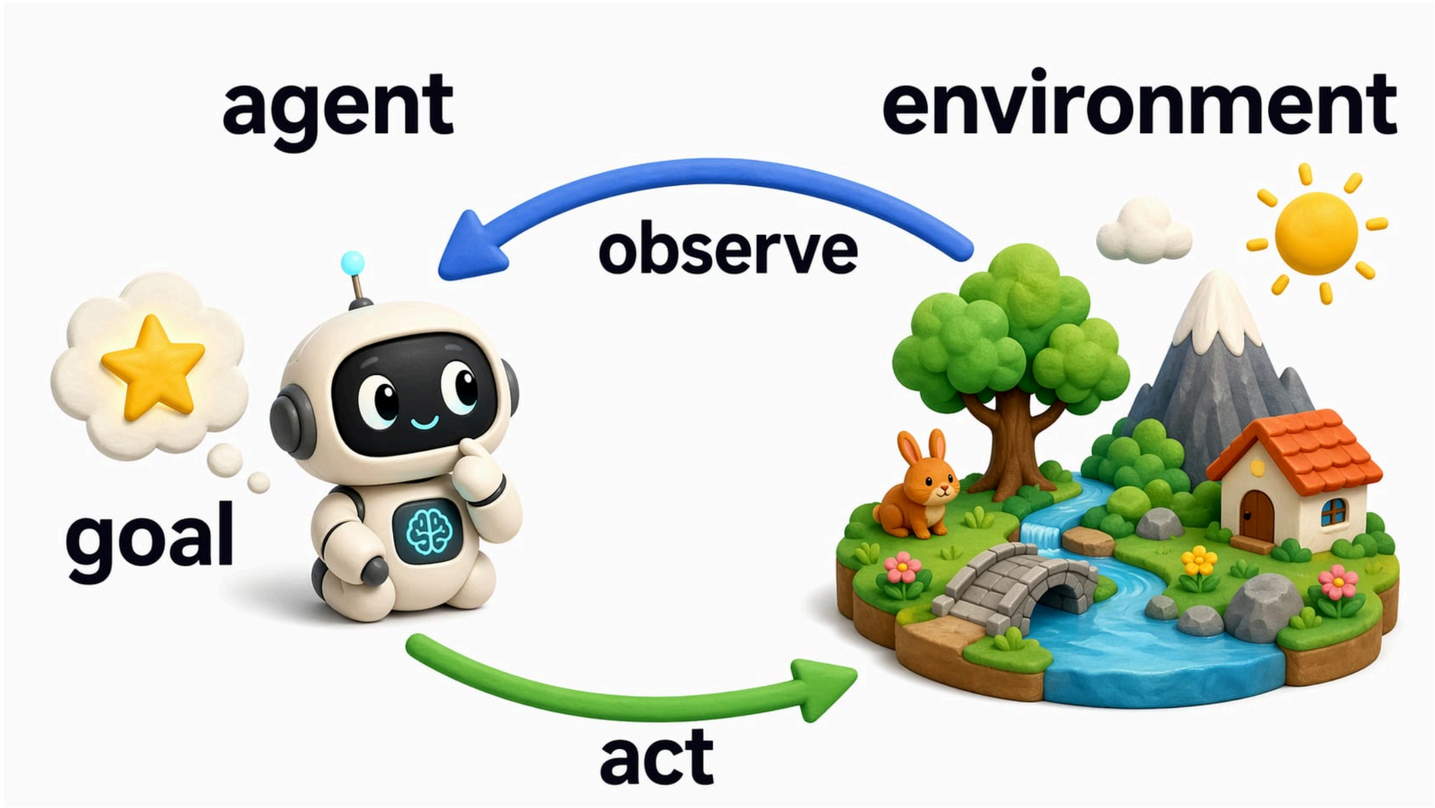
Artificial intelligence (AI) is a broad field, which includes many other techniques and applications that we'll not cover today, such as image/video/speech recognition and generation, recommendation systems, robotics, game-playing systems, and planning systems.

Agents and intelligence

What are we talking about?

Agent

An agent observes its environment and takes actions.



It may be a person, a robot, a chess engine, or a chatbot.

Intelligence

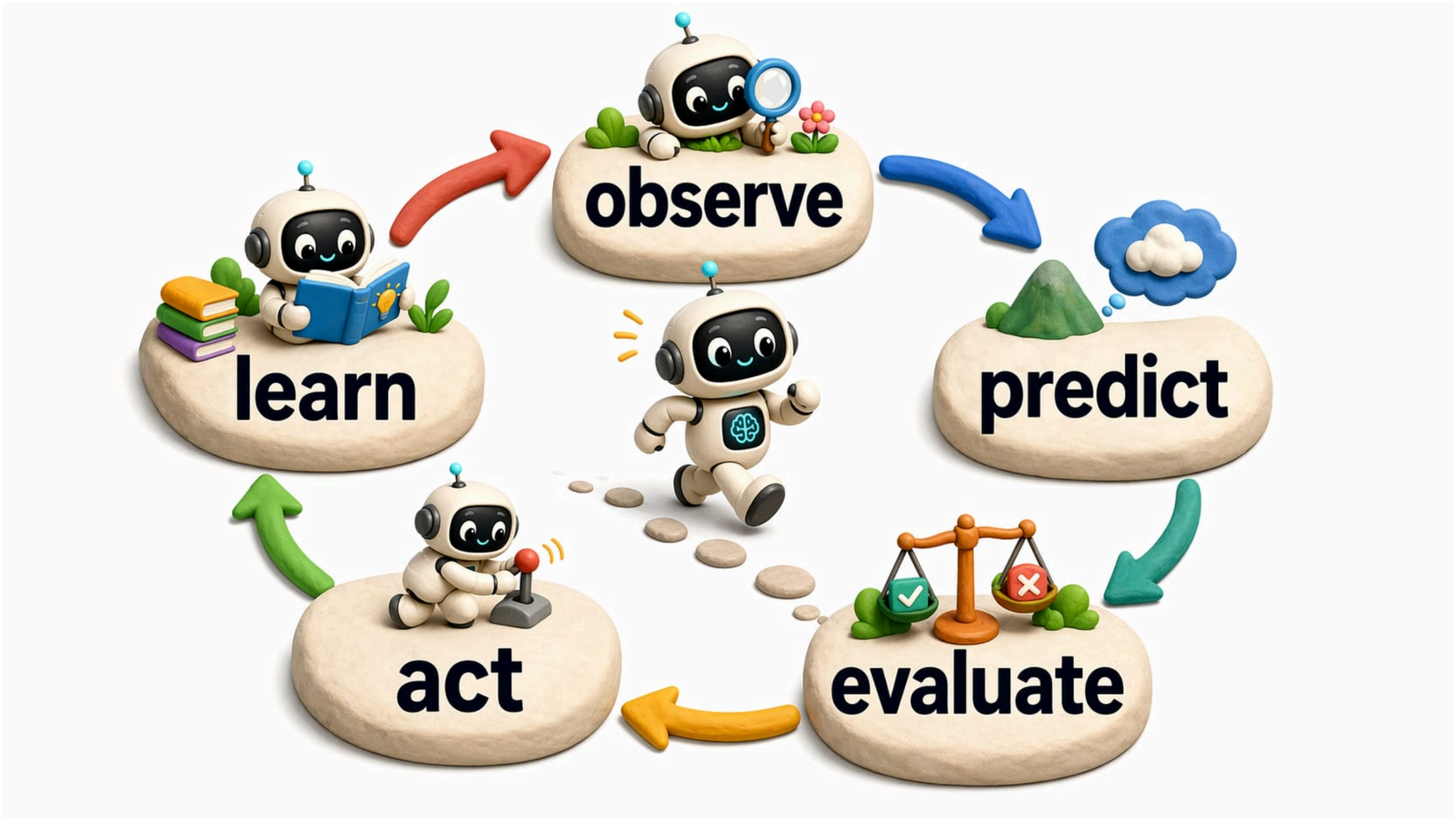
A useful working definition:

> Intelligence measures an agent's ability to achieve its goals in a wide range of unknown environments.

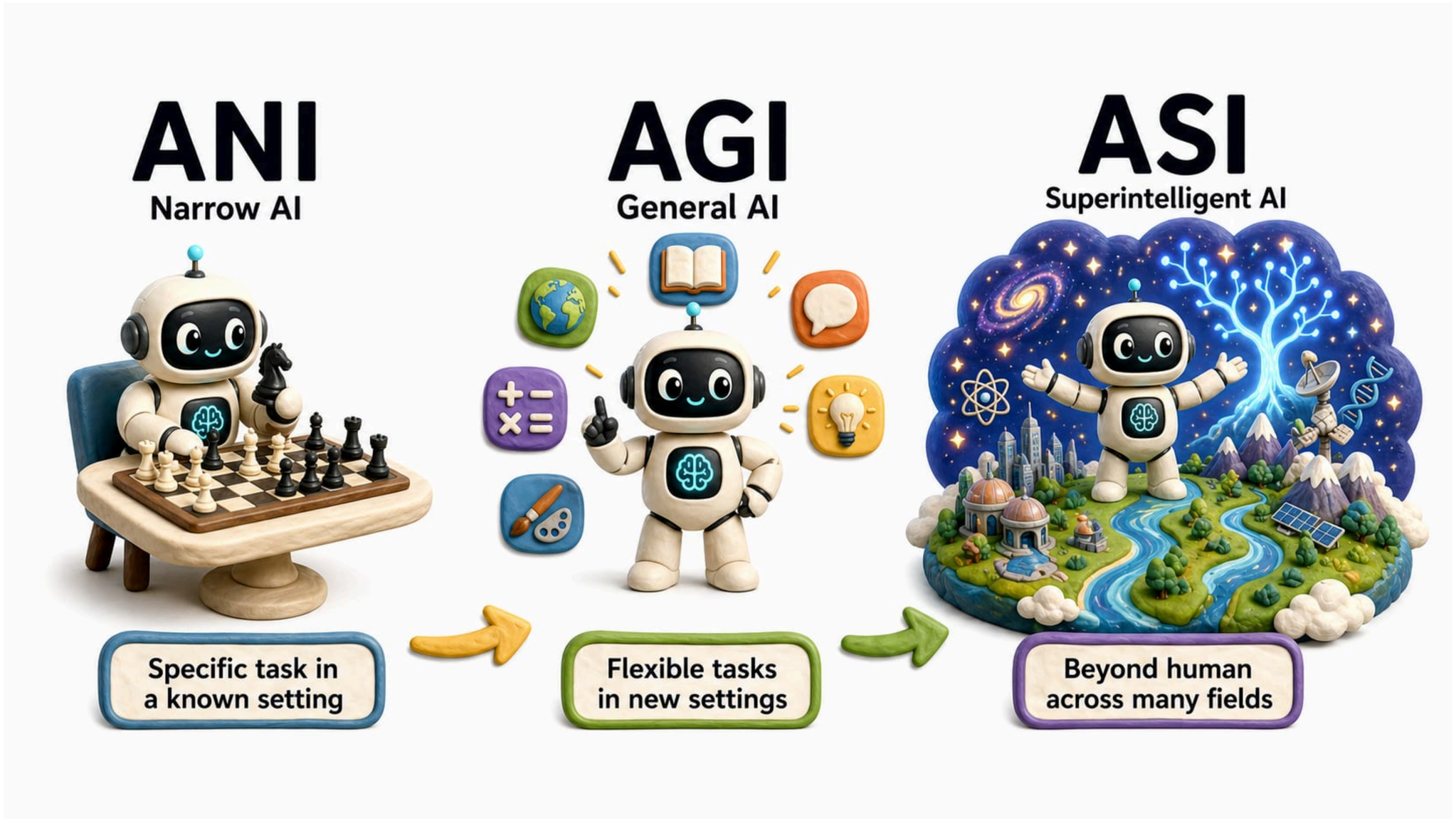
Source: A definition of machine intelligence

