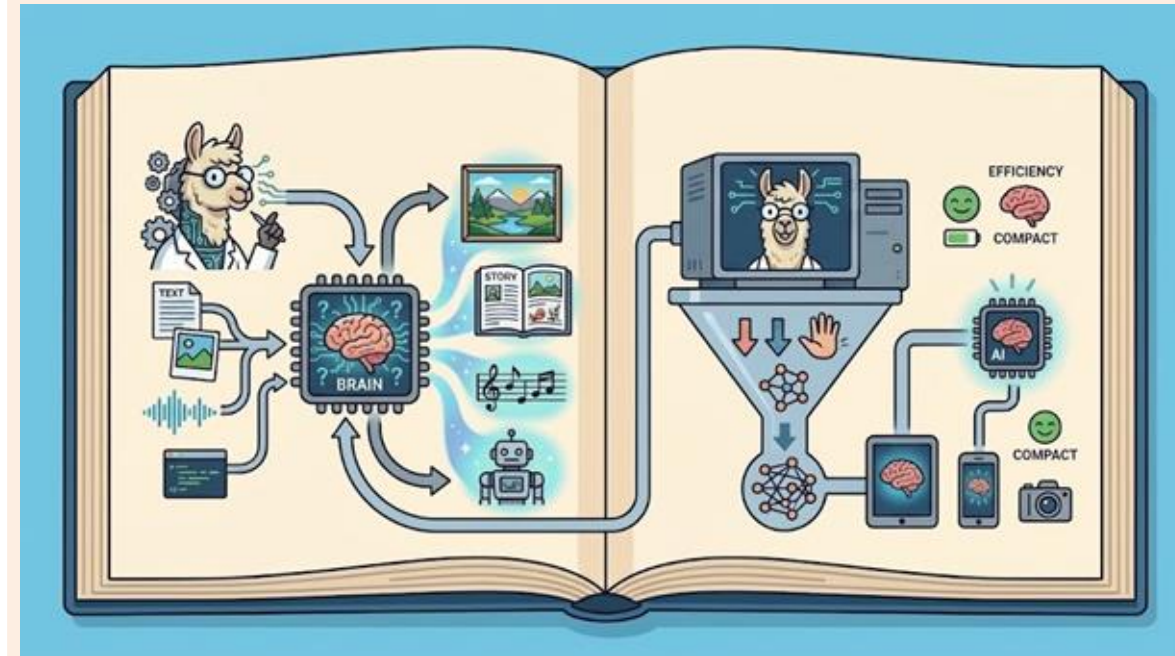


How Generative AI Works, and How It Gets Small

The ideas behind the projects in this book

No commands in this chapter — concepts only

Marcelo Rovai • UNIFEI



WHAT THIS CHAPTER IS FOR

By the end you will have the vocabulary.

Tokens · embeddings · attention · pretraining and post-training · pruning · distillation · quantization · the harness. Everything from here is practice.

Seven stages, one shared vocabulary at the end

- 1 Generative?**
Creating new content rather than sorting existing content into boxes.
- 2 The edge**
The model lives on the device — and the energy argument for putting it there.
- 3 Scales**
CloudML, EdgeML, TinyML: one continuum, and how to pick a tier.
- 4 LLMs & SLMs**
Transformers, tokens, embeddings, and how these models get trained.
- 5 Getting small**
Pruning, distillation, quantization — what makes edge deployment possible.
- 6 Agents**
From assistant to agent, and the harness that makes the difference.
- 7 Tooling**
Four ways to run an SLM at the edge — and which one this book uses.

STAGE 1 OF 7

What makes a model "generative"

Creating new content, rather than sorting existing content into boxes.

1

THE DISTINCTION

Same board, same Linux, different kind of machine

PART 2 • A CLASSIFIER

Picks from a fixed list

It looks at a photograph and answers 'robot' or 'periquito'.

