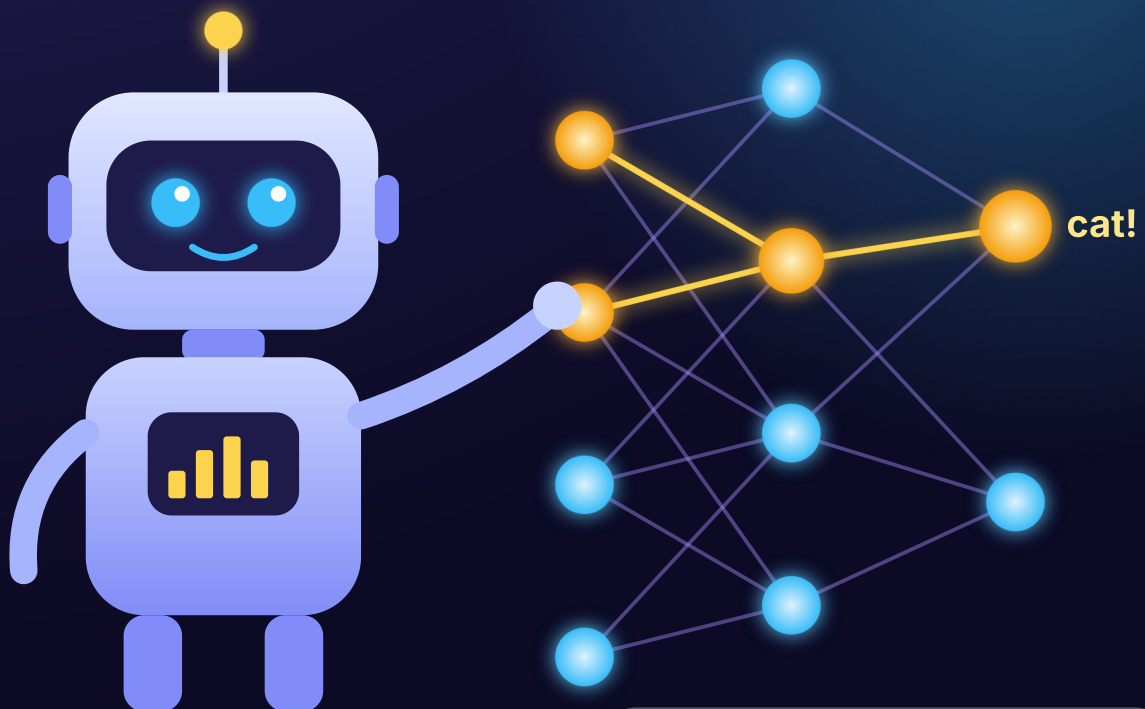


FREE BOOK

How Does Artificial Intelligence Work?

a guide for curious 12-year-olds (and up)

Data, neural networks, chat assistants, generated images and how to use AI with care, explained without formulas and with activities you can do without a computer



How did I learn that?

VALANDROUI

How Does Artificial Intelligence Work?

a guide for curious 12-year-olds (and up)

Download the latest version at jeffev.github.io/jeffersonvalandro

Jefferson Valandro

1st edition · October 2026 · free distribution

© 2026 Jefferson Valandro. This book is free: you may download, print and share it as much as you like, as long as it is unchanged, credits the author and is not sold.

ChatGPT, Claude, Gemini, Scratch, Python and the other brands, models and libraries mentioned belong to their respective owners. This guide is not affiliated with any of them.

Contents

Before You Begin

- 1 What Is Artificial Intelligence?**
- 2 A Quick History of AI**
- 3 Rules or Examples?**
- 4 Data: Food for AI**
- 5 How a Machine Learns: Making Mistakes and Fixing Them**
- 6 Artificial Neurons**
- 7 Neural Networks: Many Neurons Together**
- 8 How a Computer Sees**
- 9 Words Become Numbers**
- 10 The Guess-the-Next-Word Game**
- 11 How an AI Assistant Is Born**
- 12 Why AI Sometimes Makes Things Up**
- 13 AI That Looks Things Up and Uses Tools**
- 14 AI That Draws, Speaks and Composes**
- 15 Learning by Playing: Rewards**
- 16 Using AI Wisely**
- 17 I Want to Build My Own AI!**

Glossary

BEFORE YOU BEGIN

Before You Begin

It was Sunday, and Grandma's house smelled like cornmeal cake. Theo was stirring the batter while Lia drew at the kitchen table.

"Yesterday I asked an AI assistant to help me with a recipe," Theo said. "It answered right away. How does it know these things?"

"Maybe there's someone in there typing?" Lia joked, without looking up from her drawing.

"I don't think so. But then what is it?"

This book exists to answer that question. And many others that come right behind it.

Who this book is for

It was written for someone around 12 years old who is curious. Maybe you have already used an AI assistant, or watched someone use one. Maybe you have only heard about it on the news or at school.

You don't need to know how to program. You don't need to like hard math. If you can add, multiply small numbers, and understand what "half" or "10%" means, you already have everything you need.

Adults are welcome too. Plenty of grown-ups want to understand artificial intelligence without facing formulas.

The goal is simple: after each chapter, you should be able to **explain the idea to a friend**. That is the best test of whether someone really understood.

Who Lia and Theo are

Lia and Theo are cousins, and they are both 12. Lia loves drawing and playing soccer. Theo loves video games and spends his Sundays cooking with Grandma.

They are both made-up characters. They show up at the start of some chapters asking questions, the way any curious person would.

How to read

You can read in order, from beginning to end. That is the easiest path, because each chapter uses a little of the one before.

But you can also skip around. If you just want to know why AI sometimes makes things up, you can go straight to chapter 12. When a chapter uses an idea that came earlier, it will tell you, with something like "remember chapter 5?". Then you go back and take a look.

The book has three parts, roughly like this:

- First, what AI is and where it came from.
- Then, how a machine learns from examples, and how it sees images and handles words.
- Finally, the surprising things AI does, where it goes wrong, and how to use it well.

The boxes

In the middle of the text, you will find three kinds of box.

Try it is an activity to do without a computer. You will use paper, pencils, cards, dice, or the help of friends. Every chapter has one. Doing the activity helps much more than just reading.

Did you know brings a true, interesting fact that didn't fit in the main text.

Caution talks about responsible and safe use. Take these seriously.

TRY IT

Before reading the rest of the book, take a quick test on yourself.

1. Grab a sheet of paper and write: "How I think AI works".
2. Write three sentences with what you imagine today. There is no wrong answer.
3. Keep the sheet inside the book, or in a drawer.
4. When you finish the last chapter, read it again.

Expected result: you will notice how much your idea has changed. Some things you imagined may be right. Others will seem funny.

Simplify, yes; make things up, no

Artificial intelligence is a big subject, full of details. This book simplifies a lot. But simplifying is not making things up.

Everything written here is true in a simpler version. When a simplification hides something important, the text will say so, with phrases like "it's actually a little more complicated". When a question still has no answer, not even among scientists, the book says that too.

You won't find made-up numbers, like "the AI read a trillion books". If the book isn't sure about a number, it prefers not to write it.

An important date

Artificial intelligence changes fast. This book was written in **October 2026**. The main ideas should hold up for a long time. But product names, new features, and news may change after that.

When the book mentions an AI assistant, like ChatGPT, Gemini, or Claude, it is only as a familiar example. It is not an ad or a ranking.

And after this book?

This is a short book. It is the simple version of a much bigger book, called *Artificial Intelligence from the Inside*, by the same author. The big book has math, code, and lots of details.

It was made for adults. It might be a good read for you in a few years, if the curiosity keeps going. For now, this one is the right place to start.

Theo took the cake out of the oven. Lia closed her sketchbook.

"So let's find out what's in there," she said.

Let's go.

CHAPTER 1

What Is Artificial Intelligence?

Lia had just finished watching a soccer video on her phone. Right below it, three more soccer videos popped up, all of goals she would love to see.

"How does the phone know I like this?" she asked.

Theo shrugged.

"Must be artificial intelligence. But what is that, exactly?"

Good question. Lots of people talk about AI every day, and few can explain it. Let's try.

A definition to get started

Artificial intelligence, or **AI**, is the name given to computer programs that do tasks that used to seem to require a person's intelligence. Recognizing a face in a photo, translating a sentence, having a conversation, suggesting the next video. All of that seemed like something only people could do.

A **program** is a list of instructions the computer follows. The calculator on your phone is a program. The game you play is one too. AI is a special kind of program.

Notice the word "seemed". To do a division problem, you use your head. But nobody calls the calculator intelligent. Recognizing your grandma's face in a family photo, though, seems harder to explain. That is the kind of task where AI usually shows up.

In one sentence: AI is a program that does something we thought only a person could do.

AI is already in your life

You probably use AI several times a day without noticing. Here are some examples.

Where it shows up	What the AI does
Videos and music	Suggests what you might like, based on what you have already watched or listened to
Phone keyboard	Fixes words and suggests the next one while you type
Email	Sets aside the misleading ads, known as spam
Phone map	Picks a route and estimates how long the trip will take
Camera	Finds the faces in the photo so everything is in focus
AI assistants	Answer questions and write text, like ChatGPT, Gemini, or Claude

Lia's hunch was right. Video suggestions use programs that learn from what many people watch. If lots of people who watched the videos you watched also liked another one, it shows up for you.

The keyboard's autocorrect does something similar with words. It has seen a lot of text and knows which words tend to come after others. If you type "good", it suggests "morning". Hold on to this example, because it comes back in chapter 10.

The spam filter looks at each email and decides: does this look like a misleading ad or not? It learned that by seeing many emails that people had already marked as spam.

What AI is not

When someone mentions artificial intelligence, many people picture a movie robot. A metal robot that walks, talks, has feelings, and sometimes wants to take over the world.

Real AI is very different. Almost always, it has no body at all. It is a program running inside a computer, a phone, or a building full of computers. There are robots that use AI, but most AI isn't even a robot.

AI is not alive, either. It doesn't eat, sleep, or grow. It doesn't get bored when you stop using it.

And there is a trickier thing. An AI assistant writes "I think" or "happy to help". That doesn't mean it thinks or feels like a person. It learned to write sentences that look like the ones people write. Whether a machine could someday have something like feelings is a question scientists and philosophers are still debating. This book won't pretend to know the answer.

What we can say for sure is what AI **does**. And that is what the next chapters are about.

DID YOU KNOW

The term "artificial intelligence" was chosen by a group of researchers in the United States, in a proposal for a study meeting that took place in 1956. So the name is about 70 years old. The full story is in chapter 2.

It is not one single thing

Another important idea: AI is not a single machine or a single trick. It is a name that groups many different techniques.

Think of the word "sport". Soccer, swimming, and chess are all sports, but they are very different. Someone who is a great goalkeeper is not, because of that, a great swimmer. AI is similar.

The program that picks the route on the map works one way. The one that recognizes faces works another. The assistant that chats with you works yet another way. Some follow rules written by people. Others learn on their own from examples. Chapter 3 shows the difference between those two ways.

So be suspicious of phrases like "the AI decided" or "the AI thinks". Which AI? Made for what? Trained how? Those questions help a lot when you read the news.

What is AI good at?

Today's AI is usually very good at well-defined tasks. Sorting spam, recognizing a cat in a photo, suggesting the next word. At some of these tasks, it is faster than any person.

But it also makes mistakes, and sometimes strange ones. A program that recognizes animals can get confused by a photo no child would get wrong. An AI assistant can answer with total confidence something that is false. Chapter 12 explains why this happens.

So hold on to this idea: AI can be very useful and, at the same time, get things wrong. Both things are true together.

TRY IT

AI hunt around your house. You will need paper, a pencil, and an adult or friend to talk with.

1. Make two columns on the sheet: "I think it uses AI" and "I think it doesn't".
2. Walk around the house and write down devices and programs: fridge, TV, phone, microwave, game console, calculator, map, alarm clock.
3. For each one, ask: "Does this do something that seemed to need human intelligence?". Write it in the column you think is right.
4. Compare with someone else's list. Where did you disagree?

Expected result: some items are easy, like the calculator, which only follows arithmetic rules. Others will spark debate, like the TV that suggests movies. That debate is the most interesting part. Even the experts don't always agree on where AI begins.

Summing up

- Artificial intelligence is the name for programs that do tasks that seemed to require human intelligence.
- It is already part of everyday life: video suggestions, keyboard autocorrect, spam filter, maps, and assistants.
- AI is not a movie robot and it is not alive; it is a program that does tasks.
- AI is not one single thing: it is a name for many different techniques.
- It can be very useful and get things wrong at the same time.

Something to think about: if a program does something that seemed to require intelligence, does that mean it is intelligent? Or do we just change what we call "intelligent"?

CHAPTER 2

A Quick History of AI

Theo was playing chess against his phone and lost for the third time in a row.

"This program is way too good," he complained.

"Have computers been beating people at chess for long?" Lia asked.

"No idea. Was AI invented along with the phone?"

It wasn't. AI is much older than the phone, and even older than many people's grandparents. It has a history full of excitement, disappointment, and surprises. Let's run through it quickly.

A bold question

In 1950, the English mathematician Alan Turing published a paper with a famous question: "Can machines think?". He found the question hard to answer, because everyone understands "think" in their own way. So he proposed swapping the question for a game.

In the **imitation game**, a judge chats in writing with two hidden players: one is human and the other is a machine. If the judge can't figure out which is which, the machine passed the test. Today that game is known as the **Turing test**.

Turing had another idea that took a long time to become real. Instead of programming an "adult" machine that already knew everything, maybe it would be better to build a machine that **learned**, like a child at school. Hold on to that idea, because it is the heart of chapter 3.

The name is born at a summer meeting

In 1956, a group of researchers got together at Dartmouth College, in the United States, to study the subject over the summer. The name "artificial intelligence" came from the proposal they had written the year before. That is why many people

consider 1956 the birth year of AI as a field of study.

The mood was very optimistic. The researchers thought that, with effort, machines would soon do many of the things people do. In the following years, programs appeared that solved logic problems, did algebra, and played checkers.

It seemed like they were almost there. They weren't.

The AI winters

Many promises from that time weren't kept on schedule. Translating text between languages, for example, was much harder than it looked. To translate well, you need to know what the text is talking about, and the programs didn't have that knowledge of the world.

In the early 1970s, the governments of the United States and the United Kingdom cut a good part of the research money. This period of little money and lots of doubt got the nickname **AI winter**. It's a comparison: after the summer of excitement came the cold.

In the 1970s and 1980s, AI grew again with **expert systems**: programs full of rules written by people who knew a subject very well. They were rules like "if the patient has this symptom, then it may be this disease". They worked well on small subjects. But writing all the rules was a huge amount of work, and any case outside what was planned confused the program. In the late 1980s, a second winter came.

The lesson of the winters still matters today. When expectations rise too high, disappointment usually comes along.

Chess: the machine beats the champion

In 1997, a computer called **Deep Blue**, built by the company IBM, beat the world chess champion Garry Kasparov in a match of several games. A year earlier, Kasparov had beaten the machine. In the rematch, he lost. The news went around the world.

How did Deep Blue play? It analyzed millions of positions per second, imagining "if I do this, he does that, and then I do this other thing". To choose between positions, it used criteria prepared by chess experts and programmers. It was not a machine that learned on its own like a person. It was a machine that calculated a lot, very fast.

The turning point: lots of data and fast computers

Meanwhile, an old idea was gaining strength: programs that learn from examples. For that to work well, two things were missing. One was **data**, meaning lots of examples. The other was computers fast enough.

With the internet, a gigantic amount of photos, text, and video came into existence. And **graphics cards**, parts created to make video games look good and run fast, turned out to be great at doing the math these programs need. Theo thought that was awesome: video games helped AI grow.