Loop

Most intelligent behavior can be drawn as a loop:

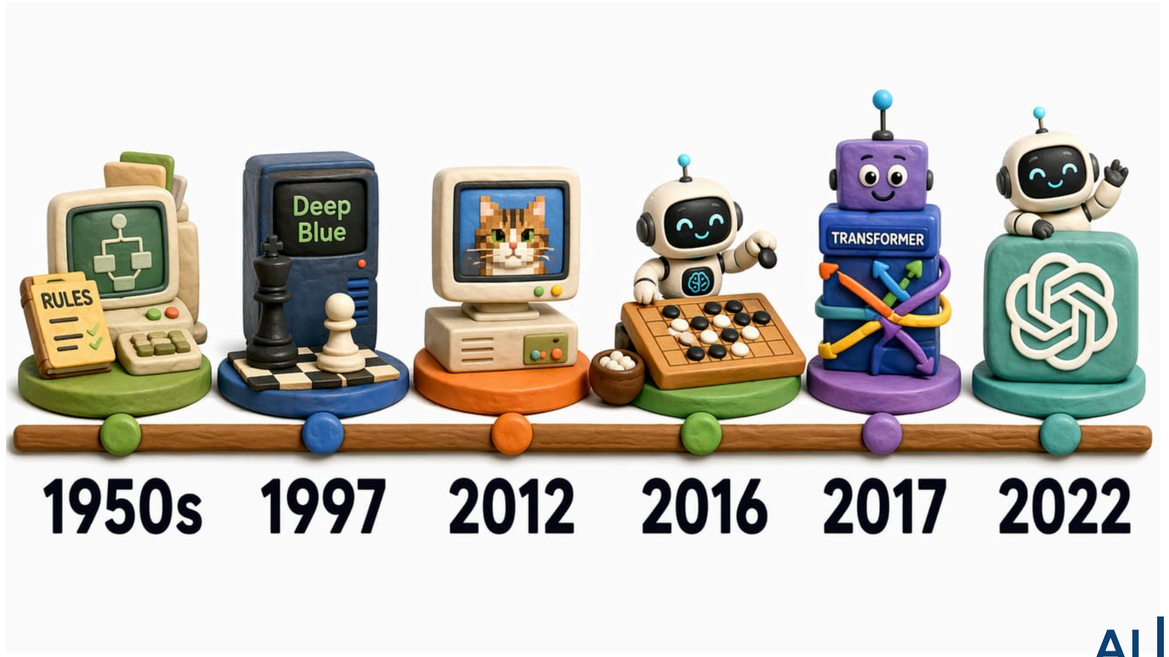


Artif. narrow, general & super-intelligence



Short history

- 1950s-1980s: Symbolic AI and expert systems.
- 1997: Deep Blue beats Garry Kasparov at chess.
- 2012: AlexNet changes computer vision.
- 2016: AlphaGo beats Lee Sedol at Go.
- 2017: Transformers reshape language modeling.
- 2022: ChatGPT makes LLMs mainstream.



Further reading: Deep Blue, AlexNet, AlphaGo

Why AI matters

Computers process information.

Classic software works when we can specify the steps.

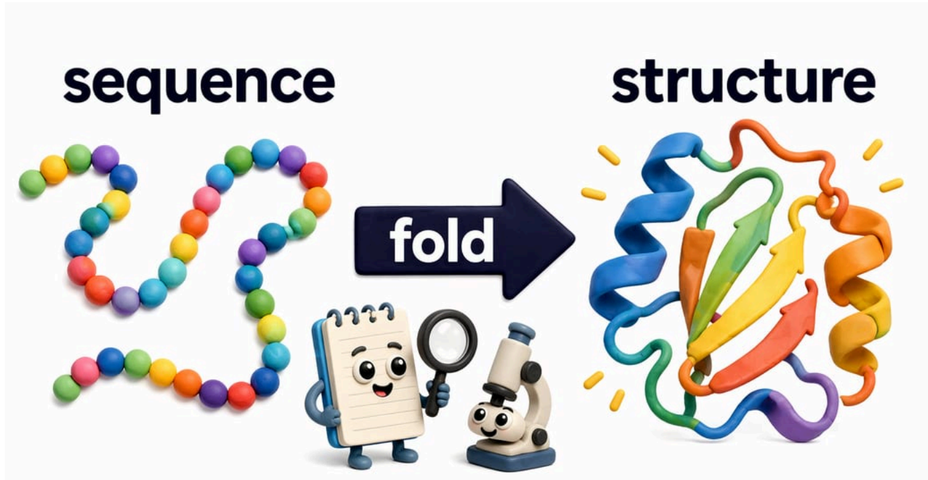
Learned systems help when the steps are hard to write, but examples, feedback, or verification are available.

This massively expands what computers can do.

Example: AlphaFold

Proteins fold into shapes that determine what they do.

Predicting those shapes used to be slow and difficult.



AlphaFold made high-accuracy predictions useful at scale.

Demis Hassabis and John Jumper shared half the Nobel Prize in Chemistry 2024 for protein structure prediction.

Source: Nobel Prize, AlphaFold

Surprising order of progress

Ten years ago, many people expected AI to automate physical labor before intellectual tasks followed by creative work if at all – but it's going exactly the other way around.

The order of miracles depends on two variables:

1. How much usable training data exists, and
2. How low the acceptable error rate must be.

Sources: Sam Altman on surprising progress; On the ordering of miracles

Specifiability vs. verifiability

Software 1.0 automates what we can specify.

Software 2.0 automates what we can verify.

This is why games, code, math, protein folding, and some creative tasks moved faster than expected.

Verification changes what practice can improve.

Further reading: Karpathy on verifiability

Task: Agent loop

Pick one familiar system.

- Recommendation feed
- Chess engine
- Chatbot
- Thermostat

Identify its environment, observations, actions, and goal.

Solution: Agent loop

Example: recommendation feed.

- **Environment**: a user, an app, and a stream of content.
- **Observations**: clicks, pauses, skips, likes, history.
- **Actions**: choose what to show next.
- **Goal**: maximize engagement and/or satisfaction.

(Clicks and watch time are only proxies for satisfaction.)

Learning as compression

Find patterns which generalize

Core question

How can a learner use past observations to make good guesses about future observations?

This question appears in statistics/science and daily life.

Induction

Guess a general pattern from limited examples:

- The sun rose every day so far, so we expect a sunrise tomorrow.
- A medicine helped in trials, so we expect it to help patients.

An AI model trained on a gigantic amount of text sequences gets good at predicting the next token.

Occam's razor

Occam's razor is not "simple things are always true".

It is a preference rule:

> If two explanations predict the observations equally well, prefer the one which makes fewer assumptions.

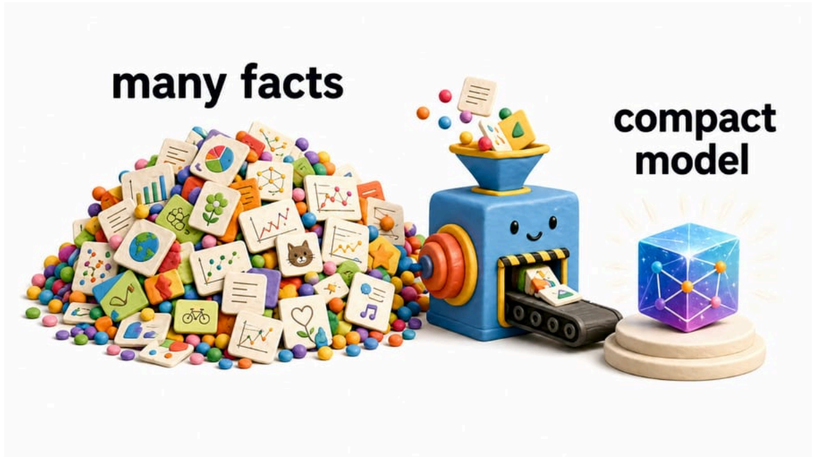
(The simpler explanation has less suspicious machinery.)

Wikipedia

Compression

Compression means representing information in a smaller form.

Lossless compression keeps all information, like `.zip`. Lossy compression discards details, like `.jpeg`.



Key insight: **A good theory compresses observations**.

Without a theory, every fact must be stored separately.

With a theory, many facts follow from a compact model.

Example: scientific theories

Evolution explains anatomy, sex ratio, mate preferences, aging, adaptation, fossils, and the branching tree of life.

Heliocentrism with Earth's axial tilt explains seasons, eclipses, retrograde motion, and planetary positions.

Both theories turn a pile of facts into a smaller story that keeps predicting new facts.

Creationists and flat-earthers need special rules to fit the same facts.