It chooses from a set of labels it was trained on. It cannot produce anything outside that set.

PART 1 • A GENERATIVE MODEL

Writes something new

It is asked about a mosquito breeding site and writes a sentence nobody has written before.

It learned patterns from a large body of data and produces outputs that did not exist.

Same hardware, same operating system, completely different kind of machine.

THE LANDSCAPE

The families you have heard of, by size and modality

Where they run	Text	Vision + language
Data center	GPT, Claude, Gemini, Grok, Kimi	Gemini, GPT
Edge devices	Gemma, Phi, Llama, Qwen, SmoLLM	Moondream, PaliGemma, Florence, Qwen-VL

Hold on to that second row. Those are the ones that will actually run on the hardware in this book.

STAGE 2 OF 7

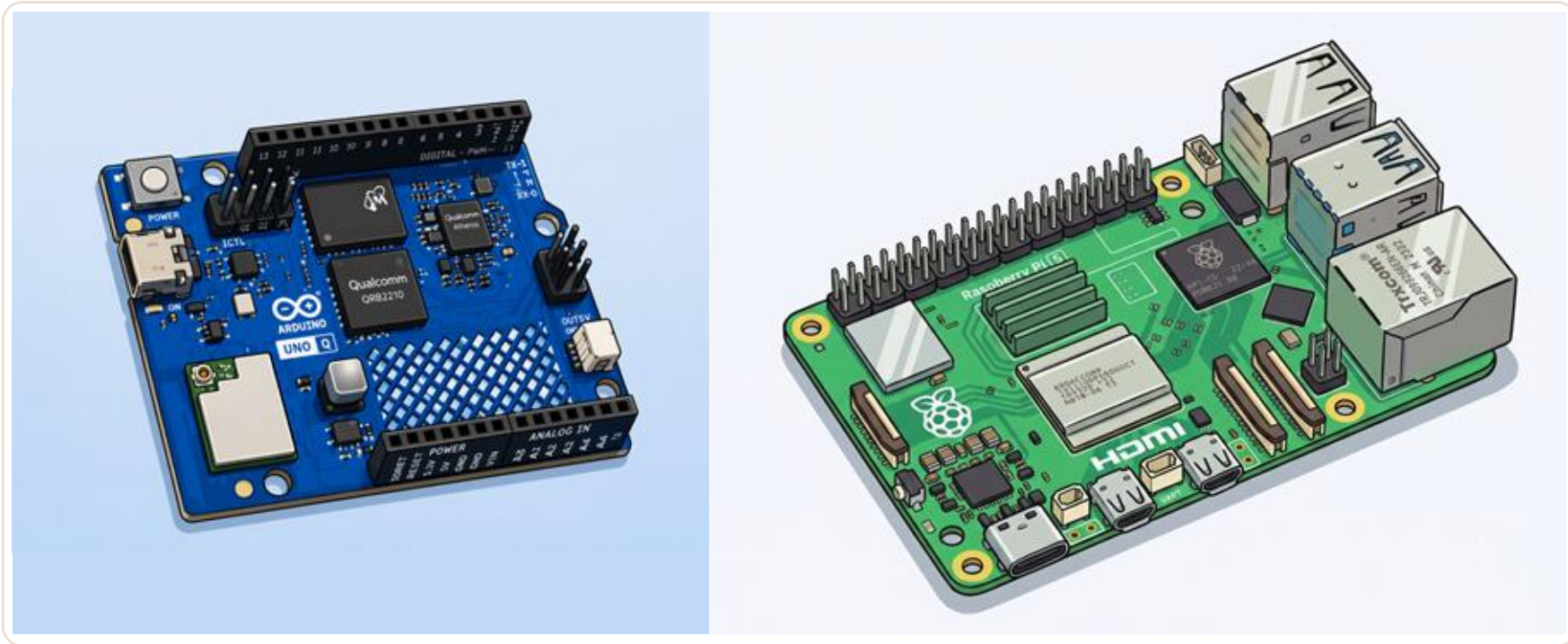
What "at the edge" adds

2

The model lives on the device — not in a data center.