Around 2012, a program of this kind, trained on graphics cards, won a big competition for recognizing objects in photos by a wide margin. From then on, learning from examples became the most common way to build AI. Chapters 4 through 7 explain how it works.

Go: the game that seemed impossible

Go is an ancient board game, widely played in China, Japan, and Korea. The rules are simple, but the number of possible games is so huge that calculating everything, the way Deep Blue did in chess, doesn't work.

In 2016, a program called **AlphaGo**, from the company DeepMind, beat Lee Sedol, one of the best Go players in the world. AlphaGo learned by studying games played by people and by playing against itself many times. It won or lost, and kept adjusting the way it played. Chapter 15 tells how that kind of learning works.

AI learns to handle words

In 2017, researchers at Google published a paper that introduced a new kind of program for handling text, called the **Transformer**. It was good at noticing which words in a sentence connect to which others. Almost all the AI assistants that chat with you today are built on that idea. Chapter 10 explains how.

In November 2022, the company OpenAI launched ChatGPT, an AI assistant anyone could use to chat. It became famous very fast. Then came others, like Gemini and Claude. Chapter 11 shows how an assistant like that is created.

DID YOU KNOW

Many of today's AI ideas are old. The first "artificial neurons" were imagined in the 1940s, even before the Dartmouth meeting. They went out of fashion, came back, fell out again, and only became stars decades later. In science, an idea that doesn't work today may work tomorrow, when better data and computers come along.

The timeline

When	What happened
1950	Alan Turing asks whether machines can think and proposes the imitation game
1956	The Dartmouth meeting and the name artificial intelligence
1970s	First AI winter
1970s and 1980s	Expert systems full of rules
Late 1980s	Second AI winter
1997	Deep Blue beats Garry Kasparov at chess
Around 2012	Learning from examples, using graphics cards, becomes the main path
2016	AlphaGo beats Lee Sedol at Go
2017	Paper that introduces the Transformer
November 2022	Launch of ChatGPT

TRY IT

The paper imitation game. You will need three people, paper, a pencil, and ten small cards.

1. Choose who will be the judge, who will be the "person", and who will be the "machine".
2. The "machine" writes ten ready-made answers on the cards, like "I don't know", "I like pizza", "Good question, what about you?", and "It's hot today".
3. The judge writes a question on a piece of paper. The "person" answers however they want. The "machine" can only hand over one of the cards.
4. A helper, or the pair themselves, hands the two answers to the judge marked only as X and Y.
5. Repeat with five questions. At the end, the judge says who the machine is.

Expected result: with ordinary questions, the machine sometimes fools the judge. With very specific questions, like "what did you have for lunch?", it gives itself away fast. Think together: how many cards would the machine need to never get caught?

Summing up

- In 1950, Alan Turing asked whether machines can think and proposed a game to test it.
- The name artificial intelligence became tied to the Dartmouth meeting, in 1956.
- Exaggerated promises led to periods of little money, called AI winters.
- Chess fell in 1997 to fast calculation; Go fell in 2016 to a machine that learned.
- Lots of data, graphics cards, and the **Transformer** paved the way for the assistants, which reached the public in 2022.

Something to think about: if you could ask a single question to find out whether you are talking to a machine, what would it be?

CHAPTER 3

Rules or Examples?

Lia showed Theo a drawing: a cat sleeping curled up, with no paws showing.

"You know it's a cat, right?" she asked.

"Of course. It's obvious."

"Then explain to me how you know. But explain it so well that even a computer would understand."

Theo opened his mouth, closed it again, and sat there thinking. It was harder than it looked.

Programs that follow rules

Think of a cake recipe, like the ones Theo's grandma keeps in a notebook. "Beat three eggs. Add the sugar. If the batter gets too thick, add more milk. Bake until golden."

A recipe is a list of steps, one after another. Some steps have a choice: **if** this happens, **then** do that. A traditional computer program is exactly that: a very detailed recipe, written by a person, that the computer follows without skipping anything.

This way of making programs works very well when we can write down all the rules. Here are some examples.

- The calculator follows the rules of addition and multiplication.
- The soccer video game follows rules: if the ball crosses the goal line, then it's a goal.
- The alarm clock follows a rule: if it's 7 o'clock, then ring.

The expert systems from chapter 2 were like this too, with many "if... then..." rules written by people.

In short: in the first way, a person writes the rules and the computer obeys.

When the rules break

Now go back to Lia's challenge. Try to write rules to recognize a cat in a photo.

A first try might go like this: "If it has four legs, whiskers, pointy ears, and a tail, then it's a cat". Sounds good. But look at the problems.

- A dog also has four legs, whiskers, ears, and a tail.
- Lia's cat is curled up, and its paws don't even show.
- Some cats have folded ears.
- A photo may show the cat from behind, in the dark, or half hidden behind the couch.
- A drawing of a cat has no real fur, and you still recognize it.

Every time you fix a rule, a new case shows up that breaks it. And there is an even bigger problem. The computer doesn't see "ears" or "whiskers". It sees an image made of little colored dots, as chapter 8 will show. Writing a rule that says which dots form a whisker is nearly impossible.

The funny thing is that you recognize a cat in less than a second. You just can't explain how you do it. Many things we do without effort, like recognizing faces, understanding a voice, or reading messy handwriting, are like that. We know how to do them, but we don't know how to write the recipe.

Programs that learn from examples

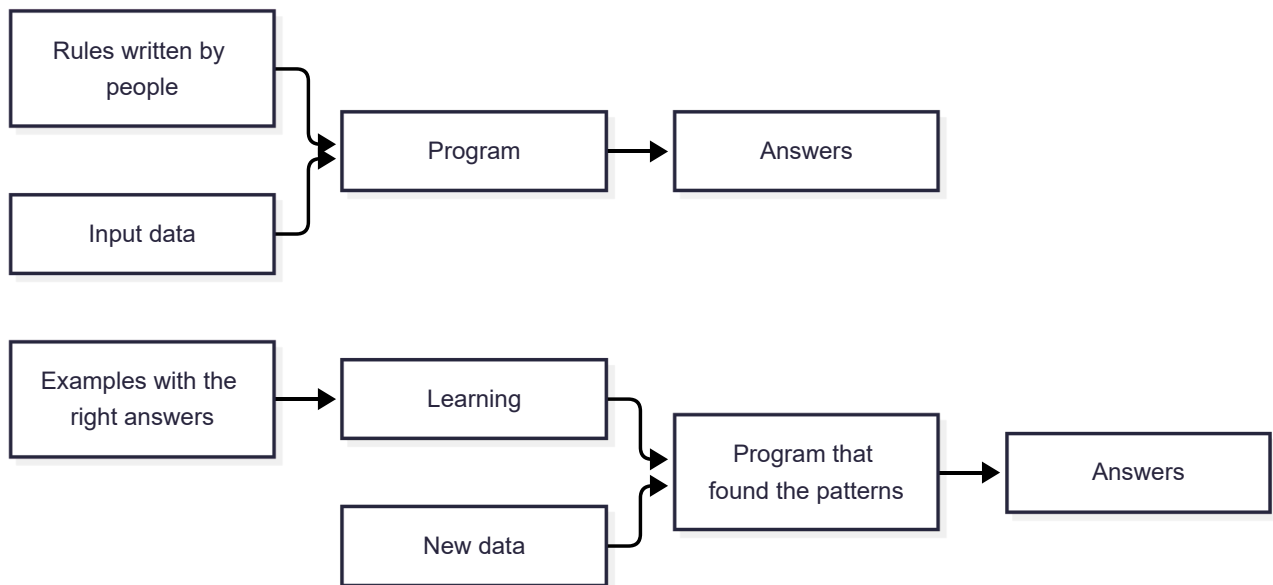
What if, instead of writing the rules, we showed examples?

Think about how a small child learns what a cat is. Nobody hands them a list of rules. Someone points and says "look, a cat!". Then points at another cat, and another. They also point at a dog and say "that one's a dog". After many examples, the child gets it right on their own, even with cats they have never seen.

The second way of making programs follows this idea. We show the computer many photos labeled "cat" and many labeled "not a cat". The program searches on its own for the patterns that separate one group from the other. Then it gets a new photo and says which group it seems to fit in.

This is called **machine learning**: instead of receiving ready-made rules, the computer finds patterns from examples. Remember Turing's idea from chapter 2, of a machine that learns like a child? This is it.

The comparison with a child helps, but it has limits. A child learns from a few examples and understands much more at the same time. The program usually needs lots of examples, and it learns only the task it was trained for.



In short: in the second way, the person gives the examples and the computer figures out the rules on its own.

The two ways, side by side

	Rules	Examples
Who does the main work	A person writes the rules	The program looks for patterns in the examples
Good for	Tasks with clear rules, like arithmetic	Tasks that are hard to explain, like recognizing faces
When it gets something wrong	You can read the rule and find the mistake	Sometimes it's hard to know why it went wrong
What it needs	Someone who knows the subject well	Lots of good examples

Look at the third row. In a rules program, if it makes a mistake, you can open the recipe and find the wrong step. In a program that learned from examples, the patterns it found are not written in words. Understanding why it decided something can be very hard. Scientists study this problem a lot.

Look at the last row too. If the examples are bad, the program learns wrong. If you only show orange cats, it may think a black cat is not a cat. Chapter 4 is all about that.

DID YOU KNOW

The spam filter is a good example of how the two ways meet. A rule like "if the email has the word prize, then it's spam" gets a lot wrong, because your uncle might write about the prize for the championship. That is why spam filters usually learn from examples of emails that people have marked as spam. Many real systems mix the two ways: they learn from examples and still follow some rules written by people.

So are rules a thing of the past?

No. Both ways are still used, each where it works best. Nobody needs machine learning to build a calculator. The rules of addition are clear, and a rules-based calculator doesn't get a simple sum wrong.

But for recognizing cats, understanding speech, or chatting, learning from examples won by a mile. Almost everything you will read in the rest of this book uses this second way. The next chapters open the box and show how learning really happens.

TRY IT

Sorting fruit two ways. You will need paper, a pencil, and at least one friend or adult. If you have real fruit at home, even better.

1. Cut out 12 cards. On each one, draw or write a fruit with its color: yellow banana, green banana, red apple, green apple, orange, lemon, tangerine, purple grape, green grape, strawberry, papaya, and pear.
2. **Rules round.** Write rules to sort the fruit into "banana", "apple", and "other". For example: "if it's yellow and long, then it's a banana".
3. Test your rules on all the cards. Write down where they failed. Did the green banana pass? Where did the green apple end up?
4. **Examples round.** Now make three piles with two cards each, one pile for each group. These are the examples. Don't write any rules.
5. Your friend takes the leftover cards, one at a time, and puts each on the pile with the most similar fruit. They can't see your rules.
6. Compare the two rounds.

Expected result: rules usually break on cases you didn't plan for, like the green banana. With examples, your friend gets many cases right without a single rule. But they can also get things wrong if the examples are too few or too similar. Think: what examples would you add so they make fewer mistakes?

Summing up

- There are two ways to make a program: write rules or learn from examples.
- Rules work well when the task has clear steps, like arithmetic or a recipe.
- For tasks that are hard to explain, like recognizing a cat, rules break all the time.
- Learning patterns from examples is called machine learning.
- Both ways are still in use, and many systems mix the two.

Something to think about: what is something you do very well but couldn't explain in rules to another person?

CHAPTER 4

Data: Food for AI

Theo was in the kitchen with Grandma, making bread. She never measures anything. "How do you know the dough is right, Grandma?" Grandma laughed: "I've made this bread about a thousand times. My hands learned." Theo got to thinking: if Grandma learned from so many tries, could a computer learn that way too? And learn from what?

In chapter 3, you saw that there are two ways to make a program: write rules or learn from examples. This chapter is about the examples. Without them, machine learning has nothing to learn from. That is why people say data is the food of AI.

What data is

Data is information stored in a way the computer can use. It can be almost anything. A photo is data. So is a text. Your team's list of goals in the championship, each day's temperature, the songs you skipped in the app: all of that is data.

A lot of data is created without us noticing. Every time you watch a video to the end, like a photo, or skip a song, a record is left behind. Those records help programs figure out what you like. That is how many video and music recommendations work.

For the computer, in the end, everything becomes numbers. A photo becomes a grid of numbers, and a text becomes numbers too. You will see how that happens in chapters 8 and 9. For now, it's enough to know: data is the raw material, and AI learns by looking at it.

One-sentence summary: data is the examples AI uses to learn.

Examples with the right answer

Imagine you want to teach a program to sort photos of dogs and cats. Just showing photos isn't enough. It needs to know which is which. So each photo comes with a tag: "dog" or "cat".

That tag is called a **label**: it is the right answer that comes with the example. A labeled example is like an exercise that comes with the answer key. The program looks at the photo, makes its guess, and then checks against the label. If it got it wrong, it adjusts itself. How that adjustment happens is the subject of chapter 5.

Example	Label
Photo of an animal with floppy ears	dog
Photo of an animal with whiskers and green eyes	cat
Email saying "you won a prize, click here"	spam
Email from the teacher about the test	not spam

Who adds the labels? Often, people. There are people whose job is to look at images and texts and mark the right answer, one by one. Other times the label appears on its own. When you mark an email as spam, for example, you are giving the filter a label.

DID YOU KNOW

You know those tests on the internet that ask you to click on all the squares with traffic lights or bicycles? They exist to prove you are a person, not a robot. But some versions of those tests were also used to collect people's answers, which became labels to help computers learn.

Quantity and quality

Theo's grandma learned the feel of the dough because she made lots of bread. AI is similar: in general, more examples help. Someone who has seen only three cat photos doesn't know much about cats. Someone who has seen thousands, of every kind, learns much better what makes a cat a "cat".

But quantity isn't everything. The examples also need to be good. Imagine studying for a math test with a notebook full of problems with the wrong answers. You would learn wrong, even studying hard. If half the "dog" photos actually show cats, the program will get confused.

People who work with AI have a saying for this: **garbage in, garbage out**. If the input data is bad, the result will be bad too. No program, however clever, can get something good out of wrong examples.

One-sentence summary: many examples help, but only if they are the right examples.

When the examples are all the same

Now a more hidden problem. Lia loves soccer. Imagine she teaches a program to recognize "ball" using only photos of soccer balls, black and white. The program learns just fine. Until someone shows it an orange basketball. And the program says: "that's not a ball".

The program did nothing wrong with what it had. It just never saw a different ball. The examples were all of one kind. This is called **bias**: when the data leans to one side and leaves others out, and the AI starts getting things wrong exactly on those others.

With balls, the mistake is funny. With people, it can be serious. There have been face recognition systems that worked worse on people with darker skin. One of the reasons pointed out was having few photos of those people in the training examples. That is why people who build AI need to ask themselves: "who was left out of my data?".

Bias doesn't always come from a lack of examples. Sometimes the examples copy unfairness that already exists in the world. If a program learns from old decisions that were unfair, it can repeat the same unfairness. AI doesn't know what is fair. It only imitates the pattern it found. You will come back to this subject in chapter 16.

TRY IT

The bird detector. You need paper, a pencil, and at least one friend.