Fisher's principle, selection shadow, heliocentrism, retrograde motion

Kolmogorov complexity

Kolmogorov complexity asks for the shortest program that generates the given data.

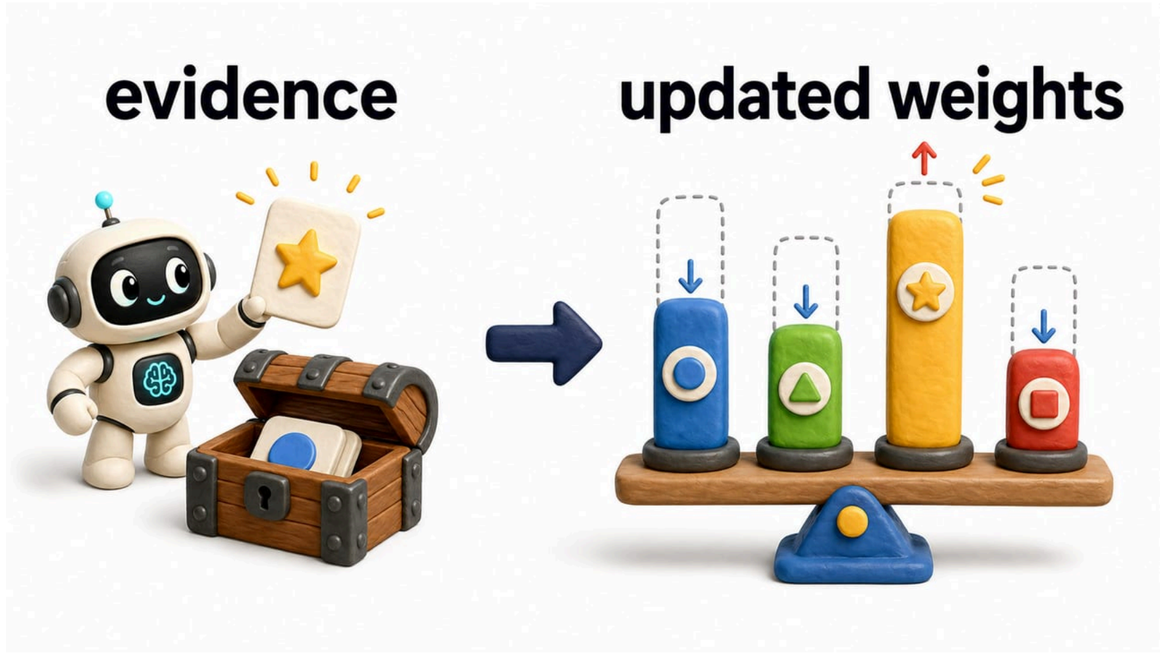
We won't formalize it today. (Cue: Turing machine.)

The intuition is enough: random-looking data needs a long description, patterned data has a short generator.

Wikipedia

Changing weights

Evidence does not usually leave us with one hypothesis. It shifts weight between many still-possible hypotheses.



Contradicted explanations lose weight.

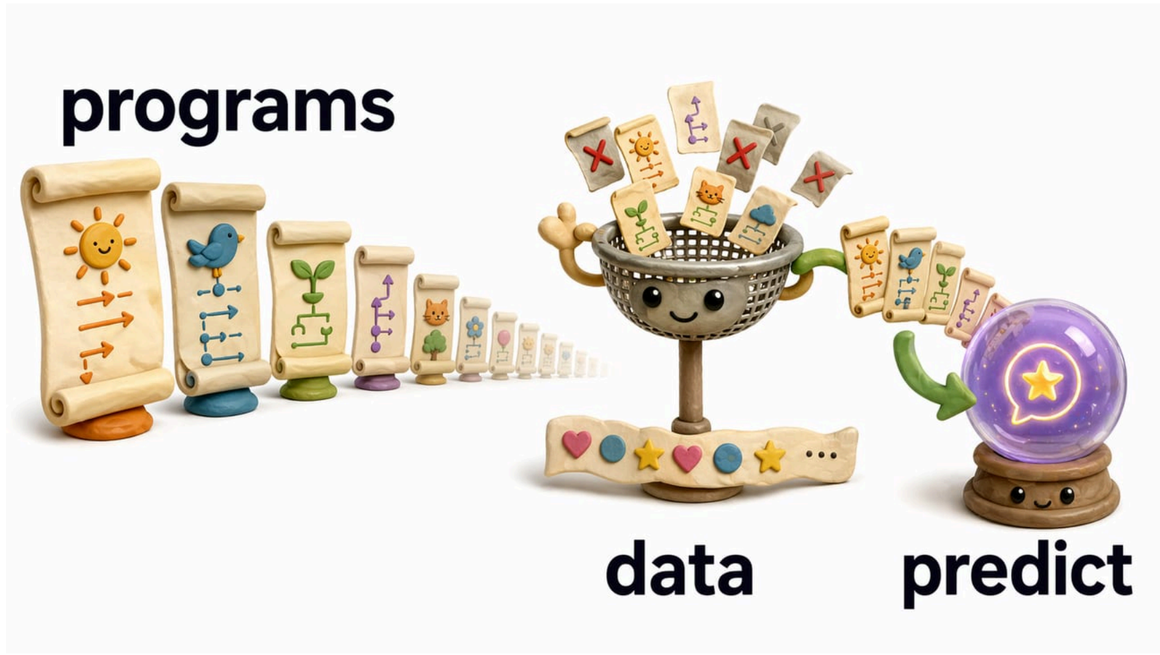
Explanations that keep predicting well gain weight.

This is the everyday intuition behind Bayesian updating.

Solomonoff induction

Imagine every program that could have generated the observations so far.

Shorter programs start with more weight.



Programs that fail to match the data are ruled out.

The remaining weighted mixture predicts the next value.

Connection to LLMs

Large language models are not Solomonoff induction.

LLMs approximate the data as well as they can within the model size that was chosen before the training.

But next-token prediction rhymes with the same idea: predict the next value from compressed experience.

The intelligence of LLMs emerges from the pressure to compress and predict well. The better you understand the world, the better you can finish human sentences.

Task: Kolmogorov complexity

Find a short description for each string as follows:

- Example: `101101010` = `(101)²0` repeated
- `01110101011` = ?
- `0110111010110111001` = ?

Given your descriptions, which character comes next?

Note that `01²` = `0(1)²` = `011`, which is just as short.

Solution: Kolmogorov complexity

Possible compact descriptions:

– `01110101011` = `01³(01)²` repeated → `…1`

– `0110111010110111001` = `((011)²1)²0` rep. → `…1`

Have you found shorter descriptions?

Task: computed sequence

The syntax from the previous task was very limiting, as it cannot perform computations.

What is a short description of the following string in human language?

```
01101110010111011110001001101010111100
```

Solution: computed sequence

Output one binary number after another, starting with 0:

`0`, `1`, `10`, `11`, `100`, `101`, `110`, `111`, `1000`, `1001`, `1010`, `1011`, `1100`.

The next four characters in this sequence are `1101`.

Task: Wason 2-4-6

I am thinking of a rule for triples of integers.

The triple `2, 4, 6` fits the rule.

Figure out the rule by testing other triples with me.

I will answer "yes" or "no" for each triple you want to test.

When you are confident, write down the rule.

Alternatively, play it online at task246.dylancastillo.co

Solution: Wason 2-4-6

The rule is not "add two", "even numbers", or "1×, 2×, 3×".

The rule is: any ascending sequence of numbers.

Many people test only examples that fit their guess:

`4, 6, 8`, `10, 14, 20`, or `3, 6, 9`.

Tests should eliminate hypotheses, not confirm just one.

Premature hypothesis elimination is part of the problem.

Always seek contradictions to your favorite theories.

Wikipedia, Less Wrong, The New York Times

Task: better explanation

Which explanation compresses the observations better?

1. Retrograde motion happens because each planet needs its own special exception in the sky.
2. Retrograde motion happens because Earth and the other planets orbit the Sun at different speeds.

Solution: better explanation

The second explanation compresses more.

It explains retrograde motion, planetary order, and future positions with one geometric model.

The first explanation can fit observations, but it spends a new exception for each pair of planets.

Artificial neural networks

Trainable function approximators

Artificial neural networks

Art. neural networks are statistical learning algorithms.

They are loosely inspired by biological neural networks.

They are used to approximate unknown functions that depend on many inputs and are hard to solve with handmade rules.

Further reading: Deep Learning

Simple network

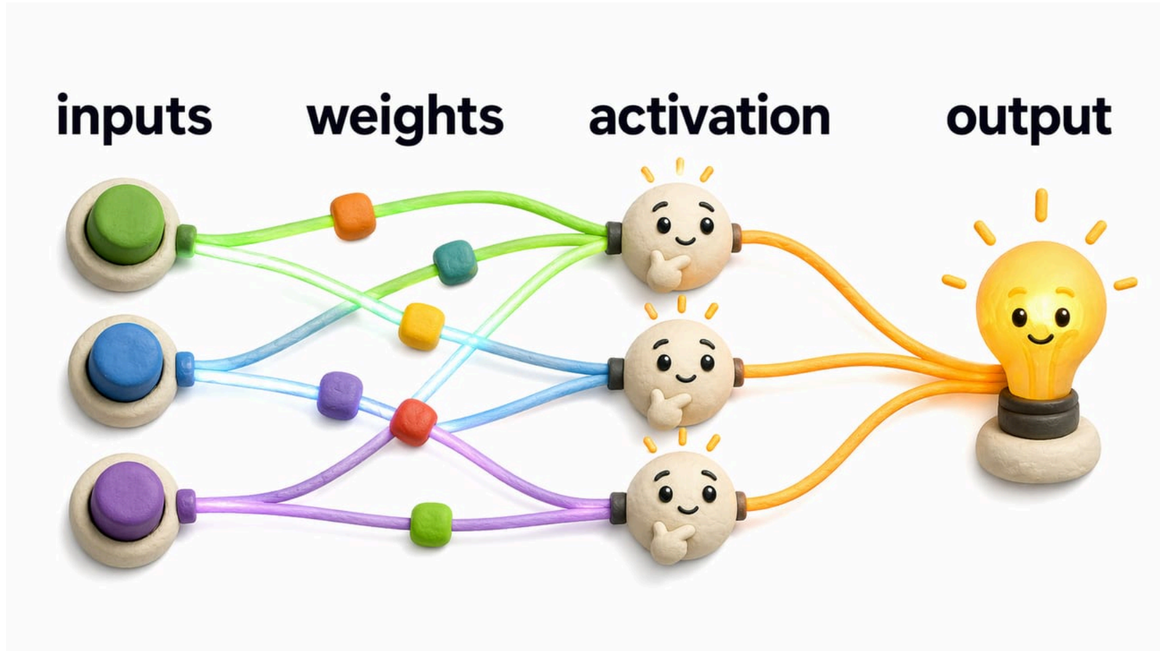
A tiny network has inputs, weights, an activation, and an output.