THE TREND WORTH NAMING

Small, specialised and local keeps getting more viable



Sub-10B models are increasingly competitive with much larger generalists on specific tasks

A focused model trained well on a narrow domain does not need the breadth of a frontier model. This is a moving target — but the direction has held for years.

THE ENERGY ARGUMENT

Rough power draw, by where inference happens

Microcontroller (TinyML)	~0.5 W
Edge SBC — UNO Q, Raspberry Pi	~5–20 W
Cloud GPU (server side, per query)	100 W+

100×

more energy-efficient,
order of magnitude

Multiply that across billions of inferences a day and sustainability stops being a footnote — it becomes an engineering and economic argument.

AND A THIRD ARGUMENT

Offline. A few watts. Nothing per token.

For a project in a developing country: a device that runs offline, draws a few watts and costs nothing per token is usable in places where a cloud subscription and reliable connectivity are not assumptions you get to make.

STAGE 3 OF 7

One field, many scales

3

Machine learning lives on a spectrum defined by where the computation happens.

THE SPECTRUM

Three tiers, one continuum

CLOUDML

Data centers

Massive models, effectively unlimited compute.

The cost is latency, connectivity and privacy.

EDGEML

Capable local devices

Phones, single-board computers, the Arduino UNO Q.

Models run locally, on hardware you own.

TINYML

Microcontrollers

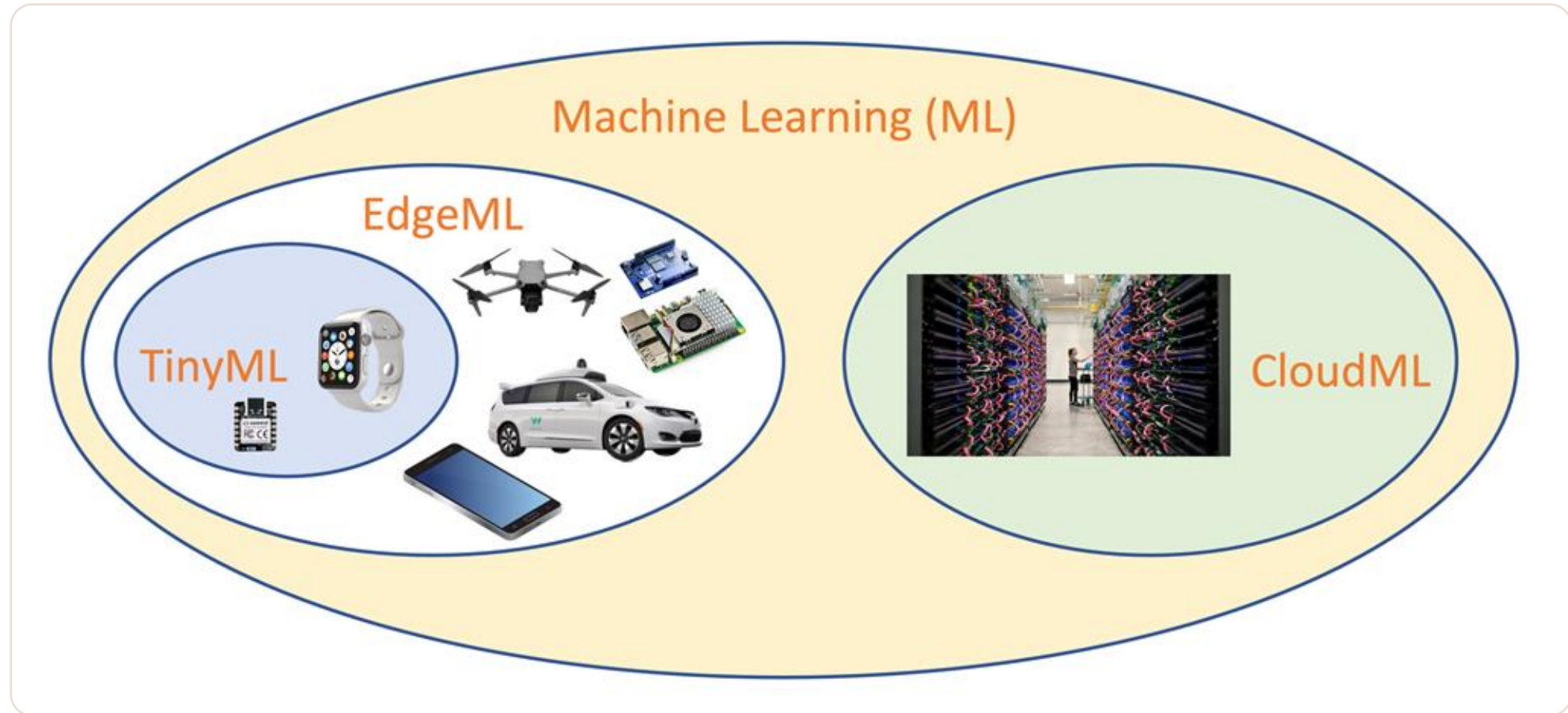
Milliwatts, kilobytes of RAM, no operating system at all.