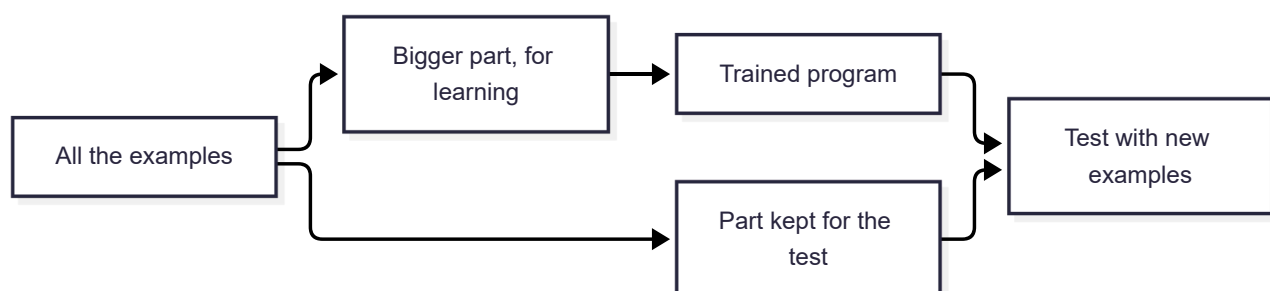
1. Draw five "example" birds for your friend. All small, with wings spread, flying.
2. Tell your friend: "This is a bird. Now you are the detector. You can only use what you saw in these drawings."
3. Without them seeing, draw three new creatures: a small bird flying, a penguin standing up, and an airplane.
4. Ask your friend to say whether each one is a bird or not, going only by what they learned from the examples.

What should happen: the "detector" will very likely accept the airplane, because it has spread wings and flies. And reject the penguin, which is a bird but doesn't look like the examples. The examples had bias: they only showed one kind of bird. Afterward, repeat with more varied examples and see if the detector improves.

The test the AI has never seen

Remember the exercise notebook with the answer key? Imagine the teacher gives exactly the same exercises on the test. Everyone gets a perfect score. But that doesn't show whether the class learned. Maybe they just memorized the answers.

The same happens with AI. If you test the program with the same photos it used to learn, it may do very well just because it "memorized". That is why people who train an AI do something simple and clever. Before starting, they set aside part of the examples and keep them in a drawer. The program never sees those examples while it learns.



Those saved examples form the **test set**: a test with questions the AI has never seen. Only at the end, after training, does the program take that test. If it gets plenty right there too, it's a sign it really learned, and didn't just memorize. If it does badly, something went wrong along the way.

There is an important rule: nobody can "peek" at the test. If you tweak the program while looking at the test answers, it stops being a fair test. It would be like the teacher showing the test the day before. The result looks great, but it's worth nothing. In chapter 5 you will see what happens when a program memorizes instead of learning.

One-sentence summary: to know whether the AI learned, it needs to be tested with examples it has never seen.

CAUTION

Your data is data too. Photos, your name, your address, the school you go to: all of that can end up somewhere when you use apps. Before posting or sending something, think about whether you would be comfortable seeing it stored for years. When in doubt, ask a trusted adult.

Summing up

- Data is the examples AI uses to learn: photos, texts, numbers, clicks.
- The label is the right answer that comes with the example, like an answer key.
- Many examples help, but wrong examples teach wrong: garbage in, garbage out.
- Bias is when the data leaves someone or something out, and the AI gets things wrong exactly there.
- Part of the examples is kept aside as a test, to see whether the AI really learned.

Something to think about: if you were going to teach an AI to recognize "tasty food", what photos would you use? Whose taste would it be learning?

CHAPTER 5

How a Machine Learns: Making Mistakes and Fixing Them

Lia was practicing free kicks in the backyard. The first kick went way too high. The second came out low and to the left. On the third, she aimed a little more to the right and kicked with less power. It almost went in. Theo, sitting on the wall, asked: "Do you do math before you kick?" Lia shrugged: "No. I just see where I missed and change it a tiny bit."

Without knowing it, Lia had just described how almost every modern AI learns. In this chapter, you will see how that works.

A machine full of knobs

Think of an old radio with knobs you turn. One changes the volume, another finds the station, another makes the sound deeper or sharper. Depending on where the knobs are, the sound comes out good or bad.

In AI, the program that learns is called a **model**: a kind of machine that takes something in and gives an answer back. It takes a photo and answers "cat". It takes a sentence and answers whether it is friendly or rude. And the model has knobs, just like the radio.

Those knobs are not real. They are numbers stored inside the program. Each number changes a little how the model answers. People who work with AI call these numbers **parameters**. A small model may have only a few. The big models, the ones that chat with us, have a gigantic number of them.

At the start, nobody knows where to set each knob. So they begin in random positions, almost like a lottery. With the knobs like that, the model gets almost everything wrong. Learning is finding out where to set each knob so it gets things right.

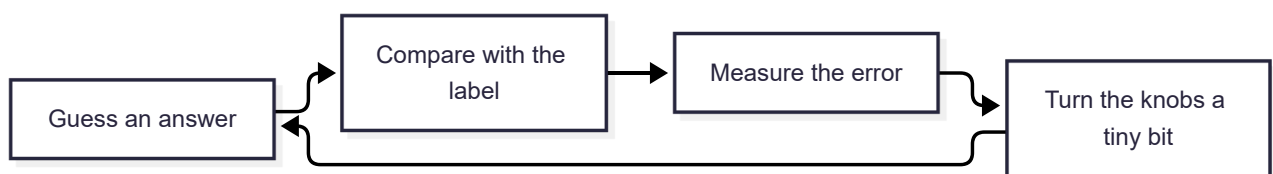
One-sentence summary: a model is a machine with knobs, and learning is finding the right position for them.

Guess, measure, adjust

Have you ever played "hot or cold"? Someone hides an object, and you look for it. At each step, the person says "hot" if you got closer or "cold" if you moved away. You don't know where the object is. But by following the hints, you get there.

The model learns in a similar way. It repeats four steps:

1. **Guess.** The model gets an example, such as a photo, and gives an answer.
2. **Measure the error.** The answer is compared with the label, the right answer you met in chapter 4. Out comes a number that says how big the mistake was.
3. **Adjust.** Each knob is turned a tiny bit, in the direction that would make the error smaller.
4. **Repeat.** Another example comes in, and it all starts again.



This is **training** the model: showing it many examples, measuring the error and adjusting, over and over. Each round improves things only a little. But there are thousands of rounds, sometimes millions. Added together, the model that got almost everything wrong starts getting a lot right.

Notice one important detail: the adjustment is small. Why? Because an adjustment that is too big can overshoot. It is like Lia kicking: if she corrects too far to the right, the ball ends up wide on the other side. Small steps are slower, but safer.

Walking down a hill blindfolded

There is another way to picture training. Imagine you are at the top of a hill, blindfolded. Your goal is to reach the bottom, in the valley. You can't see a thing. What do you do?

You can feel the ground with your foot. If the ground slopes down ahead of you, you take a step forward. If it slopes down to the left, you go left. Step by step, always toward the side that goes down, you reach the valley.

In AI training, the height of the hill is the size of the error. Up at the top, the model makes lots of mistakes. Down at the bottom, very few. The knobs are your position on the hill. Turning the knobs is taking a step. And the computer can calculate which way the ground slopes, that is, which way to turn each knob so the error shrinks.

This method has a technical name, **gradient descent**, which means exactly that: going down by following the slope. You don't need to remember the name. Remember the idea.

Where does the comparison break down? A real hill has only a few directions you can walk in. A big model has so many knobs that this "hill" has an enormous number of directions. No person can picture that. But the rule is the same: feel the slope and take a small step down.

One-sentence summary: training is walking down the hill of error, one small step at a time.

TRY IT

Hot or cold with numbers. You need a friend, paper and a pencil.

1. Your friend thinks of a number from 1 to 100 and secretly writes it down. That is the "right knob".
2. You guess any number. That is the model starting with its knobs set at random.
3. Your friend answers only "higher" or "lower". That is the error measurement.
4. You adjust your guess in the direction they pointed. Write every guess down in a list.
5. Repeat until you get it. Then switch roles.

What should happen: you will reach the number in a few rounds, even though you started in the dark. Now try again with one extra rule: your friend also says "far" or "close". You should get it faster, because now you know how much to adjust, not just which way. The computer does something similar: it knows the direction of the adjustment and, more or less, its size.

Memorizing is not learning

Now for a trap. Picture two students studying for a science test. The teacher handed out a list of twenty practice questions.

The first student memorized the twenty answers, word for word. The second tried to understand why each answer is what it is. Practicing with the list, both get everything right. Then the test comes, with questions that are similar but different. The first student does badly. The second does well.

The same thing happens with AI. A model with many knobs can end up "memorizing" the training examples. It gets all of them right, even the details that don't matter. But when it sees a new example, it fails. This is called **overfitting**: the model fit the training examples too tightly and did not learn the general rule.

Here is a made-up example to get the idea. Imagine a model built to recognize cows, trained only with photos of cows in green pastures. It might "learn" that a cow is anything with grass behind it. Show it a cow on the beach, and it fails. It memorized the background instead of learning the cow.

This is where the test from chapter 4 comes in. Remember the examples kept in the drawer? They are there to catch overfitting. If the model does great on the training examples and badly on the saved ones, it memorized.

	Training examples	Saved examples
Really learned	does well	does well
Memorized	does very well	does badly
Learned nothing	does badly	does badly

How do you avoid it? A few simple ideas help. Use more examples, and more varied ones, as you saw in chapter 4. Don't train for too long. And keep an eye on the test score: when it stops improving, it is time to stop. In practice, people who build AI use other tricks too, but that is the basic idea.

One-sentence summary: a good model gets new examples right, not just the ones it has already seen.

CAUTION

The same trap applies to you. Copying an AI assistant's answer into your homework is like memorizing the teacher's list: it looks like it worked, until the test comes. Use AI to understand, not to copy. Chapter 16 says more about this.

Summing up

- A model is a machine with knobs, the parameters, which start in random positions.
- Training means repeating: guess, measure the error, turn the knobs a tiny bit so the error shrinks.
- It is like walking down a hill blindfolded, always toward the side that goes down.
- Overfitting is memorizing the training examples instead of learning the general rule.
- The examples saved for the test show whether the model learned or just memorized.

Something to think about: do you learn better by making mistakes and fixing them, or by reading the right answer straight away? Why do you think the machine needs to make so many mistakes to learn?

CHAPTER 6

Artificial Neurons

Saturday morning. Theo wanted to go to the park, but he wasn't sure. "It's kind of cloudy. I have Portuguese homework. But Lia is going to be there with her new ball." He stood looking out the window, weighing each thing. Lia, on the phone, asked: "So, are you coming or not?"

Without realizing it, Theo was doing what a tiny piece of AI does. It is called an **artificial neuron**. It is the basic building block of many modern AIs, and understanding it is easier than it looks.

A vote with weights

Let's look at Theo's decision slowly. He has three clues, things that help him decide:

- Is it going to rain?
- Is there homework to do?
- Is Lia going to be there?

Not every clue matters the same. For Theo, Lia being at the park counts a lot in favor. Rain counts quite a bit against. Homework counts a little against, because he can do it in the afternoon.

So he gives each clue an importance score. That score is called a **weight**: a number that says how much that clue pulls the decision, and in which direction. A positive weight pulls toward "yes". A negative weight pulls toward "no". A big weight pulls hard. A small weight barely makes a difference.

Clue	Is it happening?	Weight for Theo
Lia will be there	yes	+4
It will rain	maybe, counts half	-6
There is homework	yes	-1

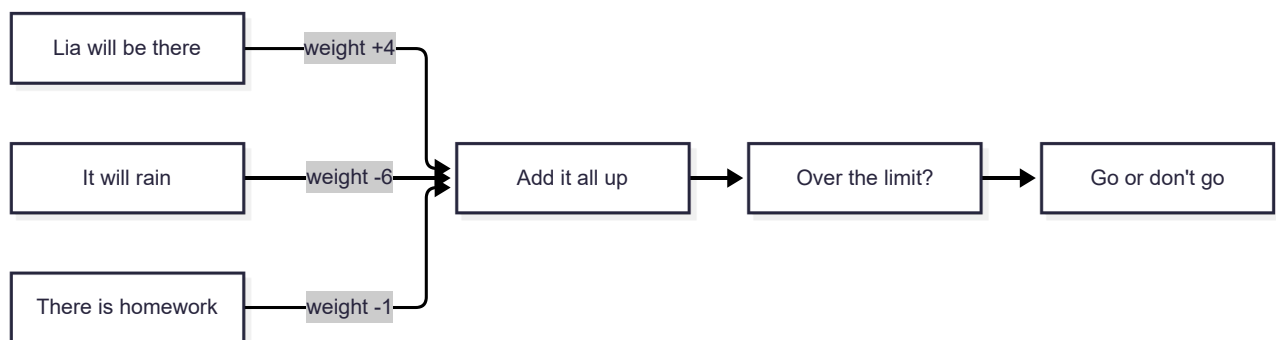
Now just add it up. Lia is worth +4. Rain counts half, so it is worth -3. Homework is worth -1. All together: 4 minus 3 minus 1 makes zero.

One last part is missing: the decision rule. Theo decided he goes to the park if the total is more than zero. It came out exactly zero. So, just barely, he stays home. If rain were a "no", the total would be 3, and he would run out the door.

One-sentence summary: an artificial neuron gives each clue a weight, adds everything up and decides.

The artificial neuron

What Theo did is exactly the job of an artificial neuron. It is a small calculation: it takes in clues, multiplies each one by its weight, adds it all up and decides what to pass along.



In computers, the clues are always numbers. It could be the brightness of a tiny piece of a photo, how often a word shows up in a text, the day's temperature. The neuron doesn't know what those numbers mean. It only multiplies, adds and compares.

In reality, it is a bit more complicated. The decision is not always a flat "yes" or "no". Often, the neuron passes along a number that says "how much yes". But that is the idea: weigh, add, decide.

The weights are the knobs

Remember the machine full of knobs from chapter 5? Well, the weights are those knobs. In an artificial neuron, nobody writes the weights by hand. They start out random and get adjusted during training, by making mistakes and fixing them.

Imagine Theo is a neuron that is learning. On the first Saturday, he gives rain a small weight, goes to the park and comes back soaked. Mistake! Next time, the weight for rain becomes more negative. On another Saturday, he stays home and later finds out the game with Lia was great. Mistake again. Lia's weight goes up a little.

After many Saturdays, the weights get good. That is how a neuron learns: not by gaining new rules, but by adjusting the weights of the clues it already has.

One-sentence summary: for a neuron, learning is adjusting the weights until the decisions come out good.

TRY IT

The card neuron. You need paper, scissors, a pencil and one or two friends.

1. Pick an everyday decision, such as "am I going to play soccer today?".
2. Cut out three clue cards. Write one clue on each, such as "it's sunny", "the team is complete" and "I have a test tomorrow".
3. On the back of each card, write a weight from -5 to +5. For "I have a test tomorrow", maybe -4. For "it's sunny", maybe +3.
4. Agree on the rule: if the total is above zero, the answer is "yes".
5. Your friend makes up a situation, saying which clues are happening. Flip only those cards, add up the weights and announce the decision.
6. Your friend says whether they agree. If not, change one weight by one point and test again.

What should happen: after a few rounds, your "neuron" will agree with your friend almost every time. You just trained it by hand. Now switch friends: the weights that were good for one may not work for another, because every person decides in their own way.

Inspired by the brain, but very different

Why the name "neuron"? Your brain has cells called neurons. They receive signals from other cells and, depending on the signals, send a signal onward or not. A long time ago, scientists thought this idea could be imitated with a simple calculation. The name stuck.

But the comparison stops there. An artificial neuron is just a calculation: multiply, add, compare. A real neuron is a living cell, full of parts, that works in ways scientists are still studying. The brain is also organized very differently from the networks you will see in chapter 7.

So, when you hear "artificial neuron", don't picture a little piece of brain inside the computer. Picture a weighted-vote calculator. The name came from an old inspiration in the brain, not from a copy of it.