Inputs are numbers.

Weights say how strongly each input matters.
(Weights can also be negative.)

The activation bends the result.

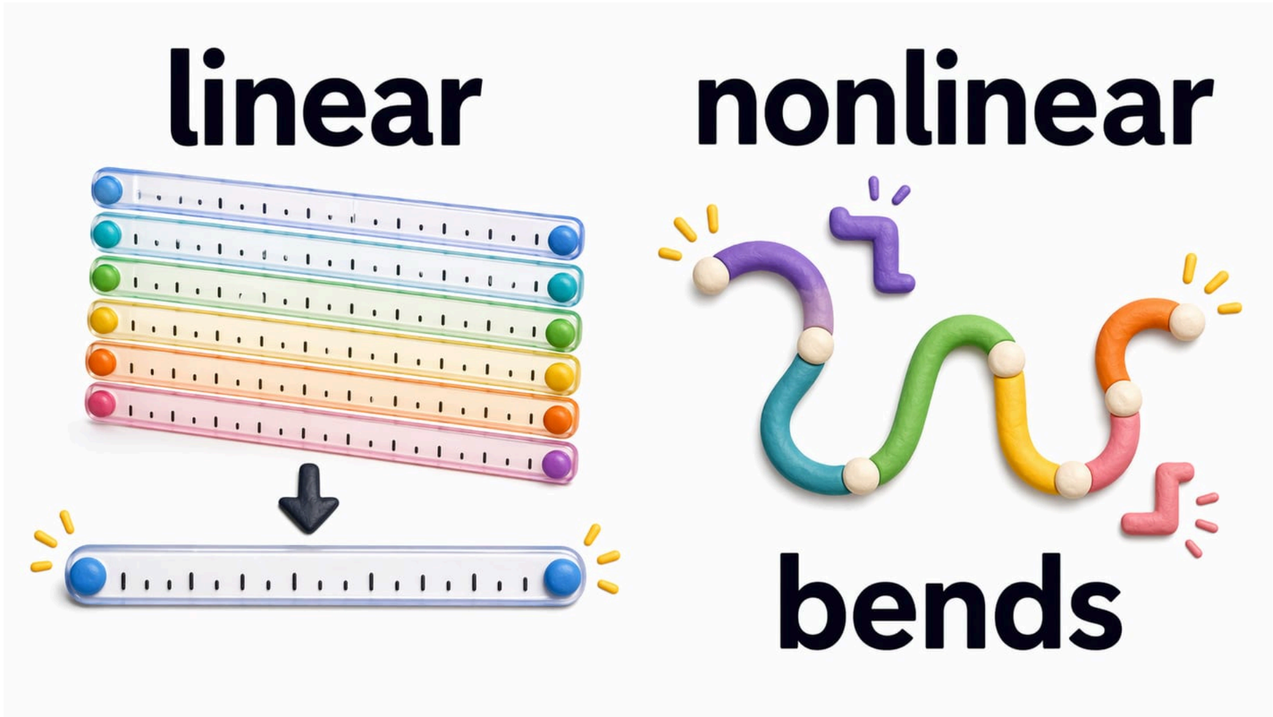
The output is the network's current guess.



Activation functions

Without nonlinear activations, layers collapse.

Stacking many linear layers is still just one linear layer.



Nonlinearity gives the network bends, corners, thresholds, and combinations.

Those bends let it approximate richer patterns.

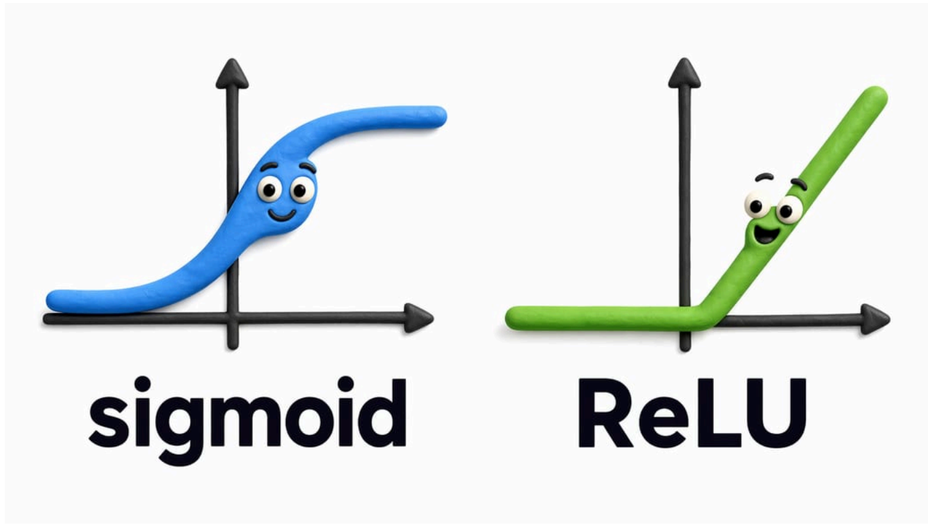
Popular activation functions

Sigmoid functions squash values into a smooth S-curve.

Rectified linear units (ReLUs) keep positive values and clip negative values to zero.

Modern networks often use relatives of ReLU.

The exact choice affects learning speed and behavior.



Wikipedia

Comparison to biological neurons

The brain analogy is useful only up to a point.

Artificial neurons are simple mathematical operations.

Biological neurons are living cells in a messy body.

Modern AI borrows the idea of many connected units, not a faithful model of the brain.

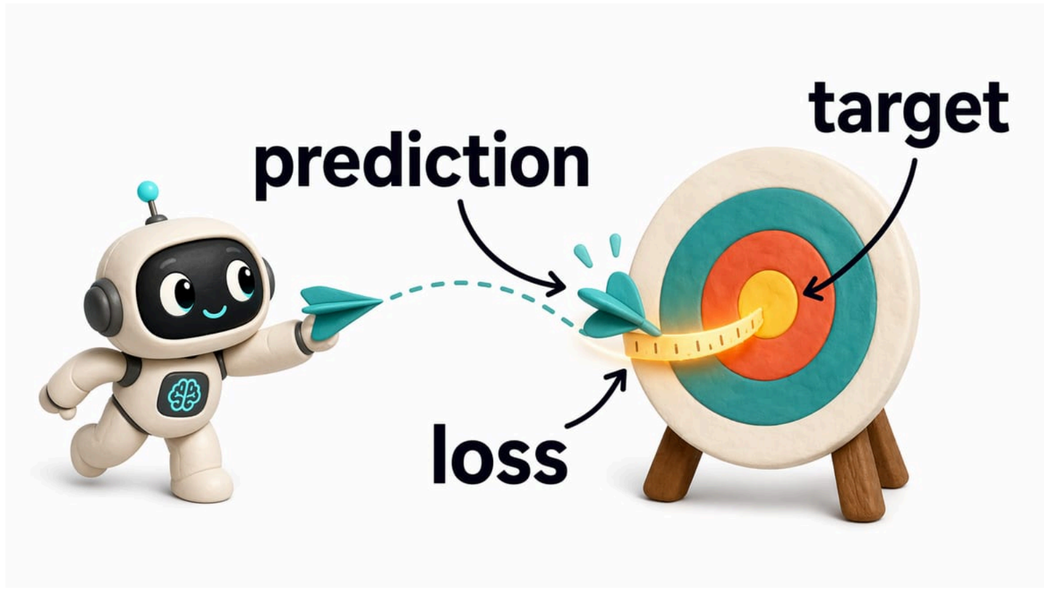
Loss function

A loss function gives the model a score to improve.

Low loss means the output was close to the target.

High loss means the output was bad in a measurable way.

Training needs this number because "better" must become something the system can optimize.



Optimization problem

Training is an optimization problem:

Find weights that make the loss smaller.

There are far too many weights to tune by hand.

So the training algorithm searches for changes that improve the model's guesses.

Gradient descent

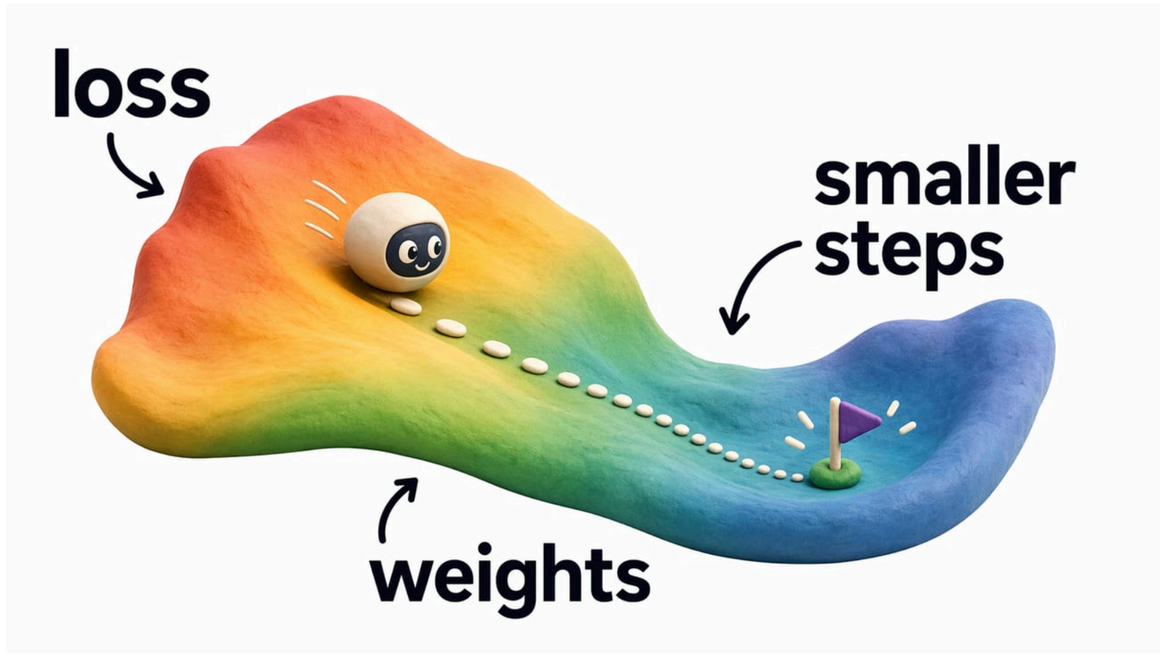
The basic idea of gradient descent is going downhill.