Machine learning where there is barely a computer.

The UNO Q is interesting precisely because it straddles two of these at once.

THE SPECTRUM

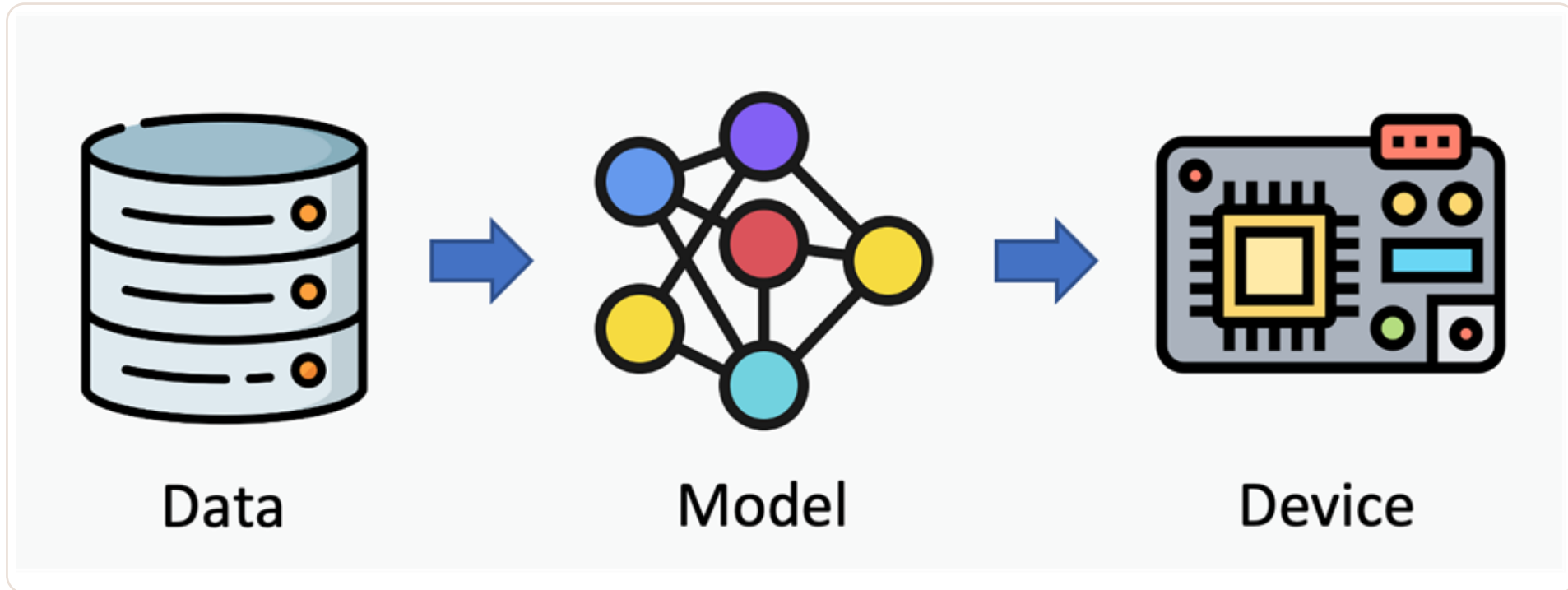
From microcontrollers to the cloud



Not competing camps — a continuum

THE THREE PILLARS

data → model → device

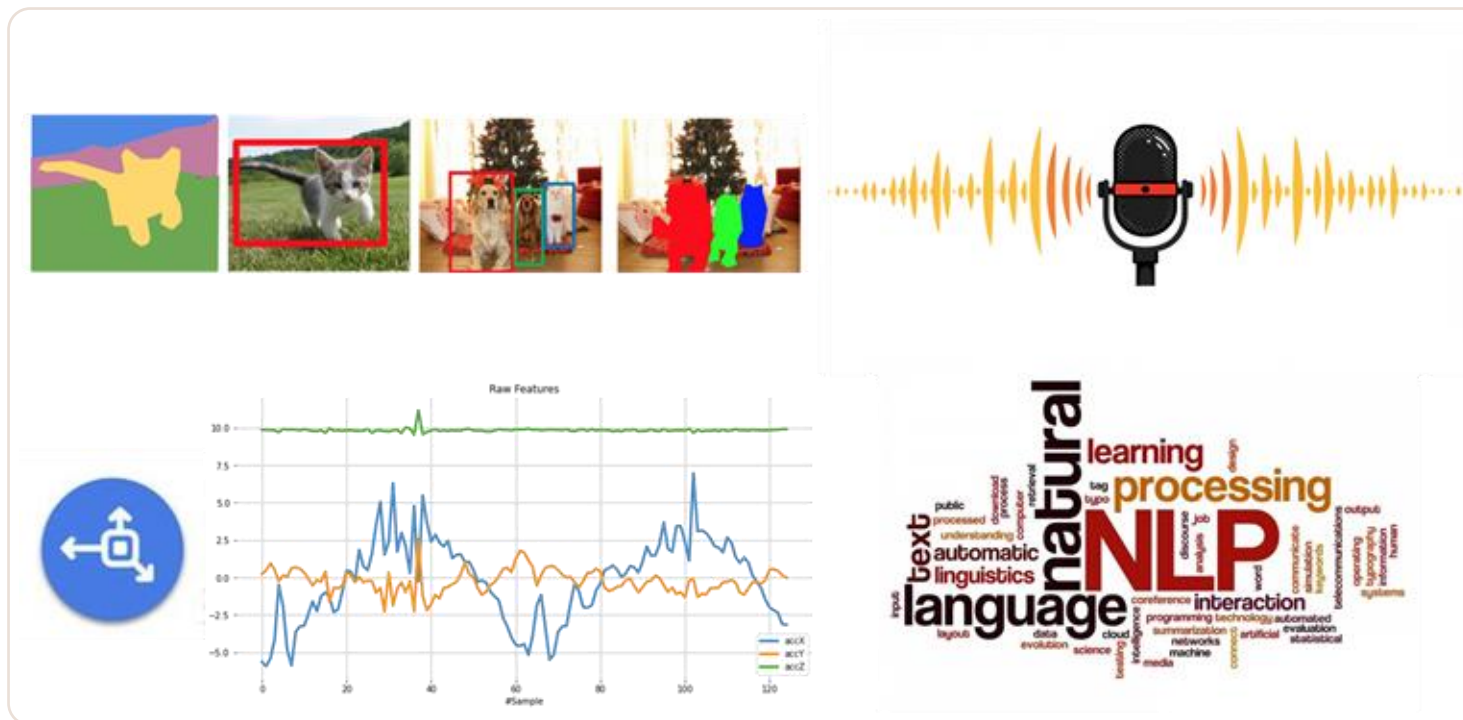


Whatever tier you target, the deployment pipeline has the same shape

Collect and prepare data · train and optimise a model · deploy onto hardware. Nearly every engineering decision at the edge is a trade-off among those three.

WHY NEURAL NETWORKS

Most edge data is unstructured



Sound, images, vibration, raw text — it does not arrive in neat rows and columns

That is exactly why neural networks matter here: they find patterns in messy, high-dimensional, real-world signals that rule-based code cannot keep up with.