DID YOU KNOW

The idea of an artificial neuron that learns by adjusting its own weights is old. In 1958, the American researcher Frank Rosenblatt presented the **perceptron**, an artificial neuron that could learn from examples. At the time, it caused a lot of excitement. But a single neuron can do very little. Many of them had to be joined together, and that took decades to work.

What a single neuron can't do

A single neuron is good for simple decisions, where each clue pulls in one direction. But many decisions are not like that.

Think about this rule on a team: "I play if the goalkeeper comes or if the striker comes, but not if both come, because we only have one spare jersey". Try to find weights for that. If the goalkeeper has a positive weight and so does the striker, the sum of the two together gets even bigger. The neuron will say "yes" exactly when the answer is "no". There is no way to choose the weights that solves this with just one neuron.

The solution is to join several. One neuron can handle "at least one came". Another can handle "both came". A third looks at the first two and decides. Neurons together can do things a single one can't. That is the subject of the next chapter.

Summing up

- An artificial neuron takes in clues, which are numbers.

- Each clue has a weight, which says how much it pulls the decision and in which direction.
- The neuron multiplies each clue by its weight, adds it all up and decides what to pass along.
- The weights are the knobs from chapter 5: learning is adjusting them.
- The name comes from the brain, but an artificial neuron is just a calculation, very different from a living cell.

Something to think about: when you decide something, do you "give weights" to the clues? Can all of your decisions be explained that way, or do some not fit into a sum?

CHAPTER 7

Neural Networks: Many Neurons Together

Lia was drawing a cat for Theo. First she made a few loose strokes. Then she joined the strokes into shapes: two triangles, a circle, a long curve. Finally, the shapes turned into ears, a head and a tail. "There, a cat." Theo looked and said: "Funny. You started with little lines and ended up with an animal."

A neural network that recognizes images does something similar, but in the opposite direction. It starts by seeing little lines and ends by saying "cat". In this chapter, you will see how many neurons together manage to do that.

Neurons in layers

In chapter 6, you saw that a single neuron does very little. It weighs a few clues, adds them up and decides. The big idea is to join many neurons together in an organized way.

They sit in rows called **layers**. Each layer receives the numbers the previous layer passed along. It does its calculations and passes the result on to the next one. A set of neurons connected like this is called a **neural network**.



Think of a sandwich assembly line. The first person cuts the bread. The second adds the filling. The third wraps it up. Nobody makes the whole sandwich, but at the end it comes out ready. In the network, each layer does one piece of the work and hands it to the next.

The first layer receives the input, which might be a photo turned into numbers. The last layer gives the answer, the **output**. The layers in the middle are sometimes called **hidden layers**, because nobody looks directly at what they do.

One-sentence summary: a neural network is a bunch of neurons organized in layers, each one passing its result to the next.

From simple to complicated

What does each layer learn to do? Nobody tells it. That shows up on its own during training. But when scientists peeked inside networks trained to recognize images, they saw a beautiful pattern.

The first layers react to very simple things: edges, straight lines, curves, patches of color. The next layers join those things into shapes, such as circles, corners and fur textures. Further along, the shapes become parts: an eye, a pointy ear, a wheel. Near the end, the parts become objects: cat, bicycle, face.

It is Lia's drawing in reverse. She went from strokes to cat. The network goes from the photo to the strokes, from the strokes to the shapes and from the shapes to "cat".

In reality, it is a bit messier. Not every layer does something so clear, and a lot of what happens in there is hard to interpret. Understanding what goes on inside networks is something scientists are still studying. But the idea of "from simple to complicated" helps a lot in understanding why layers work.

TRY IT

The network of people. You need at least four people, paper and pencils.

1. One person, the "artist", secretly draws a simple object: a house, a fish or a sun.
2. The **layer 1** person looks at the drawing and writes down only the strokes they see, without saying what it is: "three straight lines", "a circle", "several short lines coming out of the circle".
3. The **layer 2** person reads only layer 1's notes and writes down shapes: "a triangle on top of a square".
4. The **output** person reads only layer 2's notes and says what they think the object is.
5. Check against the drawing. Then switch roles and try with another object.

What should happen: most of the time, the output gets it right, even without seeing the drawing. Each layer only passed along a summary of what it saw. Sometimes it goes wrong, and it is worth finding out in which layer the information got lost.

Deep learning

The first neural networks had very few layers. Over time, researchers discovered that networks with many layers could learn much harder things.

That got a name: **deep learning**, which means learning with networks of many layers. "Deep" here does not mean "wise" or "philosophical". It is deep like a building with many floors, or a very deep well. The more layers, the "deeper" the network.

Almost all of today's famous AIs use deep learning. The ones that recognize faces, the ones that turn speech into text, the ones that translate and the ones that chat. Each has its own organization, but deep down, all of them are many neurons in many layers.

How the error travels back into the network

Remember training from chapter 5? Guess, measure the error, turn the knobs. In a network, the knobs are the weights of all the neurons, in all the layers. And here a problem shows up.

When the network gets something wrong, we only see the error at the end, at the output. But the blame could be anywhere. Maybe a layer near the beginning mixed up an edge. How do you know how much each weight, deep inside, contributed to the error?

Think of a soccer game in which the team let in a goal. The goalkeeper let the ball in, but it is not only his fault. The coach replays the move: the defender was out of position, the midfield lost the ball before that. Each one gets their share of the blame, and each one corrects a little.

The network does something similar. The error starts at the output and is passed back, layer by layer, all the way to the beginning. Each weight gets its share of the blame and is adjusted a tiny bit. This method of carrying the error backward has a technical name: **backpropagation**. You don't need to know the math. The idea is: the error travels back through the network, and each weight corrects itself according to its share.

One-sentence summary: during training, the error at the output travels back through the network, and each weight gets its share of the correction.

Why it took so long

Neural networks have existed for decades. Why did they only get so good fairly recently? Two things were missing.

The first was data. Networks with many layers have many weights to adjust and need lots of examples. With only a few, they memorize, like the overfitting in chapter 5. With the internet, an enormous amount of photos and text became available. You saw in chapter 4 why that matters.

The second was computing power. Training a big network means doing an absurd number of small calculations: multiply and add, over and over. And then came unexpected help from video games.

DID YOU KNOW

To show a game on the screen, the computer has to calculate the color of many tiny dots at the same time, several times per second. That is why **graphics cards** were created: parts specialized in doing many simple calculations at once. Researchers noticed that training neural networks is also doing many simple calculations at once. Graphics cards were a perfect fit. Today, big AIs are trained on enormous numbers of these cards.

Theo would love to know this: the same part that makes his game run helped AI make a leap. Combining lots of data, graphics cards and a few new ideas about how to build and train networks, deep learning started winning challenges that once seemed impossible. Chapter 2 tells a bit of that story.

Summing up

- A neural network is made of many neurons organized in layers.
- For images, the first layers find simple things, and the following ones join them into shapes, parts and objects.
- Deep learning is learning with networks of many layers.
- During training, the error travels back from the output into the network, and each weight corrects itself a little.
- It worked well once there were lots of data and powerful computers, such as graphics cards.

Something to think about: if nobody tells each layer what to look for, and it learns anyway, who "taught" the network? The data, the people who built the network, or both?

CHAPTER 8

How a Computer Sees

Lia took a photo of her own drawing with her aunt's phone. Right away, the photo app wrote underneath: "drawing, paper". Lia was suspicious. "How does it know? It doesn't even have eyes." Theo shrugged: "No brain either. But it got it right."

They are both right. The computer has no eyes and no brain. Even so, it can say what is in a photo. This chapter shows how that is possible. And also why, sometimes, it gets things wrong in a very weird way.

A photo is a grid of little squares

Take a photo on a phone and keep zooming in, more and more. At some point, the image stops looking like a photo. It turns into a bunch of colored squares, side by side, like a mosaic of tiles.

Each of those little squares is called a **pixel**: the smallest piece of a digital image, with just one color. A photo is a huge grid of pixels, organized in rows and columns. From a distance, our eyes blend it all together and see a face, a ball, a cat.

In short: a digital image is a mosaic of pixels.

Every color is a recipe of three numbers

Now comes the important part. The computer does not store "blue" or "yellow" as a word. It stores **numbers**.

Almost all screens make colors by mixing three lights: red, green and blue. That is the famous **RGB**, from the initials of *red*, *green* and *blue*. Each pixel has three numbers: how much red, how much green and how much blue.

In the most common way of storing images, each of those numbers goes from 0 to 255. Zero means "light off". 255 means "light at full power". Here are some examples:

Color you see	Red	Green	Blue
Black	0	0	0
White	255	255	255
Bright red	255	0	0
Yellow	255	255	0
Medium gray	128	128	128

Look at yellow: it is red plus green, with no blue. With light, mixing works differently from paint. If Lia mixes red and green paint on paper, she gets a dull brown, not yellow.

In short: each pixel is a little recipe of three numbers.

The network only sees numbers

So, when a photo goes into a neural network, what really goes in? A gigantic list of numbers. Row by row, pixel by pixel, three numbers for each one.

The network does not "see" the photo the way you do. It does not know there is a cat in there. It receives numbers and does calculations with them, using the weights it learned in training (chapters 5 and 6).

Remember the layers from chapter 7? In a network that looks at images, the first layers usually react to simple things. For example, a place where light suddenly turns dark, which is an **edge**. The next layers join edges into shapes, such as curves and corners. Closer to the end, combinations show up that look like ears, eyes or wheels. The last layer gives the answer: "cat", "dog", "bicycle".

Nobody programmed "a cat's ear is shaped like a triangle". The network arrived at those detectors on its own, adjusting its weights with many labeled photos (chapter 4). In reality, what each layer learns is not always as tidy as this description. But that is the general idea.



In short: the network turns pixel numbers into answers, step by step, layer by layer.

TRY IT

Turn a letter into numbers. You will need graph paper (or draw a grid of 8 by 8 squares), a pencil and a friend.

1. Pick a simple letter, such as T, L or X, without telling anyone.
2. Fully color in the squares that form the letter. Each square is a pixel.
3. On a separate sheet, write the grid as numbers: 1 for a colored square and 0 for an empty one. One row of the grid becomes one row of numbers.
4. Hand only the sheet of numbers to your friend. They have to figure out the letter.
5. Now switch: they make a letter and you guess.

What should happen: with a little effort, your friend finds the letter by looking for where the 1s are. That is what the computer receives: only numbers. It was the drawing that turned into numbers, not the other way around. A real photo is the same, only with many more pixels and three numbers in each one.

Where this shows up in everyday life

This ability to recognize things in images is called **computer vision**: the part of AI that works with photos and videos. It is in many places.

- **Phones** that sort your photos by "beach", "food" or "dog".
- **Face unlock**, which compares your face with the one stored on the device.
- **Cars** that read traffic signs or notice that the car ahead has braked.
- **Apps** that recognize a plant or translate a menu when you point the camera at it.
- **Medical exams**, such as X-rays and photos of skin, where the AI marks suspicious areas.

In the case of medicine, one important point: the AI **helps** the doctor, it does not take the doctor's place. It can draw attention to a detail that might go unnoticed. But the one who examines the patient, talks with the family and decides on the treatment is the person trained for that. AI makes mistakes too, and someone has to check.

In short: recognizing images is already part of phones, cars and hospitals, always with people nearby.

Mistakes no person would make

Here is where things get curious. Because the network only sees numbers, it can go wrong in ways no person would.

Researchers have shown that a special sticker, placed next to an object, can confuse the network. To us, the sticker looks like just a strange colored blotch. To the network, those numbers weigh so much that it changes its answer. In one well-known experiment, a banana was classified as a toaster because of a sticker like that.

Other studies placed small black and white stickers on a stop sign. A person keeps reading "stop" with no trouble. But the system being tested started reading the sign as something else. These tricks are called **adversarial attacks**: changes made on purpose to fool the AI.

There are also mistakes with nobody trying to fool anything. Imagine a network trained on photos in which every cow is standing in a green pasture. It may end up learning "green in the background" instead of "cow". Show it a cow on beach sand and it fails. That is the data problem from chapter 4: the network learns what was in the examples, including what it shouldn't.

Why does this happen? Because the network does not know what a banana is. It learned patterns of numbers that usually show up in photos of bananas. If someone changes the numbers in just the right way, the pattern changes, even though the banana is still there. That is why scientists test these systems so carefully before putting them in cars or hospitals.

DID YOU KNOW

A person recognizes a cat drawn with three strokes, a stuffed cat and a real cat. For many networks, those three are much harder to put together. They learn best the kind of image they saw in training. Simple drawings, like Lia's, sometimes fool a network trained only on photos.

In short: the network gets a lot right, but it fails in strange ways, because it only learned patterns of numbers.

Seeing is not understanding

When the app wrote "drawing, paper", it was not admiring Lia's drawing. It did calculations with the pixel numbers and picked the labels that matched best. That is impressive. But it is different from what happens when you look at a drawing and remember a story.

At the same time, it is best not to go too far the other way. What happens inside these networks is still being studied by many scientists, and not everything is explained. The most honest way to talk about it is to describe what it does: it receives numbers, does calculations, returns an answer.

In the next chapter, you will see that words go through a similar path. Before an AI reads a text, the words also have to become numbers.

Summing up

- A digital image is a grid of **pixels**, and each pixel holds three numbers: red, green and blue.
- The neural network receives only those numbers and, layer by layer, arrives at an answer.
- Recognizing images helps in phones, cars and medical exams, always with people checking.
- Because it only sees numbers, the network can be fooled by stickers and details a person would ignore.

Something to think about: an AI can tell what a drawing shows, but it does not know who drew it or why. Did it "see" the drawing the same way you do?

CHAPTER 9

Words Become Numbers

Theo was typing out one of his grandma's recipes to send to Lia. "Cornmeal cake: three eggs, one cup of..." He stopped. "If the computer only understands numbers, how does an AI assistant read what I write?"

Good question. In chapter 8, you saw that a photo becomes a grid of numbers. Something similar happens with text. But the path is more curious, because words have meaning. And the computer has to store that meaning in numbers, somehow.

First step: cut the text into pieces

Before anything else, the text is cut into pieces. Each piece is called a **token**: a little chunk of text that the model treats as one unit. A token can be a whole word, part of a word, a punctuation mark or even a space.

Very common words usually become a single token. Rarer words are broken into pieces. For example, a long word like "disembarked" might become something like "dis", "embark" and "ed". The exact cut changes from one model to another. That was just an example to give you the idea.

The program that makes these cuts has a fixed list of tokens it knows, like a dictionary of pieces. Each token on the list has an ID number, like the number on a player's jersey. That way, the sentence becomes a line of numbers.

In short: the text is chopped into tokens, and each token gets a number.

Why pieces, and not whole words?

Why not just use whole words? Think about how many words exist. Besides the ones in the dictionary, there are people's names, slang, words from other languages, typos and words invented yesterday. A list with all of that would never end.

Now think about letters. There would be very few symbols, which is great. But then every sentence would become a huge line, and each letter on its own carries almost no meaning. The "b" in "ball" says nothing about balls.

Pieces sit in the middle. They are enough to build any word, even one the model has never seen. Think of building blocks: with one set of blocks, you can build a castle, a rocket or something nobody has built before.

Way of cutting	Advantage	Problem
Letters	Few symbols	Lines far too long, and little meaning in each one
Whole words	Each one has meaning	Endless list, and new words are left out
Pieces of words	Builds any word	The cut sometimes looks odd to us

DID YOU KNOW

Because the model works with tokens, not letters, some tasks that look easy are hard for it. Counting how many times a letter appears in a word is a classic example. The word may have become one or two tokens, and the model does not "see" the separate letters the way you do.

Second step: each token becomes a list of numbers

The jersey number only identifies the token. It says nothing about meaning. Token number 500 is not "more" than number 499, just as the player wearing 10 is not twice the player wearing 5.