Measure which direction makes the loss decrease.

Move the weights a little bit in that direction.

Repeat many times.

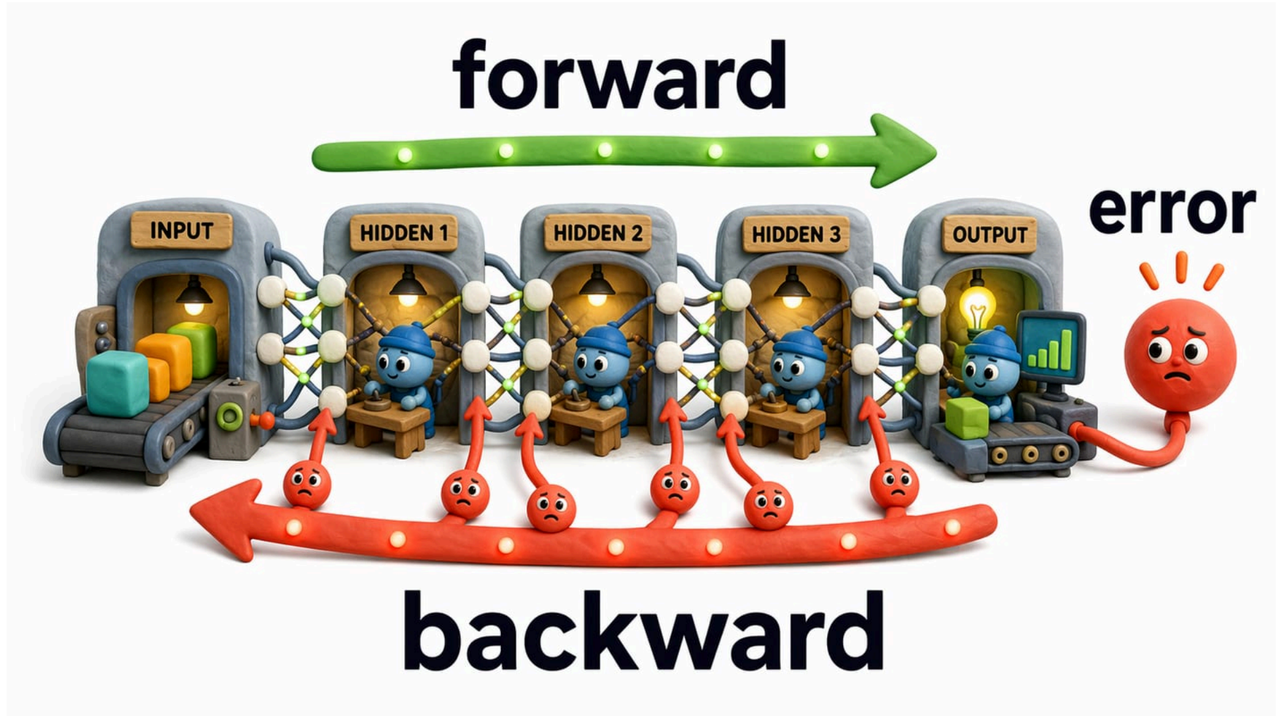
Tiny improvements can accumulate into useful behavior.



Backpropagation

Backpropagation calculates blame efficiently.

It asks how much each weight contributed to the error.



It reuses computations layer by layer, working backward from the output, making large networks trainable.

Backpropagation, partial derivative, chain rule

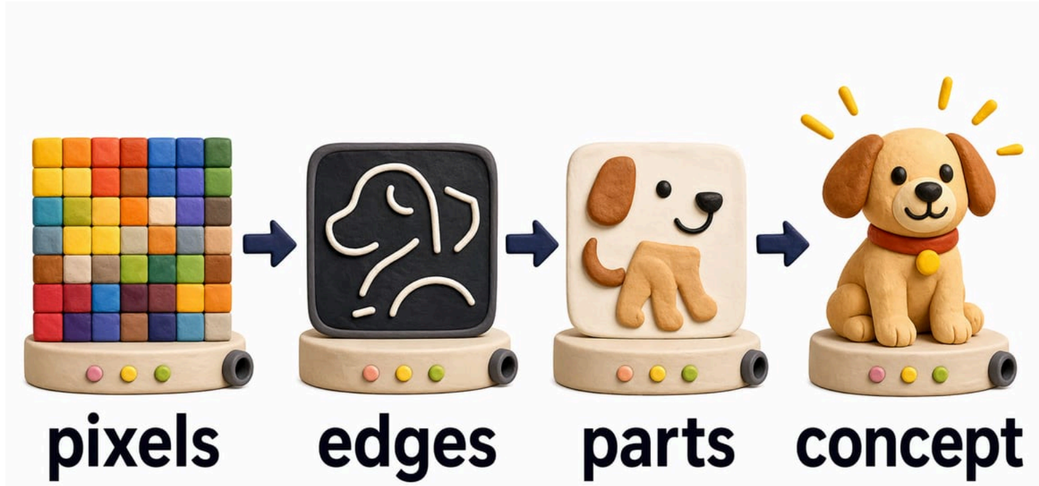
Representation learning

The important features do not have to be hand-coded.

A network can learn internal representations that make the task easier.

Early layers may detect simple patterns.

Later layers combine them into more useful concepts.

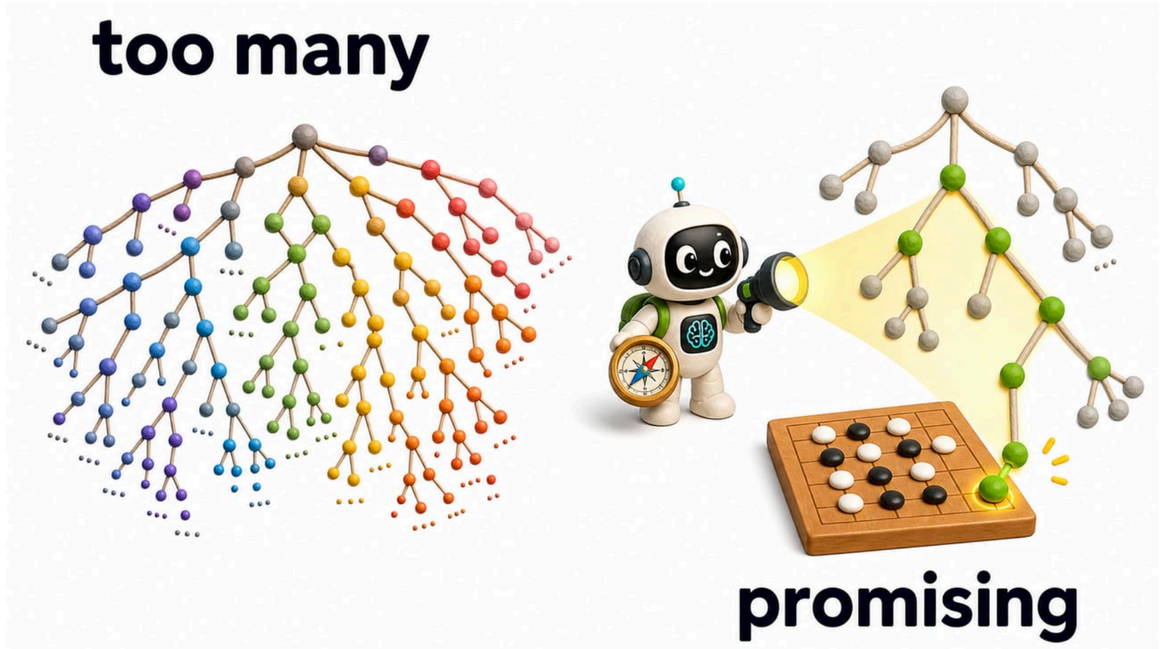


Wikipedia

Combating combinatorial explosion

Some search spaces grow absurdly fast.

Go has more possible games than there are atoms in the universe.