ARCHITECTURES

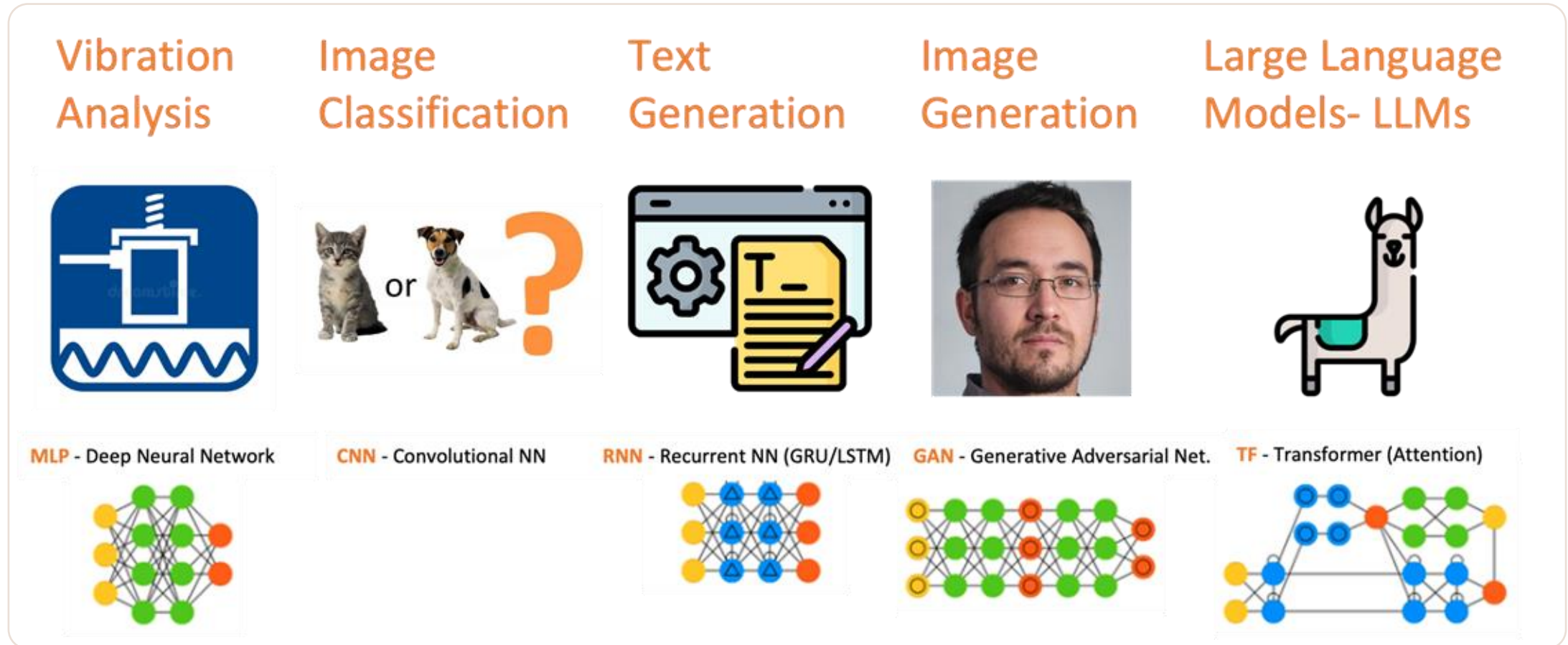
Different problems, different networks

Application	Architecture
Vibration analysis, simple classification	MLP — deep neural network
Image classification	CNN — convolutional neural network
Sequence and time-series tasks	RNN — GRU / LSTM
Image generation	GAN — generative adversarial network