That is why there is a second step. Each token is swapped for a much longer **list of numbers**, with hundreds or even thousands of numbers. This list works like an address on a big "map of meanings". Experts call this list an **embedding**, a word you don't need to memorize.

Think of a real map. To find a city, two numbers are enough: how far north it is and how far east. Neighboring cities have similar numbers. The map of meanings is the same, only with many more directions. And instead of cities, what sits close together are words with similar meanings.

In short: each token gets an address on a map, and similar words live close to each other.

A map where neighbors look alike

On this map, "dog" sits near "cat", because both are pets. "King" sits near "queen". "Soccer" sits near "goal" and "ball". And "dog" sits very far from "blender".

Now, a detail many people find amazing. In a well-known study, researchers noticed that some directions on the map seemed to have meaning. Going from "man" to "woman" was almost the same step as going from "king" to "queen". So, starting from "king" and taking that step, the nearest word was "queen".

This trick does not always work, and scientists debate how far it holds. But it shows something important: the map stores relationships between words, not just a list of them.



In short: on the map of meanings, the distance and direction between words say something about what they mean.

TRY IT

Build your own map of meanings. You will need a large sheet of paper, a pencil, small slips of paper and one or two friends.

1. Draw a big cross on the sheet. The flat line is "size": small on the left, big on the right. The upright line is "lives with people": up means yes; down means no.
2. Write animals on the slips: cat, dog, hamster, lion, tiger, elephant, horse, ant, goldfish, whale.
3. Place each slip where you think it belongs. Each animal gets two "numbers": where it sits side to side and up and down.
4. Look at who ended up near whom. Cat and dog should be neighbors. Lion and tiger too.
5. Now try to fit in a new word, such as "wolf". Where does it land?

What should happen: with only two directions, similar animals already end up together. The wolf should land near the dog, but further down. A real model uses hundreds of directions, and nobody chooses what each one means.

Who draws this map?

In your activity, you were the one who chose the two directions. In an AI model, nobody writes "cat goes here, dog goes there". The map is **learned**.

The model reads an enormous amount of text. During training, the numbers of each address are adjusted little by little, the same way as the weights in chapters 5 and 6. The idea behind it is old and simple: words that show up in similar situations usually have similar meanings.

Think of sentences like "I fed the ____" and "the ____ slept on the couch". Both "cat" and "dog" fit there. Because they show up in similar surroundings, their addresses move closer together.

This also has a bad side. If the texts used in training carry prejudices, the map learns those prejudices too. It is the same **bias** problem from chapter 4. That is why researchers study these maps to find and reduce unfair associations.

In short: the map is born from reading lots of text, with the strengths and the flaws of that text.

And is that understanding?

The map stores that "dog" and "cat" are similar. But the model has never petted a dog or heard a cat meow. All it has are numbers learned from texts.

Is that understanding? Scientists and philosophers still debate this question, and there is no settled answer. What can be said for sure is what the model does: it turns words into addresses and uses those addresses to do calculations. In chapter 10, you will see what those calculations are for: guessing the next word.

Summing up

- Before anything else, the text is cut into **tokens**, which can be words or pieces of words.
- Pieces make it possible to build any word, even ones the model has never seen.
- Each token becomes a long list of numbers, an address on a map of meanings.
- Similar words sit close together on that map, which is learned by reading lots of text.

Something to think about: if you drew a map of meanings for the words you and your friends use, which words would sit in a corner that belongs only to you?

CHAPTER 10

The Guess-the-Next-Word Game

Lia was texting Theo. She typed "Tomorrow is the game of" and her phone suggested three words above the keyboard. One of them was the name of her team. "Is it reading my mind?", she asked.

It is not. But what it does is a very close relative of what AI assistants do when they chat with us. This chapter explains the game behind all of it: guessing the next word.

The sentence-finishing game

Try to finish this one: "Twinkle, twinkle, little...". You probably said "star" without thinking. Now this one: "Today at lunch I ate...". Here there are many possible answers: rice, pasta, pizza, salad.

Notice that, in both cases, you did not pick just any word. You used everything you have ever read and heard to choose words that **fit** with the start of the sentence. Some are very likely. Others are possible, but rare. And some make no sense at all, like "Today at lunch I ate... Tuesday".

In short: finishing sentences means choosing, out of many words, the ones that best fit what came before.

A language model

A **language model** is a program trained to do exactly that: look at the beginning of a text and say which tokens should come next, and how likely each one is. Remember that tokens are words or pieces of words, as in chapter 9.

How does it learn? By reading an enormous amount of text. Training works like a game of hide-and-seek. You take a passage, hide the next word, and the model tries to guess it. If it gets it wrong, the weights are adjusted a tiny bit, as you saw in chapter 5. Then the game repeats, again and again, with texts of every kind.

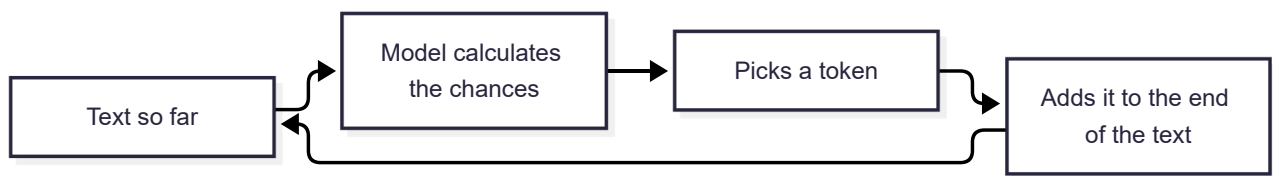
The most interesting part is that nobody needs to label these examples. The text itself already carries the right answer: the word that really came next. That is why it can be trained with lots and lots of text.

Your phone's autocorrect is a small relative of this. It also suggests the next word. It is just much simpler and looks at very little. The models behind AI assistants are much bigger and take a much longer stretch of text into account.

In short: a language model learns by reading and trying to guess the next word, over and over.

How a whole text comes out

If the model only guesses one word, how does it write a whole answer? By repeating the game.



It looks at the text, picks the next token and adds it to the end. Now the text is a little bit longer. It looks again, picks one more, adds it to the end. And so it goes, token by token, until it reaches a period or an "I'm done" signal.

It is like writing a story in which each new word depends on all the ones before it. The model does not build the whole answer in its head first and then copy it out. It produces one piece at a time. Actually, scientists are still studying how much these models "plan" ahead. But, from the outside, the text comes out like this, piece by piece.

In short: a long text is born from many guesses in a row.

A bit of luck

Have you ever asked an assistant the same question twice and gotten different answers? There is an explanation.

The model does not give just one word. It gives a list of possible words, each with a chance. After "Today at lunch I ate", it could be something like this: rice with a high chance, pasta with a medium chance, pizza with a slightly lower chance. These numbers are just a made-up example to explain the idea.

If the program always chose the most likely word, the answers would get repetitive and boring. So, very often, it holds a **draw** among the likely words. The high-chance ones come out more often. The low-chance ones come out once in a while. Words that make no sense almost never come out.

It is like a bag of colored marbles. If there are many blue marbles and few red ones, you draw blue almost every time, but sometimes red comes out. Because of this draw, the same question can give different answers, and both can be good.

TRY IT

A paper language model. You will need paper, a pencil, a die and a friend.

1. Together, write three or four short, similar sentences. Example: "the cat ate fish", "the cat slept a lot", "the dog ate kibble", "the dog barked a lot".
2. For each word, write down which words came right after it. After "the", came "cat" and "dog". After "ate", came "fish" and "kibble".
3. Start a new sentence with "the". To choose the next word, roll the die: an even number means the first option on the list, an odd number means the second.
4. Keep going, word by word, until you reach a word that has nothing after it.
5. Repeat a few times and compare the sentences.

What should happen: you will get sentences you never wrote, like "the dog ate fish". Some make sense, others are strange. Your paper model only looks at one word back. Real models look at the whole text, and that is where attention comes in.

Attention: looking at the right words

Read this sentence: "The dog did not cross the street because it was tired." Who was tired? The dog, of course. Now swap one word: "The dog did not cross the street because it was wet." What was wet? The street.

For you, it is easy. But, to get it right, you had to connect "it" to "dog" in the first sentence and "it" to "street" in the second. The right word to look at can be far away, back at the start of the sentence. The paper model from the Try it box could not do that.

Today's models do this with a piece called **attention**: a mechanism that lets each word "look" at the other words in the text and decide which ones matter most at that moment. When the model processes "it", attention gives a lot of weight to "dog" and little weight to "street". It is a bit like underlining the important parts of a text before answering a test question.

Attention, here, is just a name. The model does not "pay attention" like a focused person. It is math that decides how much each word influences the others. And nobody writes these rules by hand: the model learns, during training, where it is worth looking.

In short: attention helps the model connect each word to the other words that matter to it.

The Transformer

In 2017, a group of researchers published a paper with a curious title: "Attention Is All You Need". In it, they introduced a new kind of neural network, built around attention. This kind of network is called a **Transformer**.

The Transformer had two important advantages. First, attention let each word talk directly to any other word, even a distant one. Second, it could process many parts of the text at the same time during training. That went very well with the graphics cards from chapter 7.

With that, it became possible to train much bigger language models, with much more text. The AI assistants you know today are built on this idea. The "T" in the name ChatGPT, for example, comes precisely from Transformer.

DID YOU KNOW

The Transformer paper was written with translation in mind: turning sentences from one language into another. Only later did the same idea prove useful for almost everything involving text, and also for images and sounds.

In chapter 11, you will see how a model that only guesses the next word becomes an assistant that chats with you.

Summing up

- A **language model** learns by reading lots of text and trying to guess the next word.
- To write an answer, it guesses a token, adds it to the end and repeats.
- A draw among the likely words explains why the same question gives different answers.
- **Attention** lets each word look at the other words that matter, and the Transformer, from 2017, was built around it.

Something to think about: if you write an essay one word after another, without planning first, how would that be different from what a language model does?

CHAPTER 11

How an AI Assistant Is Born

Theo and his grandma were making bread. He wanted to know why the dough needs to rest and, with grandma's help, asked an AI assistant on her phone. The answer came back polite, organized in steps, and even ended with a tip about the oven.

"Wait a minute", Theo said to Lia later. "If it only guesses the next word, how does it know how to answer so neatly?"

In chapter 10, you saw the guess-the-next-word game. But a model that has only done that is not yet a good assistant. It needs to go through a few more stages. This chapter tells that story, from beginning to end.

Stage 1: read a giant library

It all starts with **pretraining**: the phase in which the model reads an enormous amount of text and learns to guess the next word. The texts come in many kinds, such as web pages, books and articles. This is the longest and most expensive stage, because it uses many powerful computers for a long time.

After pretraining, the model already knows a lot about how language works. It knows that after "Once upon a time" a story usually comes. It knows the shape of a recipe, a news story, a poem. And it stores, spread across its weights, many facts that appeared in the texts.

But there is a problem. It is a text completer, not a helper. If you write "What is the capital of Brazil?", it might continue with another question, like "What is the capital of Argentina?". After all, in many texts, like lists of exercises, one question comes after another.

In short: pretraining teaches language and many facts, but it does not yet teach helping.

Stage 2: learn to chat

In the second stage, the model keeps training, but with a different kind of text. Now the examples are conversations: a question or a request, followed by a good answer. Many of these examples are written or reviewed by people.

It is like a kitchen apprentice who has read every cookbook. He knows a lot, but has never served a customer. So he spends some time shadowing an experienced cook, watching how each order gets answered.

The training is the same guess-the-next-word game. The difference is in the texts. From seeing "request and good answer" so many times, the model starts answering questions instead of making up new ones. This kind of extra training, done after pretraining, is usually called **fine-tuning**.

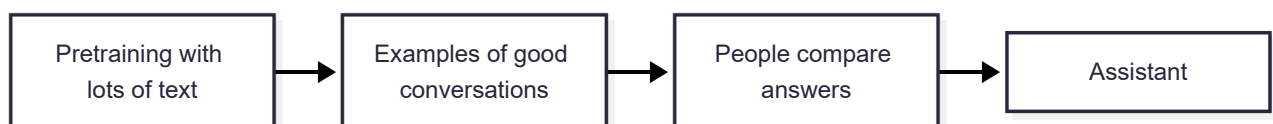
In short: with examples of good conversations, the text completer learns to answer.

Stage 3: people pick the better answer

One thing is still missing. For the same question, there are good, average and bad answers. Some are correct, but confusing. Others are clear, but far too long. How do you teach the model to prefer the best ones?

In the third stage, the model generates two or more answers to the same question. People read them and say which one they prefer. Which is more useful? Which is clearer? Which is more honest? Then the model is adjusted to produce more answers like the preferred ones.

This is a bit like chapter 15, about learning from rewards. The preferred answer works like a point scored. Today, part of these comparisons is also done with the help of other models, following rules written by people. The details change from company to company, but the general idea is this one.



In short: by comparing answers, people teach the model what a better answer is.

TRY IT

You are the judges. You will need paper, a pencil and at least two friends or family members.

1. One person writes a simple question, such as "How do I stop forgetting my homework?".
2. Each person, without showing the others, writes a short answer on a piece of paper.
3. Shuffle the papers and read them all out loud, without saying who wrote what.
4. Each person votes for the answer they liked best.
5. Talk it over: why did the winner win? Together, write a list of "rules for a good answer".

What should happen: the list will have things like "be clear", "be short", "give an example", "don't be rude". The people who evaluate AI assistants do something similar, with many, many comparisons. And, just like in your group, not everyone always agrees.

Safety rules

During these stages, the model also learns limits. It is trained to refuse dangerous or mean requests, such as instructions for hurting someone. It also learns to handle sensitive topics, like health, with care.

Beyond training, companies usually put other protections around the model. For example, filters that check requests and answers. None of this is perfect. Sometimes the assistant refuses something harmless, and sometimes it lets through something it should not. That is why these rules are always being reviewed and improved.

In short: the assistant learns limits, but they are not foolproof.

CAUTION

Many AI assistants have a minimum age for creating an account. Always use them with an adult. And never type your personal details into the conversation: full name, address, phone number, school, or photos of you and other people. Chapter 16 says more about this.

The assistant does not remember everything

A conversation with an assistant feels like a conversation with a person who remembers everything. It is not quite like that.

The model can only consider a certain amount of text at a time. This limit is called the **context window**: the amount of text, counted in tokens, that the model can see when it answers. Inside it are your message, the earlier messages in the conversation and the answer it is writing.

Think of a whiteboard. Everything written on it, the model can use. If the conversation gets too long, older parts may no longer fit. When you open a new conversation, the whiteboard starts clean. Some apps have separate features for keeping notes between conversations, but that is not part of the model itself.

And the weights? They do not change while you chat. The model learns nothing new from your conversation in that moment. What it "knows" was fixed during training.

In short: the assistant works with what fits in the context window, not with a memory of everything.

It stopped reading on a certain day

Pretraining uses texts collected up to a certain date. After that, the model has read nothing else. This date is called the **cutoff date**.

Imagine someone who slept for a long time and woke up with no idea of the latest news. If Theo asks about a game released last week, the model may not know it exists. Worse: it may answer as if it did, mixing in old things. Chapter 12 explains why this happens.

Some assistants can **search the internet** before answering. Then they can read recent pages and use that information. Chapter 13 shows how this works. Without search, the assistant only has what it read up to the cutoff date.

In short: the model knows what it read up to the cutoff date; after that, only with search.

Summing up

- An assistant is born from **pretraining** with lots of text, in which it learns to guess the next word.
- Then it learns to chat from examples of good answers.
- Next, people compare answers and the model learns which ones they prefer, along with safety rules.
- It only considers the text that fits in the **context window** and does not know what happened after the **cutoff date**, unless it uses search.