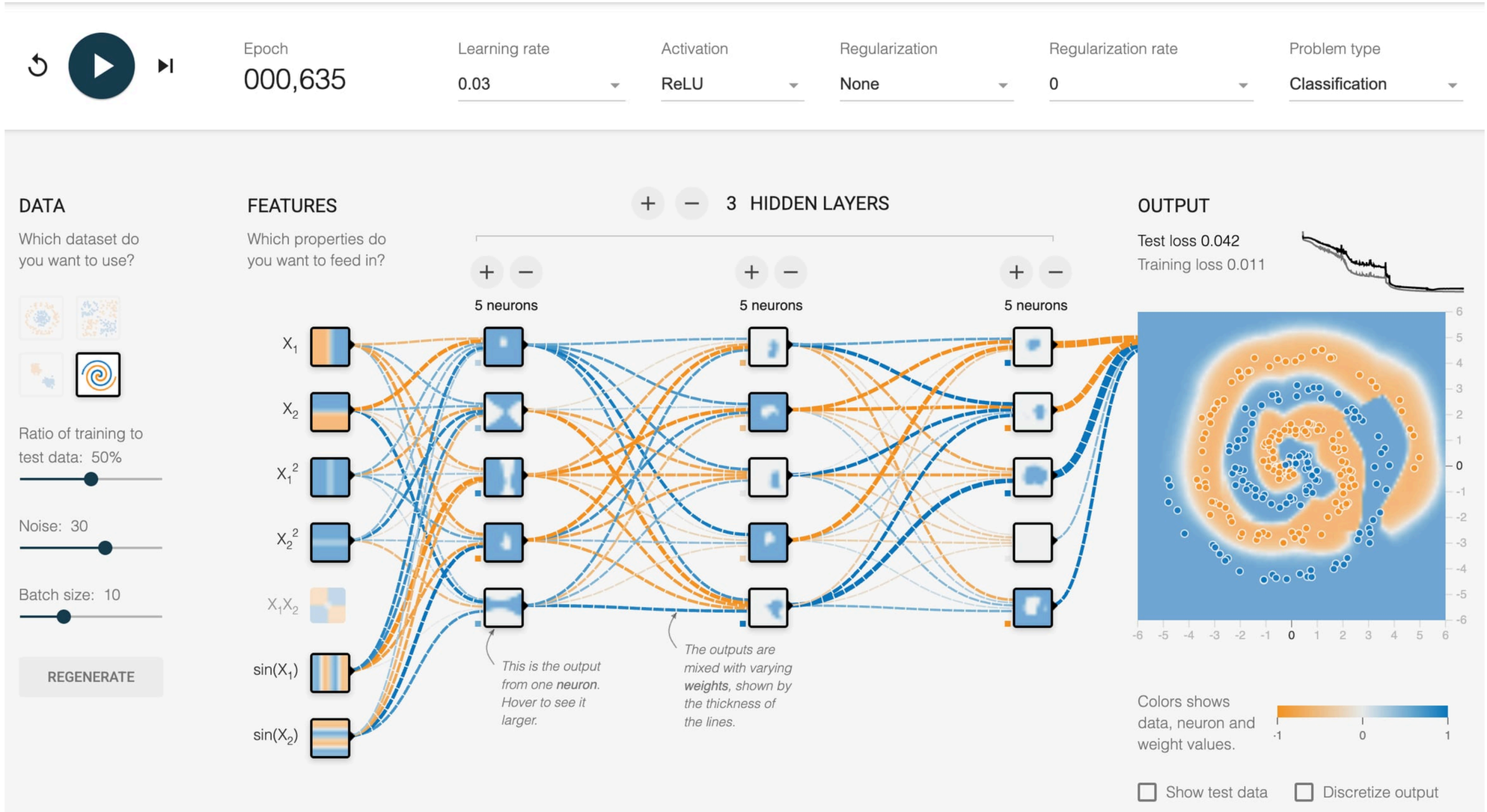


A model can learn to estimate which positions & which moves are promising.

That turns impossible brute force into guided search.

Example: AlphaGo

Watch a network get trained



Demo: TensorFlow Playground

Demo notes

More layers are not automatically better.

Too little capacity underfits simple patterns.

Too much capacity can chase noise.

Learning rate changes whether training is slow, stable, unstable, or completely broken.

Demo: TensorFlow Playground

Large language models

Neural networks trained to predict the next word

Guiding question

What are you talking to when you use ChatGPT?

Not a person. Not a database. Not pure autocomplete.

You are using a trained model inside a product harness.

GPT stands for generative pre-trained transformer

The stages of building an LLM

The training data needs to be prepared and tokenized.

Post-training turns the base model into an assistant.

A product harness adds context, tools, and an interface.

Training data

Before training, gigantic amounts of text are collected.

Raw web pages are filtered, cleaned, deduplicated, and mixed with other curated sources into a training dataset.

Example: FineWeb

Tokenization

LLMs could be built to predict just the next character.

That would use a tiny alphabet, but every word would take many prediction steps.

They could also predict whole words, but then the vocabulary would be huge and brittle.

Tokens are a compromise: common chunks become single units, rare text can still be spelled out from pieces.

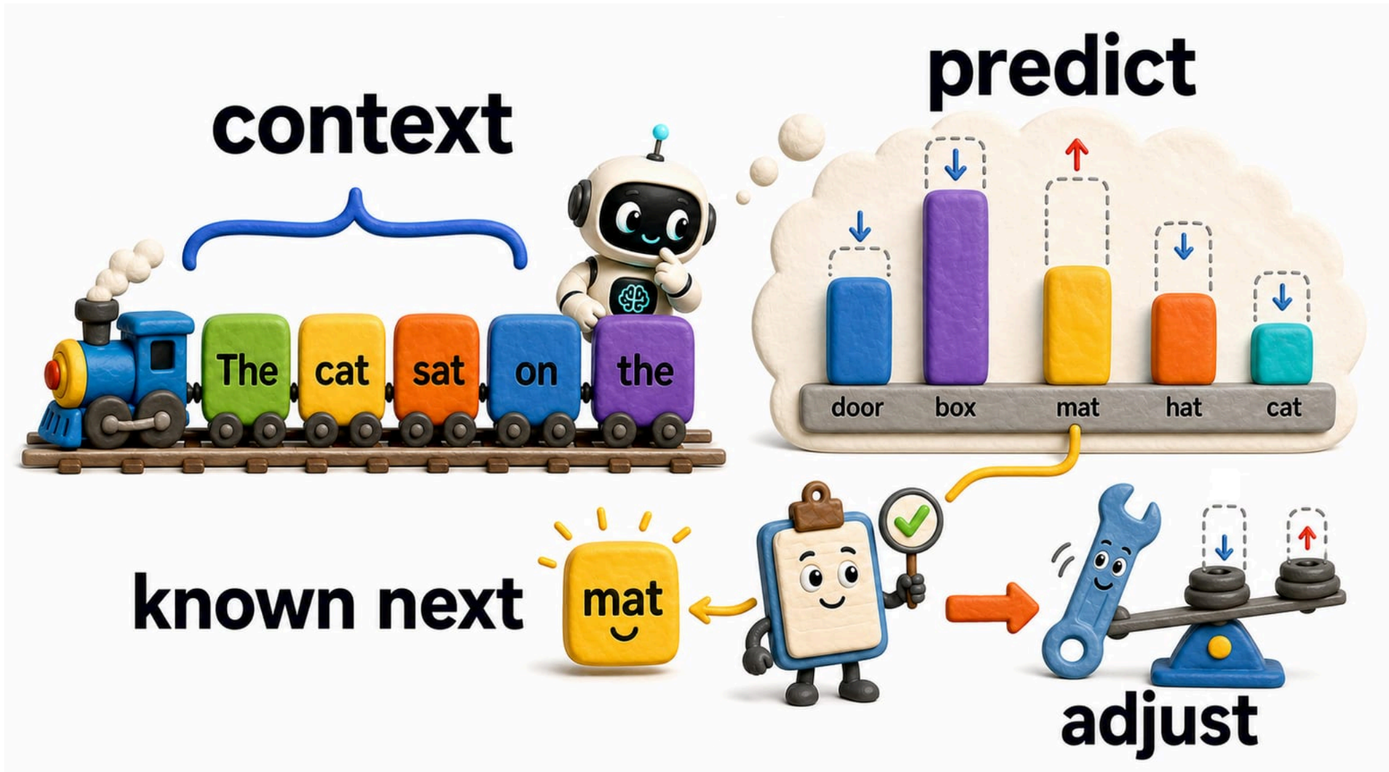
Example: OpenAI tokenizer

Training objective

During training, the model predicts the next token.

The actual next token is known from the dataset.

Training increases the probability that the known next token is predicted.



Transformer

Most modern LLMs use transformer architecture.

The key is attention: Each token can use information from earlier tokens when computing what comes next.

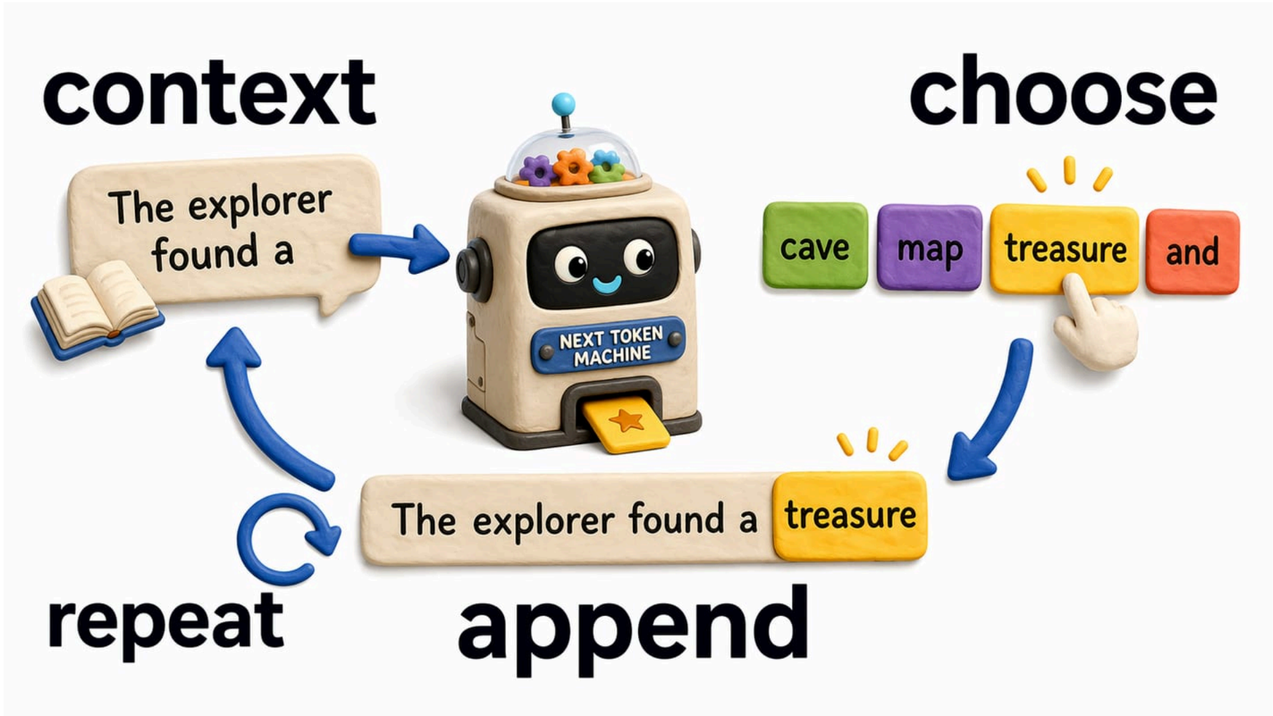
This makes context matter at every layer.

Demo: LLM visualization

Inference

After training, the weights are fixed.

The model receives context and predicts a distribution over possible next tokens.



One token is chosen, added to the context, and the process repeats. Text generation/inference is a loop.