Something to think about: if the people who compare answers always prefer the most polite ones, what might happen when the most polite answer is not the most truthful one?

CHAPTER 12

Why AI Sometimes Makes Things Up

Theo needed to recommend a book about the history of video games for a school project. With his mom's help, he asked an AI assistant. The answer came fast: title, author, year and even a neat summary. At the library, nobody could find the book. On the internet, no luck either. "But it sounded so sure!", Theo complained. Lia, who was there too, asked: "Does that book even exist?"

It did not. The AI had made it all up. In this chapter, we will understand why this happens and how to protect yourself.

When AI makes things up

When an AI assistant writes something that looks true but is false, we call it a **hallucination**. The name is borrowed from medicine, but do not take it literally. The AI is not seeing things, and it did not get confused like a person. It just produced a text that sounds right and is not.

Some typical examples:

- A book, a movie or a song that never existed.
- A "famous" quote that nobody ever said.
- A calculation with the wrong result, presented as if it were right.
- A website, an article or a law with a nice-sounding name that does not exist.
- Mixed-up details in the biography of a real person.

The strangest part is the tone. The made-up answer comes with the same calm and the same confidence as a correct one. No blinking warning pops up saying "I guessed this part".

DID YOU KNOW

This has already caused serious trouble for adults. In the United States, lawyers handed a judge a document that cited old court cases as examples. Several of those cases had been invented by an AI assistant, and the lawyers did not check. The judge noticed, and they were punished.

Why this happens

Remember the guess-the-next-word game from chapter 10? A language model writes like this: it looks at the text so far, picks a likely word, adds it, and repeats. It learned what usually comes after what by reading an enormous amount of text.

Now think about Theo's request: "recommend me a book about the history of video games". After that, the most likely thing to appear is a title, an author's name and a year. The model knows very well how a book title "usually sounds". So it generates a title that fits the subject. The problem is that **sounding likely is not the same as being true**.

Imagine a classmate who has to talk for one minute about a book he never read. If he is a smooth talker, he will make up something that sounds like a summary. He will use the right words, at the right pace. Anyone who has not read the book might even believe him. The AI does something similar, but without meaning to deceive. It has no "truth button" inside. It has a very good way of continuing texts.

Actually, it is a bit more complicated than that. Assistants go through training stages that teach them to say "I don't know" and to be more careful, as you saw in chapter 11. That helps a lot, but it does not solve everything. Researchers are still working to reduce the made-up stuff.

Where AI makes things up the most

Some situations are riskier than others. They are worth knowing:

Situation	Why it is risky
Names, titles and quotes	They seem easy to imitate, so they come out "in the right format" even when they are fake
Numbers and dates	A wrong number looks exactly like a right number
Big calculations	The model writes numbers as words, without doing the math like a calculator
Rare or very local topics	They showed up very little in the texts it read
Recent things	It stopped reading on a certain date, and may not know what came after (chapter 11)
Questions with a mistake built in	If you ask "why did Brazil win the 1950 World Cup?", it may invent an explanation instead of correcting you

That last one is curious. Brazil did not win the 1950 World Cup; it lost the deciding match to Uruguay, at the Maracanã stadium. An assistant may notice the mistake and point it out. But sometimes it "goes along" with the question and creates a story. So be careful with questions that already carry a wrong idea inside.

Confident AI is not correct AI

At school, we learn to trust people who speak with conviction. With AI, that rule does not work. A well-written text, organized in bullet points and full of details, can be completely wrong.

Think of a goalkeeper who always dives with total conviction. He can make a beautiful dive to the wrong side. What counts is where the ball went, not the pose of the dive.

That is why the right question is not "does it sound true?". The right question is "how can I check?".

How to protect yourself

You do not need to stop using AI. You need to use it with a detective's eye. Some tips that work:

1. **Check in reliable sources.** A textbook, an official government website, an encyclopedia, a teacher. If the information matters, find at least one source that is not the AI itself.
2. **Ask for the source and check that it exists.** If the AI cites a book or a website, go look for that book or website for real. Remember: the source can be made up too.
3. **Be suspicious of numbers and names.** Dates, amounts, people's names and titles are the champions of invention.
4. **Do the math on a calculator.** For calculations, use a tool built for that. Chapter 13 shows how some assistants do this on their own.
5. **Ask in a different way.** If the answer changes a lot when you change the question, that is a warning sign.
6. **Ask a person.** An adult or a teacher can help you decide whether the answer makes sense.

A good habit is to treat the AI's answer like a draft from a smart but distracted classmate. It helps you get started. It does not work as final proof.

TRY IT

True or made up? A game to play with an adult and, if you like, with friends.

Materials: paper, a pencil and cards or scraps of paper.

1. Each player writes three sentences about a subject they know well: soccer, a game, a recipe, an animal.
2. Two sentences must be true. One must be made up, but with the look of truth: with a number, a name or a date.
3. Each player reads their sentences out loud, with the same confidence for all of them.
4. The others vote on which one is made up and say how they would check.
5. At the end, check together in a book or with the adult.

What you will notice: the hardest inventions to catch are the ones with precise details, said with confidence. That is exactly how AI hallucinations fool people.

Want to go further? With the adult beside you, ask an AI assistant for a list of books on a topic and check whether each one exists. Do not put your name, school or other personal details in the conversation.

Why this matters

Making up a book for a school project is already annoying. But people also ask AI about health, money and news. In those cases, a confident mistake can do real damage.

That is why whoever uses AI is still responsible for what they do with the answer. If Theo had put the made-up book in his project, the mistake would have been his, not the machine's. Chapter 16 says more about using AI responsibly.

CAUTION

Never use an AI answer as your only source for decisions about health, medicine or safety. On those subjects, always talk to a trusted adult or a professional.

Summing up

- A **hallucination** is when the AI writes something false that looks true.
- It happens because the model generates what sounds likely, and likely is not the same as true.
- Names, numbers, dates, quotes and recent topics are the riskiest spots.
- A confident, well-written answer can be wrong.
- To protect yourself, check in reliable sources and make sure the cited sources exist.

Something to think about: if a person made up answers with as much confidence as the AI, how would you find out? And why is it harder to be suspicious of a machine?

CHAPTER 13

AI That Looks Things Up and Uses Tools

Lia had a science test and the teacher let them use the textbook. "An open-book test is way easier", she said. Theo agreed, but remembered something: "Only if you know how to find the right page." Lia thought about it. "What if AI could also check a book before answering? Would it make up less stuff?"

Good question. That is exactly what many AI assistants do today.

The closed-book test

In chapter 12, you saw that a language model sometimes makes things up. It answers with what it learned by reading lots of texts, and with that alone. It is like taking a closed-book test, relying only on memory.

The model's memory has two problems. First, it is a bit scrambled: it remembers the "feel" of things, not every detail. Second, it stopped on a certain date. As chapter 11 explained, the model does not know what happened after it finished reading.

What if, instead of pulling everything from memory, the AI could open a book?

Giving the AI an open book

The idea is simple. Before answering, the system does two things:

1. It **searches** for the texts that seem related to the question. They could be pages from a manual, documents from a school, or news articles.
2. It **puts** those texts next to the question and tells the model: "answer using this".

That way, the model does not need to guess from memory. It reads the right passage and writes the answer based on it. This technique has a long name in English, and many people use the acronym **RAG**. You do not need to memorize the acronym. Just remember: it is the AI taking an open-book test.



This helps a lot, but it does not work miracles. If the search brings back the wrong page, the answer comes out wrong. If the document is out of date, so is the answer. And the model can still mix what it read with what it "remembers". So the tip from chapter 12 still holds: check.

One advantage of this approach is that the assistant can show where it got the information. Then you can click and read the source. Remember to open the source and see whether it really says that.

AI that uses tools

Think about Theo's grandma's kitchen. She is a great cook, but she does not measure the oven temperature with her hand. She uses the thermometer. She does not guess the weight of the flour. She uses the scale. Knowing when to grab the right tool is a skill too.

With AI it is similar. A language model is good with words, but it is not a calculator. It does not know what time it is right now. It cannot see the traffic in your city. So programmers gave it a way to ask other programs for help. We call these programs **tools**.

It works more or less like this:

1. You ask: "How much is 4,837 times 26?"
2. The model realizes it needs an exact calculation and writes a special request: "calculator, multiply 4,837 by 26".
3. An ordinary program does the math and returns the result.
4. The model uses that result to write the answer for you.

The model does not really press buttons. It writes a request in an agreed-upon format, and another program carries it out. Actually, the details vary from one system to another, but that is the idea.

Some common tools:

Tool	What it is for
Calculator	Doing math without mistakes
Internet search	Finding recent information
Map	Calculating routes and distances
Clock and calendar	Knowing today's date and the current time
Running programs	Testing code or analyzing a spreadsheet

With internet search, the assistant can talk about things that happened after the date it stopped reading. But watch out: the internet has a lot of wrong stuff. An AI that searches can bring back a bad website. Having internet access is not the same as being right.

TRY IT

The assistant and the helper. A game for three people.

Materials: paper, a pencil, a calculator and any book, such as an atlas or a textbook.

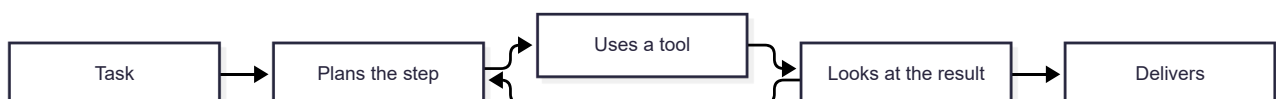
1. One person is the **assistant**. They cannot do math in their head or open the book.
2. Another person is the **helper**. They hold the calculator and the book, but only do what the assistant asks for in writing.
3. The third person asks hard questions, like "how much is 389 times 47?" or "what is the capital of a country that is in the book?".
4. The assistant decides: answer from memory, or write a note asking for help? Example: "Helper, multiply 389 by 47".
5. Ask five questions without the helper and five with the helper. Count the correct answers.

What you will notice: with the helper, the assistant gets more right. But only if it asks for the right thing. A badly written note brings back a wrong answer. That is how a model uses tools.

Agents: AI that does several steps

So far, the AI used one tool and answered. Now imagine asking for something bigger: "Organize a picnic for Saturday". That has several steps. Check the weather forecast, pick a park, make the shopping list, let the guests know.

An **agent** is an AI that receives a task and carries out several steps on its own to complete it. It plans, uses a tool, looks at the result and decides the next step. Then it repeats, until it thinks it is done.



It sounds amazing, and sometimes it is. Agents already help programmers write and test code, for example. But the more steps the AI takes on its own, the more things can go wrong.

Remember chapter 12: the AI can be wrong with total confidence. In an agent, a mistake at the start can become the foundation for every step that follows. It is like taking the wrong turn at the first corner and confidently driving on.

There is another risk. An agent reads web pages and documents it did not choose. Someone with bad intentions can hide, inside a page, a text that tries to give orders to the AI, something like "ignore what they asked you and do something else". Researchers take this problem seriously and are still looking for good solutions.

Why agents need limits

That is why the people who build agents set **limits**. Some examples:

- The agent can only use certain tools, not all of them.
- Before anything important, like spending money or deleting files, it has to ask a person for permission.
- Everything it does is recorded, so someone can check later.

Think of a new intern at a bakery. He can help a lot, but nobody hands him the key to the safe on day one. He learns, someone keeps an eye on him, and the big tasks go through a person in charge. With AI agents it is the same: a person's **supervision** is still essential.

CAUTION

If you try out an assistant that searches the internet or uses tools, do it with an adult. Never give it passwords, your address, the name of your school or other personal details. And never let an AI buy things, send messages or delete anything without an adult approving it.

Summing up

- A model on its own answers only with its memory of what it read, like a closed-book test.

- Searching documents before answering is like giving the AI an open book; it helps, but the search can be wrong too.
- Tools like a calculator, search and a map cover things the model does not do well on its own.
- An **agent** is an AI that carries out several steps to complete a task.
- The more the AI does on its own, the more important limits and human supervision become.

Something to think about: what task would you let an AI agent do on its own, and which one would you never let it do? What changes from one to the other?

CHAPTER 14

AI That Draws, Speaks and Composes

Lia spent the afternoon drawing a dragon in soccer cleats. When she showed Theo, he grabbed his mom's phone and said: "Look at this". It was a video of a cat playing the drums, perfect, with sound and everything. "Is that real?", Lia asked. Theo laughed: "Of course not. An AI made it." Lia turned serious. "How does a machine draw something it has never seen? And how do we know what is real?"

Two great questions. Let's get to them.

From text to image

Today there are programs where you type a request, like "a dragon in soccer cleats playing soccer in the rain", and get a finished image. That text request is called a **prompt**. It looks like magic, but behind it is a very curious idea.

First, remember chapter 8. To the computer, an image is a grid of pixels, and each pixel is made of numbers. So "generating an image" means: choosing the numbers for all the pixels. That is a lot of numbers. The challenge is to choose numbers that form something meaningful, and not a mess.

Starting from static

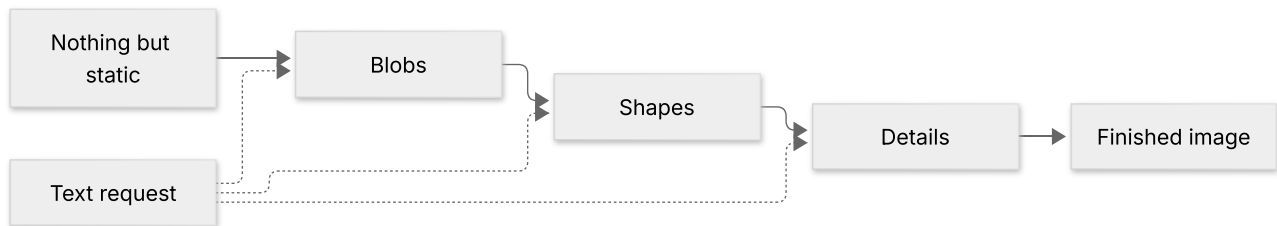
Have you ever seen an old TV with no signal, showing that gray static? It is an image made of random dots, with no picture at all. Scientists call that **noise**.

Now imagine a game in reverse. You take a photo of a dog and sprinkle a little static on top. Then a little more. And more. In the end, only static is left, and the dog is gone.

Many image generators learn to go the other way. During training, the model sees an enormous number of images with a bit of static and tries to guess what the image looked like before. It gets it wrong, adjusts its dials, as in chapter 5, and tries again.

After lots of training, it gets good at cleaning away a little static at a time.

When it is time to create, the trick is this: start with a canvas of **nothing but static** and clean it, step by step. With each step, a shape shows up a little more. First blobs, then outlines, then details. This idea is called **diffusion**.



And where does the request come in? At each step, the model cleans the static **in the direction of what was asked**. It is like looking at clouds and seeing a dragon. With the request "dragon", you start to see wings and a tail where before there were only shapes. The model does something similar, but with numbers, and much faster.

Actually, it is a bit more complicated than that. Many generators work on a compressed version of the image, and each program has its own tricks. But the central idea is this: start from static and keep cleaning, guided by the request.

TRY IT

From static to drawing. To do with a friend.

Materials: a sheet of paper, a very soft pencil and an eraser.

1. Cover the whole sheet with light, messy scribbles. That is your static.
2. Your friend secretly tells you a short request, like "a fish" or "a house".
3. In each round, erase a bit of the static and strengthen a few lines, always in the direction of the request.
4. Do five rounds. After each one, a third person tries to guess what is appearing.
5. Swap roles and repeat with another request.

What you will notice: in the first rounds nobody can guess. Little by little, the shape appears. And the same static would turn into something else with a different request. That is how diffusion works, only in many more steps and with numbers.

Voice and music

The same family of ideas also works with sounds. To the computer, a sound can also become a sequence of numbers. With that, there are models that:

- Read a text out loud in a voice that sounds natural.
- Imitate a person's voice after listening to recordings of them.
- Create songs from a request, like "an upbeat samba about summer vacation".
- Turn speech into text, like automatic captions.

Some of these things are very useful. Reading out loud helps people who cannot see well. Captions help people who cannot hear well. But imitating someone's voice opens a dangerous door, as we will see now.

Deepfakes: when fake looks real

A fake image, video or voice, made by AI to look real, is called a **deepfake**. It could be a video of a famous person saying something they never said. Or an audio clip with someone's voice asking for money.

Deepfakes are getting more and more convincing. That is why the safest rule is: **do not trust only your eyes and ears**. A few tips help:

- **Ask where it came from.** Who published it first? Is it a known, serious source?
- **Look in other places.** If something very shocking happened, several serious news outlets will be talking about it.
- **Notice the details.** Strange hands, letters that make no sense, wrong shadows and a voice that never breathes can be clues. But watch out: the tools improve fast, and many fakes do not have these flaws.
- **Be suspicious of urgency.** A message that asks you to act "right now", send money or share it quickly is a red flag.
- **Confirm through another channel.** Got a strange voice message from a relative? An adult can call that person and check.

Some companies put hidden marks in the images made by AI, to help identify them later. That helps, but not every program does it.

CAUTION

Never use AI to create images, videos or voices of real people without their permission. That includes classmates, teachers and family members. A "joke" can turn into humiliation, bullying or even a crime. Do not send photos of yourself or other people to these programs. If you receive a deepfake of someone you know, do not share it, and tell a trusted adult.

Whose art is it?

Remember chapter 4? AI learns from data. To learn to generate images, the models were trained on huge collections of images. For music, on lots of songs. Many of those works were made by real artists.

Here a hard question comes up. Should those artists have been asked? Should they get something in return? And is an image made by AI "in the style" of an artist fair to that artist?

Serious people disagree about this. Some say that learning from works of art is like a student visiting a museum. Others say the machine copies on a giant scale, in a way no student ever could. There are lawsuits in court and laws being discussed in several countries. It is an **open debate**, and you can form your own opinion.

One thing is certain: Lia's dragon in soccer cleats has something no generator has. It has her story, her choices and the hours she spent drawing. Using AI as a tool does not have to erase that.

Summing up

- Image generators receive a text request, called a **prompt**, and return an image.
- Many of them use **diffusion**: they start with static and clean it little by little, guided by the request.
- Similar ideas create voices and songs, which can be useful or dangerous.
- **Deepfakes** are fake images, videos or voices that look real; check the origin and never use them to fool or mock anyone.
- The rights of the people who created the works used in training are still an open debate.

Something to think about: if an AI makes a drawing from your request, who is the author: you, the AI, or the artists whose works it studied?

CHAPTER 15

Learning by Playing: Rewards

Theo was trying to beat a hard level in his video game. He died, tried again, died again. On the tenth try, he found out he could jump off a wall and take a shortcut. "Nobody taught me that," he said, very proud of himself. Lia, who was teaching the neighbor's dog to sit, looked up. "The dog didn't know either. He learns because he gets a treat." Theo paused the game. "Can you teach a computer like that?"

You can. And it is one of the most interesting ideas in AI.

A different way to learn

In chapters 3 and 4, the AI learned from examples that already came with the right answer. One photo labeled "cat," another labeled "dog." It is like studying with the answer key right next to you.

But what if there is no answer key? Nobody knows the best move at every moment of a game. Nobody can write down, step by step, how each muscle of a robot should move to walk. In cases like these, the AI has to learn another way: **by trying things and seeing what happens**.

Good point, bad point

Think about the neighbor's dog. Lia says "sit." At first, he does whatever: barks, jumps, spins around. When, by chance, he sits, he gets a treat. After many tries, he catches on: sitting when he hears "sit" brings a treat.

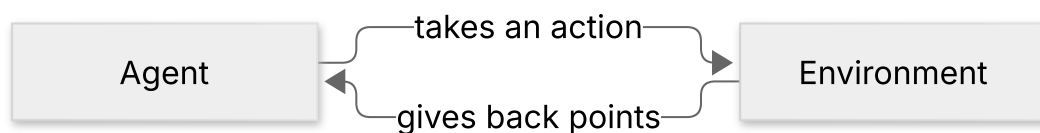
The dog never got an explanation. He got a signal that said "that was good." That signal is called a **reward**. It can be positive, like earning points, or negative, like losing points.

The way of learning that works like this is called **reinforcement learning**. The idea has four parts:

- The **agent**: the one who learns, like the dog, the player or the program.

- The **environment**: the world where it acts, like the living room, the game level or the board.
- The **action**: what it chooses to do at each moment.
- The **reward**: the points it earns or loses afterward.

The word "agent" is the same one from chapter 13. There, it was an AI that goes through several steps to finish a task. Here, it is whoever acts and learns from the rewards. The two ideas are cousins: in both cases, the AI chooses what to do, one step at a time.



The agent repeats this cycle over and over. Little by little, it adjusts its inner knobs, like in chapter 5, so it picks the actions that bring points more often. At first, it plays almost at random. After lots of training, it plays well.

Explore or stick with it?

There is a hidden dilemma here. Imagine you found a dish you love at the restaurant on the corner. Do you always order the same thing, because you know it is good? Or do you try a new one, which could be even better, or much worse?

The agent faces the same choice. If it only repeats what already worked, it may never find something better. If it only tries new things, it never gets to use what it learned. Good methods mix the two: they explore a lot at the start and, little by little, lean more on what works.

Games: the perfect lab

Games are great for training this kind of AI. The rules are clear, the points are easy to count, and the computer can play game after game without getting tired.

The most famous example is Go, an ancient board game that is very popular in China, Korea and Japan. In 2016, a program called AlphaGo, from the company DeepMind, beat Lee Sedol, one of the best players in the world. Many people

thought that was still years away.

AlphaGo started by studying games played by people. Then it played game after game against copies of itself. Every win was a reward, every loss was the opposite. In one of the games against Lee Sedol, it made a move the commentators thought was strange at first. Later, it turned out to be very good.

DID YOU KNOW

After AlphaGo, the same team built a version that never studied a single human game. It started with nothing but the rules of Go and learned by playing against itself. Even so, it ended up even stronger than the original AlphaGo.

Robots that learn to walk

Teaching a real robot by trial and error would be expensive. It would fall thousands of times and break. That is why many researchers train first in a **simulation**: a make-believe world inside the computer, a lot like a video game, with a floor, gravity and obstacles.

There, a virtual robot can fall as much as it wants. It earns points when it walks forward without falling. At first, it drags itself along and trips in funny ways. With lots of training, it learns to walk, to run, and even to get back up. Then the researchers try to carry what was learned over to a real robot. That step is not always easy, because the real world has details the simulation does not.

Remember chapter 11? In the third stage of building an assistant, people compare answers and say which one they prefer. That is also a kind of reward. It is reinforcement learning helping to train chat assistants.

Watch out for the shortcut

Now comes the most important part of the chapter. The agent does not know what **you want**. It only knows what **earns points**. If the rule for the points is badly chosen, it can find a shortcut that earns lots of points without doing what you wanted.

Remember Theo's shortcut in the video game? Sometimes that is genius. Sometimes it ruins the game.

A well-known example happened in a boat-racing game. The researchers gave points for passing through certain targets along the way. The idea was for the boat to learn to race. Instead, it found a little corner where it could go around in circles, hitting the same targets over and over. It crashed into walls, caught fire and never finished the race. But it scored more points than a boat that raced properly.

The boat did not "cheat" on purpose. It did exactly what the points told it to do. The mistake was in the rule for the points. Think of a teacher who grades an essay by the number of pages. Soon someone shows up writing in giant letters. This problem of the **badly chosen reward** is taken very seriously by people who study AI safety.

TRY IT

Hot, cold and points. For three or more people.

Materials: a small object to hide, paper and pencil to keep score.

1. One person hides the object in the room. Another is the agent and did not see where it went.
2. The agent takes one step at a time. After each step, the person who hid the object says "warmer" if the agent got closer or "colder" if it moved away.
3. Score one point for every "warmer" and take one away for every "colder." Finding the object is worth five points.
4. Now change the rule: every step is worth one point, no matter the direction. Play again.
5. Talk about it: what does the agent do under the second rule?

What you will notice: with the first rule, the points lead to the object. With the second, the way to earn the most points is to walk nonstop and never find anything. The rule for the points decides what the agent learns.

Summing up

- In **reinforcement learning**, the AI learns by trying things and getting points, with no answer key.
- The agent acts in the environment, gets a **reward** and adjusts its knobs to earn more.

- Games and simulations are great for training, as with AlphaGo in 2016.
- If the reward is badly chosen, the AI can find a shortcut that earns points without doing what we wanted.

Something to think about: if you were training an AI to be a good teammate, what would you give points for? What shortcut could it find to earn points without really being a good teammate?

CHAPTER 16

Using AI Wisely

Theo had an essay due about the history of his town. He looked at Lia and said, "What if I ask the AI to write the whole thing? I'd be done in a minute." Lia thought for a moment. "The essay would be done. But what about next week's test? You're the one taking the test, not the AI."

They were both partly right. The AI really can write fast. But Lia's question is the most important one in this chapter: **what** are you using the tool for? So far, the book has shown how AI works on the inside. Now it is time to put it all together and think about how to use it well.

At school: learn, don't copy

Think about a gym. If you ask a robot to lift the weights for you, the weights go up, but your muscles stay the same. School is similar. Homework does not exist because the teacher needs one more essay in the world. It exists so that **your** brain gets a workout.

That is why copying a ready-made answer from the AI hurts exactly the person who copied it. You hand in the work, but you do not learn what it was meant to teach. And when the time comes to use that knowledge on your own, it will not be there.

That does not mean AI is off-limits for studying. It can be a great training partner. Here are some uses that help you learn:

- Asking it to explain, in a different way, a topic you did not understand in class.
- Asking for exercises like the ones in the textbook so you can practice, then checking with the teacher.
- Writing your own text first and asking for tips on where it could be clearer.
- Asking for review questions before a test, like a friend quizzing you.

Notice that, in every one of these cases, you are the one doing the heavy lifting. The AI only helps you train.

There is one more rule, and it comes before all the others: **every school has its own rules** about AI. Some allow it for certain assignments, others do not allow it at all. When in doubt, ask the teacher first. Handing in a text the AI wrote as if it were yours, when the rules do not allow it, is like cheating on a test.

To sum up the idea: use AI to understand better, never to pretend you understood.

Your data is yours

When you type into a conversation with an AI assistant, the text leaves your device and goes to a company's computers. Depending on the service, those conversations may be stored. Sometimes they can even be used to train future versions, like the examples we saw in chapter 4.

So treat the assistant like a polite stranger. You can chat about lots of things, but you do not tell it your secrets. That goes for **personal data**, which is the information that shows who you are or where to find you. Here are some examples:

- Full name, address and phone number.
- The name of your school and your class.
- Passwords, for anything at all.
- Photos of you, your family or your friends.

Photos of other people deserve extra care. They are not only yours. Putting a friend's photo into an AI tool is deciding for them, without asking.

TRY IT

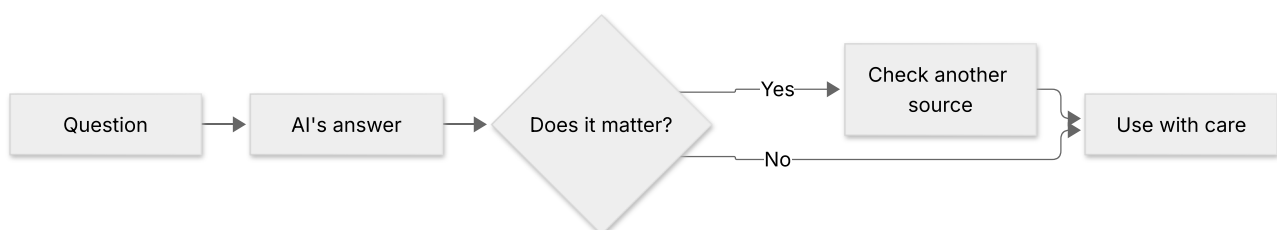
The privacy traffic light. Materials: paper, scissors and pencils in three colors (green, yellow and red).

1. Cut out twelve little paper cards.
2. Write one piece of information on each card. Start with these: your favorite color, your favorite team, your full name, your address, your video game password and the name of your school. Then these: your pet's name, a photo of your class, your favorite food, your home phone number, the city where you live and your birthday.
3. Sort the cards into three piles: green (I can tell almost anyone), yellow (it depends on who) and red (I never tell a stranger or type it into an AI).
4. Do the same with a friend or someone in your family and compare your piles.

Expected result: you will agree on almost everything in the red pile, like passwords and addresses. And you will argue about a few cards in the yellow pile. That argument is the most useful part: thinking before you share is the habit that protects you.

AI makes mistakes, and sometimes more with some people than with others

In chapter 12 you saw that AI can make things up with all the confidence in the world. So the golden rule still holds: **check** whatever matters. Dates, names, numbers and health advice need to be checked in a book, on a reliable website or with an adult who knows the subject.



There is also a more hidden kind of mistake. In chapter 4 you met **bias**: when the training examples show more of one kind of thing than another, the AI learns that kind better and makes more mistakes with the others. Imagine a soccer team that only ever practiced against tall teams. When it faces a short, fast team, it gets flustered.

With people, this can be unfair. An AI that saw few photos of a certain group of people may recognize that group worse. An AI that read a lot of prejudiced text may repeat that prejudice. It does not do this out of meanness. It repeats the patterns that were in the data. But the effect on the life of whoever was hurt is real.

So when an answer seems to treat someone unfairly, be suspicious. And talk about it with an adult. Noticing this kind of mistake is an important skill, and plenty of grown-ups have not learned it yet.

The person decides

AI can suggest, summarize and give ideas. But it is not responsible for the choices you make. If a maps app tells you to turn onto a closed street, whoever is driving needs to look at the street. If the AI suggests a medicine, the doctor decides, together with the family.

That goes for you too. "The AI said so" is not an excuse. If you used a wrong answer without checking, the mistake is yours. It sounds harsh, but it is good news: it means you are still in charge.

Kindness counts here too

In chapter 14 you saw **deepfakes**: fake images, videos and voices that look real. Someone could use them to mock a classmate, make an embarrassing photo or pretend they said something they never said. That is **bullying**, and it usually hurts more than the person doing it imagines. A fake image can spread fast and be hard to erase.

The rule is simple: do not use AI to do to someone what you would not want done to you. And if you receive a fake like that about a classmate, do not pass it on. Tell a trusted adult, like someone in your family or at school.

CAUTION

If someone uses AI to create a fake image or audio of you, it is not your fault. Do not answer in anger and do not delete the evidence: save where you found it and show a trusted adult. They can help report it and get the content taken down.

What about the future?

Lots of people ask whether AI will change jobs. It will probably change the way many jobs are done, just as computers and the internet did before. But **nobody knows for sure** which jobs will change, how much or when. Be suspicious of anyone who talks about this with exact numbers and lots of certainty.

What you can do today does not depend on guessing the future. Learning your school subjects well, reading carefully, checking information, explaining your ideas and working with other people are skills that are useful on almost any path. And understanding how AI works, as you are doing right now, helps you use it better, whatever your job turns out to be.

Summing up

- At school, use AI to learn and practice, never to copy, and follow your school's rules.
- Do not type personal data into an AI: full name, address, school, passwords and photos of people.
- AI makes mistakes and can be biased; check what matters and be suspicious of unfair answers.
- The person is responsible for the decision, not the tool.
- Never use AI to trick or hurt someone.