Base model

The result of pre-training is a base model.

You can think of it as an internet-text simulator.

It can continue sentences and complete patterns.

It is not automatically a helpful assistant.

Supervised fine-tuning

Supervised fine-tuning trains the base model on many examples of useful conversations.

The base model learns the assistant pattern:

Follow instructions, answer clearly, refuse some requests, and use the expected conversation format.

InstructGPT paper

Reinforcement learning

In verifiable domains, models can practice.

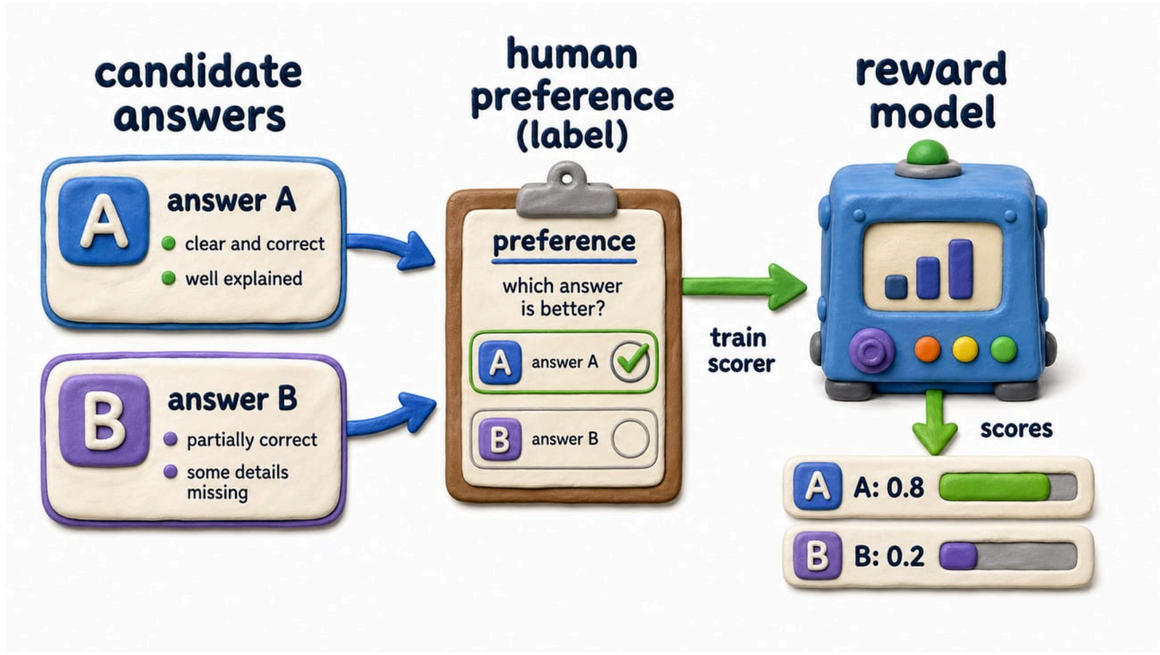
Try many approaches to a problem.

Reinforce the ones which reach a correct answer.

This is powerful when feedback is cheap and reliable.

Human feedback

Reinforcement learning from human feedback (RLHF) uses human preferences to train a reward model.



That reward model is only a proxy for human judgment.

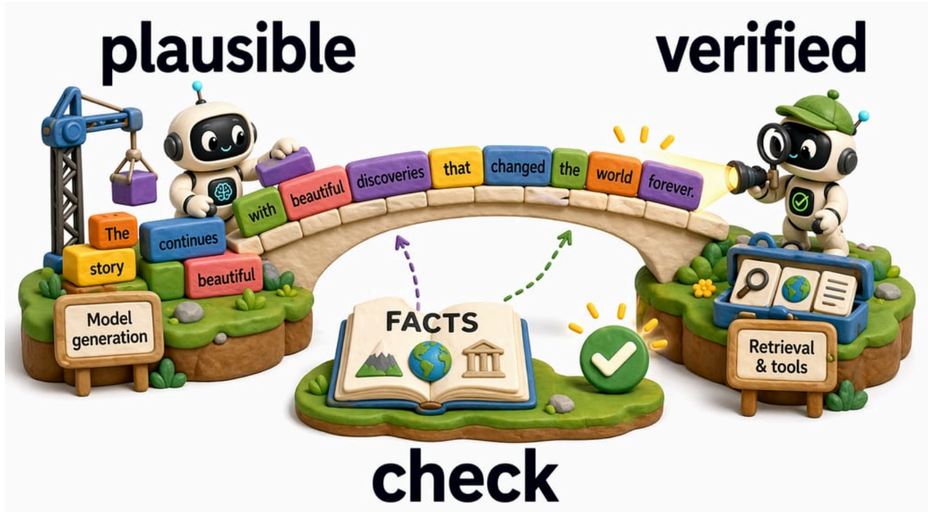
Proxies are useful, but they can also be gamed.

InstructGPT paper

Hallucinations

Hallucination is a natural failure mode of completion.

Models are trained to produce plausible next tokens, not truth.



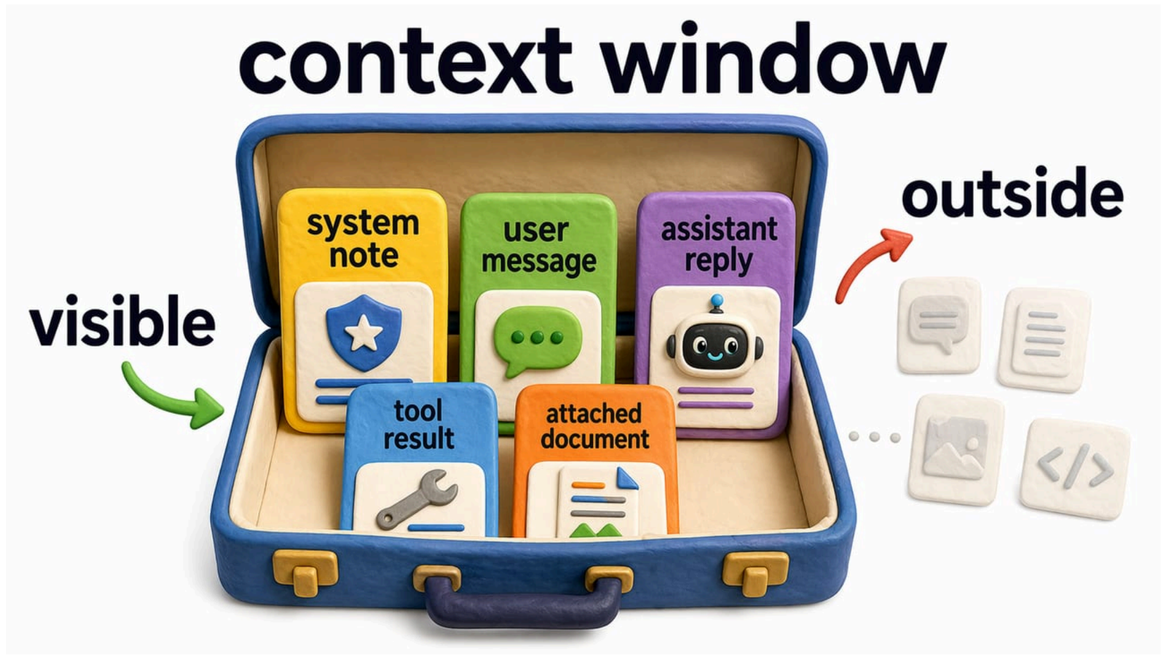
Possible mitigations include:

- Refuse or say "I don't know" when evidence is weak.
- Use tools to retrieve evidence and cite the sources.
- Check whether the solution actually works.

Context window and context length

The context window is the model's working memory.

It contains the system message, your messages, assistant replies, tool results, and attached documents.



The size of a model's working memory is limited. What falls outside the context window is no longer visible.

Wikipedia

Weights and context

Knowledge in weights is like vague recollection.

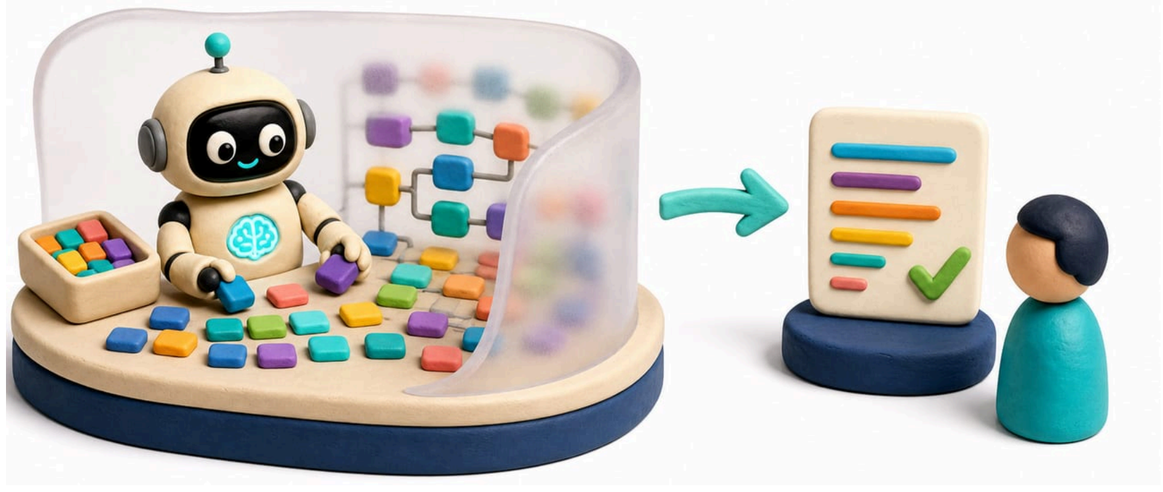
Knowledge in context is like evidence on the desk.

Models are usually more reliable when the needed facts are present in the prompt or retrieved by a tool.

Reasoning models

Humans often think by talking to themselves.

LLMs can do something similar with tokens.



A reasoning model can generate private intermediate tokens before writing the answer the user sees.

Those tokens give it room to plan, check, backtrack, and choose a better response.

Useful analogy

Non-reasoning models are like the associative thinking of our System 1: fast and intuitive.