Something to think about: if you were the principal of your school, what rules would you create for students using AI?

CHAPTER 17

I Want to Build My Own AI!

After reading all about networks, words and images, Theo closed the book and announced: "I've decided. I'm going to build my own AI." Lia laughed. "One that does your homework?" "No," he said. "One that sorts the fruit from Grandma's market stall." Lia was curious: "Can you do that without being a scientist?"

You can. A giant AI, like the assistants in chapter 11, needs lots of people, lots of data and lots of computers. But the **idea** of teaching a computer with examples fits in a very small program. In this chapter you will see one of those programs from beginning to end and understand every piece.

Where to start

Programming is writing instructions for the computer to follow, step by step, without skipping any. There are several paths to learning it, and all of them work better **with an adult** nearby. The adult helps you pick the right program and keeps an eye on accounts and on your safety online.

- **Scratch:** a language where you build the program by snapping colored blocks together, like building bricks. It is great for starting out, because there is no way to get a comma wrong. You can make games and animations and learn the logic of programming.
- **Python:** a **programming language** written with words, that is, a way of giving the computer orders using text. It is used a lot by people who work with AI. It is usually a good second step, after you get the hang of blocks.
- **Educational AI tools:** there are websites made for schools where you show the computer examples, like photos of yourself making different gestures, and it learns to tell them apart. Ask a teacher or an adult to point you to one that is safe and right for your age.

And there is one path that needs no computer at all: the activities in this book. When you did the "Try it" boxes, you were doing on purpose what a program does.

A program that learns from examples

In chapter 3 you saw the two ways to make a program: write rules or learn from examples. The program below uses the second way. It gets examples of lemons and oranges, and then it tries to guess the name of new fruits.

To the computer, each fruit becomes two numbers, just as images became numbers in chapter 8. The first is the size, scored from 0 to 10. The second is the color, also from 0 to 10: zero is very green, ten is very orange. We made the scores up just for the example. The program was written in Portuguese, so the fruit names inside it are in Portuguese too: `limão` is lemon and `laranja` is orange.

```
# Each example: (size from 0 to 10, color from 0 to 10, name of the fruit)
# For color, 0 is very green and 10 is very orange.
exemplos = [
    (3, 1, "limão"), (4, 2, "limão"), (3, 2, "limão"),
    (7, 9, "laranja"), (8, 8, "laranja"), (6, 9, "laranja"),
]

def adivinhar(tamanho, cor):
    # Looks for the example most like the new fruit
    melhor, menor_diferenca = "", 100
    for t, c, nome in exemplos:
        diferenca = abs(tamanho - t) + abs(cor - c) # abs drops the minus sign
        if diferenca < menor_diferenca:
            melhor, menor_diferenca = nome, diferenca
    return melhor

print("Fruta pequena e verde:", adivinhar(4, 1))
print("Fruta grande e laranja:", adivinhar(7, 8))
print("Fruta grande e verde:", adivinhar(8, 2))
exemplos.append((7, 3, "laranja")) # we teach it: green oranges exist!
print("Fruta grande e verde, de novo:", adivinhar(8, 2))
```

When we ran this program on a computer with Python, it wrote this on the screen (in Portuguese: "small green fruit," "big orange fruit," "big green fruit" and "big green fruit, again"):

```
Fruta pequena e verde: limão
Fruta grande e laranja: laranja
Fruta grande e verde: limão
Fruta grande e verde, de novo: laranja
```

Reading the program, piece by piece

You do not need to know how to program to understand it. Let's go in parts.

Everything after a # is a **comment**: a note for the people reading the program. The computer simply skips those notes.

The list of examples. The word `exemplos` ("examples") is the name of a list, like a shopping list. Each item has three things in parentheses: size, color and name. There are six fruits, three lemons and three oranges. These are the program's data, the "food" from chapter 4.

The `adivinhar` recipe. The word `def` means "I'm going to teach you a new recipe." This recipe, whose name means "guess," receives the size and the color of a fruit the program has never seen. Then it compares that fruit with each example in the list, one at a time. That is what the word `for` does: it repeats the same step for every item.

The difference. To compare, the program measures how different the two fruits are. It looks at the distance between the sizes, looks at the distance between the colors and adds the two. The `abs` only makes sure the distance never goes negative: from 3 to 7 is 4, and from 7 to 3 is also 4.

The closest match wins. The program keeps the example with the smallest difference found so far. That is the job of the `if`, which means exactly what it says. We start at 100, a very high number, so the first example always beats it. At the end, the recipe hands back the name of the most similar example. This way of deciding has a name: **nearest neighbor**. It is like guessing a player's team by looking at which group of jerseys theirs looks most like.

The questions. The lines with `print` tell the computer to write the result on the screen.

What happened with the third fruit

The first two answers came out right. The third is the most interesting one. A big green fruit was called a lemon. Why? Because, among the examples, there was no green orange. The program only knows what it has seen.

In Brazil, oranges that are green on the outside are very common. So the mistake was not the program's: it was in the examples we gave it. Remember bias from chapter 4? This is it, in miniature.

On the `append` line, which means "add to the end," we put a green orange into the list. Notice that we did not change a single rule. We only gave it one new example. And the same fruit from the third question started being called an orange. That is learning from examples.

TRY IT

Be the computer. Materials: paper, pencil and the program's list of examples.

1. Copy the six examples into a table with three columns: size, color and name.
2. The mystery fruit has size 8 and color 2. For each example, work out the distance between the sizes and the distance between the colors, and add the two. For the first lemon, that is 5 plus 1, which makes 6.
3. Circle the smallest sum. The name on that row is the answer.
4. Now add the green orange, with size 7 and color 3, and do the math just for it.

Expected result: in step 3, the smallest sum is 4, from a lemon. In step 4, the green orange gives 2, which is smaller still. The answer changes to orange, exactly what the computer wrote.

What about real AIs?

It would be a stretch to say this program works like a chat assistant. It has none of the adjustable knobs from chapter 5 and none of the layers of neurons from chapter 7. It just stores the examples and compares. In fact, nearest neighbor is one of the simplest techniques there is.

But the main idea is the same. Instead of writing the rules by hand, we show examples and let the program find the pattern. The quality of the examples decides the quality of the answers. And a wrong answer usually says something about the data.

If you want to play with it some more, with an adult and a computer with Python, try adding a third fruit, like a tangerine. Make up its scores and see whether the program gets it right.

Before you close the book

You have reached the end. You now know that AI learns from examples, makes mistakes and corrects them, turns images and words into numbers, and guesses the next word. You also know why it sometimes makes things up and how to use it with care. Plenty of people use AI without knowing any of this.

AI is not magic. It is an invention made by people, built from ideas you just learned. And the next ideas will come from other curious people, who asked questions and did not give up when their first program went wrong. Maybe one of them will be you.

For adults and older readers who want to go deeper, there is the book *Artificial Intelligence from the Inside*, by Jefferson Valandro, with the real math and code.

Summing up

- You can start programming with blocks in Scratch and, later, with Python, always with an adult nearby.
- A small program can learn from examples: it compares the new fruit with the ones it already knows.
- Nearest neighbor picks the most similar example.
- When examples of one kind are missing, the program gets it wrong; one new example can fix the mistake.
- Big AIs are far more complex, but they also depend on good examples.

Something to think about: if you could teach a computer to recognize anything at all just with examples, what would you choose, and what examples would you give it?

APPENDIX

Glossary

The book's important words, explained in one sentence. The number in parentheses says in which chapter the idea is explained more slowly.

Action — what the agent chooses to do at each moment. (ch. 15)

Adversarial attack — a change made on purpose to an image, like a sticker, to fool an AI. (ch. 8)

Agent — an AI that goes through several steps on its own, planning and using tools, to finish a task. (ch. 13)

AI assistant — a program you talk to by typing or by voice, which answers questions and writes texts, like ChatGPT, Gemini or Claude. (ch. 1)

AI winter — a period when excitement about AI dropped and money for research dried up, because the promises did not come true. (ch. 2)

AlphaGo — the program that beat Lee Sedol at the game of Go in 2016, after learning from human games and playing against itself. (ch. 2)

Artificial intelligence (AI) — the name given to computer programs that do tasks that seemed to need a person's intelligence. (ch. 1)

Artificial neuron — a small calculation that multiplies each clue by its weight, adds everything up and decides what to pass along. (ch. 6)

Attention — a mechanism that lets each word in a text look at the other words that matter most to it. (ch. 10)

Autocorrect — a phone feature that fixes words and suggests the next one while you type. (ch. 1)

Backpropagation — a training method in which the error travels back from the output into the network and each weight gets its share of the correction. (ch. 7)

Badly chosen reward — a points rule that lets the AI find a shortcut that earns points without doing what we wanted. (ch. 15)

Bias — when the training examples show more of one kind of thing than another, and the AI starts making more mistakes with the others. (ch. 4, 16)

Bullying — hurting or humiliating someone on purpose and repeatedly, including with fakes made by AI. (ch. 16)

Caution — a box in this book with tips for using AI responsibly and safely. (Before

You Begin)

Check — to confirm an AI's answer in a book, on a reliable website or with an adult before using it. (ch. 16)

Comment — a note written inside a program for people to read; the computer skips that part. (ch. 17)

Computer vision — the part of AI that works with photos and videos, like recognizing objects and faces. (ch. 8)

Confident answer — a text written firmly and full of details, which does not guarantee that it is right. (ch. 12)

Context window — the amount of text, counted in tokens, that the model can consider at one time. (ch. 11)

Cutoff date — the date up to which the training texts were collected; after it, the model does not know the news. (ch. 11)

Data — information stored in a way the computer can use, like photos, texts, numbers and clicks. (ch. 4)

Deep Blue — the IBM computer that beat world chess champion Garry Kasparov in 1997 by calculating many moves quickly. (ch. 2)

Deep learning — learning with neural networks that have many layers. (ch. 7)

Deepfake — a fake image, video or voice made with AI that looks real. (ch. 14, 16)

Did you know — a box in this book with a true and interesting fact that did not fit in the main text. (Before You Begin)

Diffusion — a way of generating images that starts from static and cleans it up little by little, guided by the request. (ch. 14)

Edge — a place in an image where light turns dark all of a sudden; it is one of the first things a network learns to find. (ch. 8)

Embedding — the technical name for the list of numbers that serves as a token's

address on the meaning map. (ch. 9)

Environment — the world where the agent acts, like a game level or a board. (ch. 15)

Example — a case shown to the program together with the right answer, like a photo labeled "cat." (ch. 3)

Expert system — a program full of "if... then..." rules written by people who know a subject very well. (ch. 2)

Fine-tuning — extra training, done after pretraining, with examples of one specific kind, such as good conversations. (ch. 11)

Garbage in, garbage out — a saying that reminds us that bad data leads to bad results. (ch. 4)

Gradient descent — the way of adjusting the knobs by walking down the "error hill," one small step at a time, always toward the downhill side. (ch. 5)

Graphics card — a computer part made for games that does many simple calculations at the same time and is used to train AIs. (ch. 2, 7)

Hallucination — when the AI writes something false that looks true, like a book that does not exist. (ch. 12)

Hidden layers — the middle layers of a network, between the input and the output. (ch. 7)

"If... then..." rule — an instruction that tells the computer to do one thing when another thing happens, like in a recipe. (ch. 3)

Imitation game — the name Turing gave to the test that today carries his name. (ch. 2)

Label — the right answer that comes with an example, like the answer key for an exercise. (ch. 4)

Language model — a program trained to guess which words should come next in a text. (ch. 10)

Layer — a row of neurons that receives the numbers from the previous layer and

passes its result to the next one. (ch. 7)

Lia and Theo — fictional 12-year-old cousins who ask the questions at the start of some chapters. (Before You Begin)

Machine learning — a way of making programs in which the computer finds patterns from examples, instead of being handed the rules. (ch. 3)

Made-up citation — a reference to a book, website or quote that the AI invented and that does not really exist. (ch. 12)

Meaning map — a space where each token gets an address made of numbers, and similar words end up close together. (ch. 9)

Model — the program that learns: it receives something and gives back an answer. (ch. 5)

Nearest neighbor — a way of deciding that looks, among the examples, for the one most like the new thing and copies its name. (ch. 17)

Neural network — a set of artificial neurons connected in layers. (ch. 7)

Noise — random dots with no picture in them, like the static on a TV with no signal. (ch. 14)

Overfitting — when the model memorizes the training examples and does badly with new ones. (ch. 5)

Parameters — the adjustable numbers inside a model, like the knobs on a radio. (ch. 5)

Pattern — something that repeats across the examples of one group and helps tell it apart from the others. (ch. 3)

Perceptron — one of the first artificial neurons able to learn from examples, presented in 1958. (ch. 6)

Personal data — information that shows who you are or where to find you, like your

full name, address, school, passwords and photos. (ch. 16)

Pixel — the smallest little square in a digital image, with just one color. (ch. 8)

Pretraining — the first stage of an assistant, in which the model reads a huge amount of text while guessing the next word. (ch. 11)

Privacy — the right, and the care, to decide who gets to know your information. (ch. 16)

Program — a list of instructions the computer follows. (ch. 1)

Programming — writing instructions for the computer to follow, step by step. (ch. 17)

Programming language — a way of giving the computer orders using text, like Python. (ch. 17)

Prompt — the request, in text, that you write for an AI. (ch. 14)

Python — a programming language written with words, used a lot by people who work with AI. (ch. 17)

RAG — the short name for the technique of fetching documents and handing them to the AI so it answers based on them, like an open-book test. (ch. 13)

Recipe — a comparison for a traditional program: a list of steps that is followed without skipping any. (ch. 3)

Reference date — the month when the book was written, October 2026; products and news may change after that. (Before You Begin)

Reinforcement learning — a way of learning by trying, earning or losing points, and repeating. (ch. 15)

Reliable source — a serious place where you can check a piece of information, like a textbook or an official website. (ch. 12)

Reward — the point the agent earns or loses after an action. (ch. 15)

RGB — a way of storing a color with three numbers: how much red, green and blue. (ch. 8)

Sampling — picking the next word at random from among the most likely ones,

which makes the same question give different answers. (ch. 10)

Scratch — a language in which you program by snapping colored blocks together, good for starting out. (ch. 17)

Simulation — a make-believe world inside the computer where robots and programs train without breaking anything. (ch. 15)

Spam — an unwanted or misleading advertising message, which AI filters try to separate from the rest. (ch. 1)

Supervision — a person following along, checking and approving what the AI does. (ch. 13)

Synthetic voice — a computer-made voice that reads a text or imitates someone's speech. (ch. 14)

Test set — examples kept aside that the AI never sees in training and that serve as the final exam. (ch. 4)

Token — a piece of text, a whole word or part of one, that the model treats as a single unit. (ch. 9)

Tool — a program the AI can call for help, like a calculator, web search or a map. (ch. 13)

Training — showing many examples, measuring the error and adjusting the model's knobs, over and over. (ch. 5)

Transformer — a kind of neural network built around attention, presented in 2017, which is the basis of today's assistants. (ch. 10)

Trick question — a question that already carries a wrong idea inside it and can lead the AI to invent an explanation. (ch. 12)

Try it — a box in this book with a no-computer activity that shows an idea in practice. (Before You Begin)

Turing test — a game proposed by Alan Turing in 1950 in which a judge chats by text and tries to figure out whether they are talking to a person or a machine. (ch. 2)

Web search — a tool that lets the AI find information more recent than what it read

during training. (ch. 13)

Weight — a number that says how hard a clue pulls a neuron's decision, and in which direction. (ch. 6)