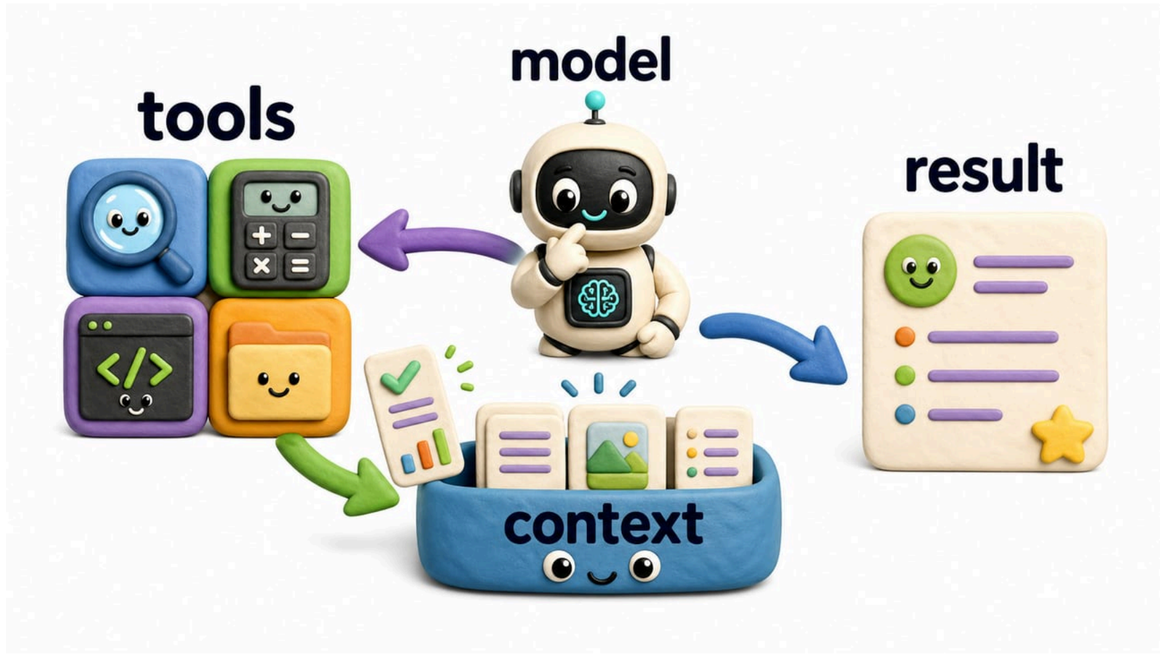
Reasoning models are closer to the analytical thinking of our System 2: slow and deliberate.

Reasoning models are typically more reliable because they work through the problem before answering.

Tool use

A surrounding harness lets the model request a search, calculator, code runner, file read, etc.

The tool result is inserted back into context.



Then the model continues with better evidence.

Tools let models enrich their context themselves.

Jagged intelligence

LLMs can look brilliant and brittle in the same minute.

They may write a good essay and miss a simple count.

They may debug code and invent a citation.

Their strengths and weaknesses are not shaped like ours.

Task: Tokenizer

Open a tokenizer and compare:

- `hello world`
- `Hello, world!`
- `internationalization`
- `ChatGPT`
- one emoji

What changes when you add spaces or punctuation?

Tool: OpenAI tokenizer

Solution: Tokenizer

You should see that tokens are not simply words.

Common chunks may become one token.

Spaces often attach to the following word.

Punctuation and unusual text can split differently.

This explains some weird spelling and counting failures.

Task: Answering from memory vs. search

Ask an LLM: "Who's the current director of the Swiss Study Foundation? Don't look it up, just answer from what you know."

Then ask again, letting it use its search functionality.

For ChatGPT-5.5, the former fails, the latter succeeds.

Source: Swiss Study Foundation

Task: Think first

Ask an instant model and then a reasoning model:

"Anna, Bert, and Cara sit in seats 1-3 from left to right.

They drink tea, coffee, and water, one each.

– Bert sits immediately right of the tea drinker.
– Anna sits left of the water drinker.
– Cara does not drink coffee.
– Anna does not drink tea.

Who drinks water? Answer with just the name."

Solution: Think first

GPT-5.5 Instant answers (repeatedly) with "Cara".

GPT-5.5 Thinking gives the correct answer: "Bert".

(Bert cannot be in seat 1. If he were in seat 2, the tea drinker would be in seat 1. Anna cannot drink tea, so Cara would have to be in seat 1, leaving Anna in seat 3. But Anna must sit left of the water drinker. Thus, Bert is in seat 3, and the tea drinker is in seat 2. Anna cannot drink tea, so Cara must be the tea drinker in seat 2. Thus, Anna is in seat 1 with coffee, Bert in seat 3 with water.)

Tips for using LLMs

Treat LLMs as powerful collaborators, not authority

Main stance

Use LLMs like collaborators with unusual strengths.

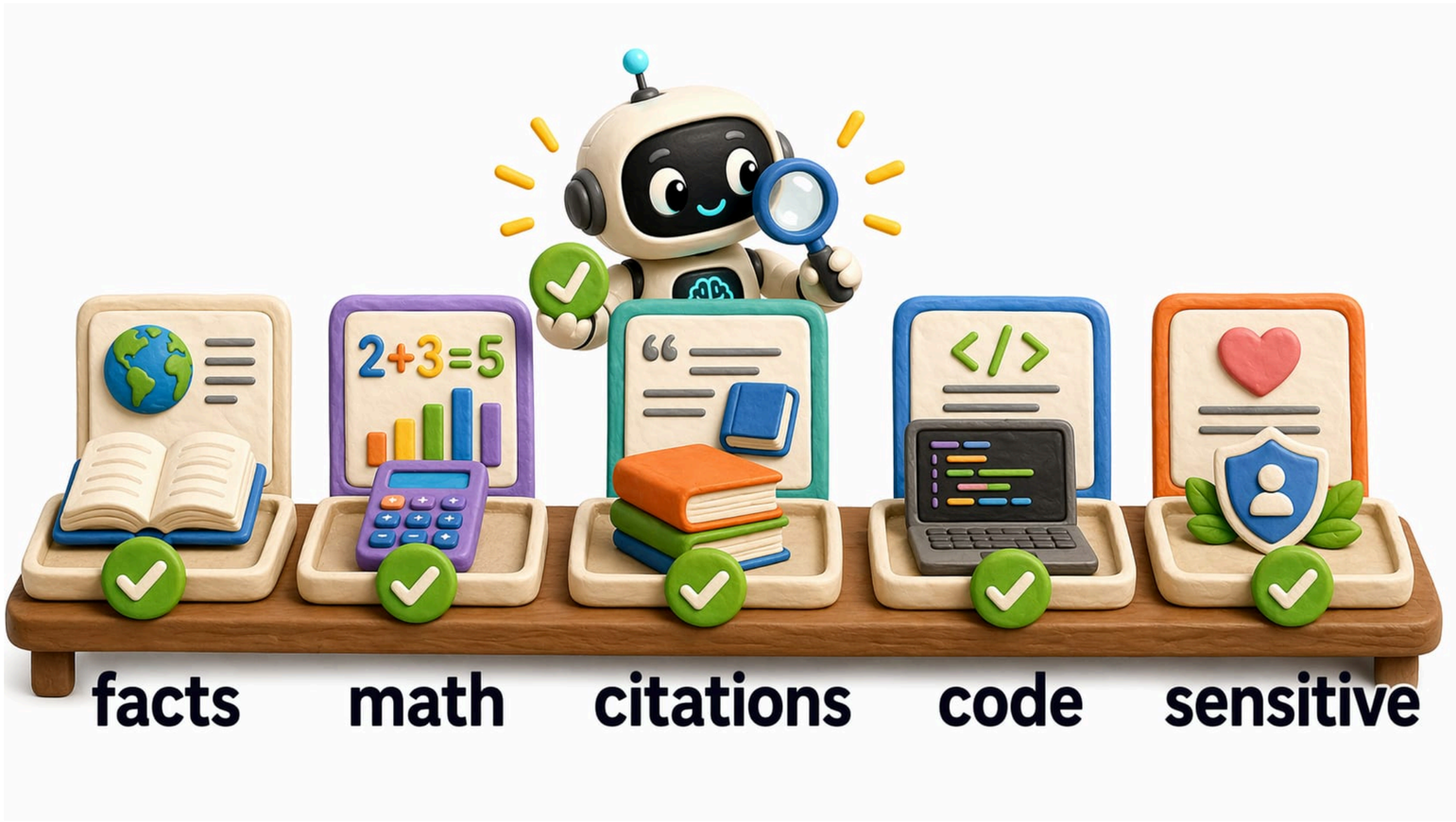
They are fast, patient, broad, and good at drafting.

They are also capable of confident nonsense.

Your job is to steer, check, and decide.

Check important outputs

Always verify high-stakes or brittle outputs:



Don't trust, verify!

Prompt recipe

Give the model the shape of the job:

(The last one is desired output format.)

Better input leads to better output.

Fresh context

Old context can pollute new work.

Start a new chat when the topic changes, the model keeps following stale assumptions, or the conversation has become too long to reason about.

Context is memory, but also baggage.

Project instructions

Persistent instructions help with repeated work.

Files like AGENTS.md can preserve preferences, constraints, naming conventions, and workflow habits.

This is especially useful for coding, writing, and research.

Meta-prompting

Use AI to improve your use of AI:

- Ask it to improve your prompt.
- Ask it to ask clarifying questions.
- Ask it to critique your instructions.
- Ask it what information is missing.
- Ask it to write the AGENTS.md file.

The four AI commandments

1. Do not send sensitive data to LLMs.
2. Do not submit AI output as your own work.
3. Use AI to learn faster, not to skip learning.
4. Use AI as your sparring partner, not your oracle.

Try new things!

You learn to master new tools by using them.

Whatever you do in life, let LLMs assist you.

It's your personal board of experts available 24/7.

Experiments beat untested opinions. What's barely working now will likely work with the next model update.

This deck itself was an experiment to automate the tedious creation of slides. I hope you liked the result!

Takeaways

- Agents pursue goals in environments.
- Learning can be viewed as compression.
- Neural networks learn underlying patterns.
- LLMs predict tokens inside a product harness.
- Tools and context make models more reliable.
- Good users steer, test, and verify.