Text generation, LLMs	Transformer (attention)

Language models sit at the top of that complexity ladder. That is exactly why running them on small hardware is hard — and why the rest of this chapter is mostly about making them smaller.

ARCHITECTURES

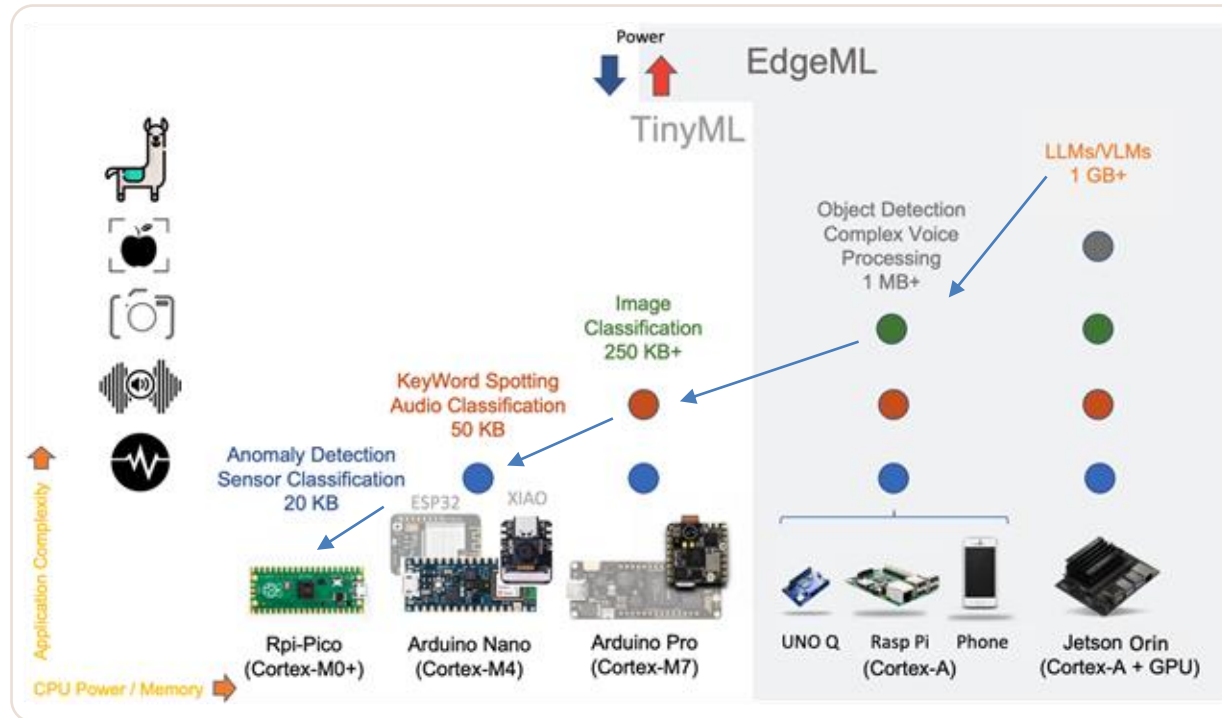
The same ladder, as a picture



From MLPs up to transformers

COMPLEXITY AGAINST HARDWARE

Where SLMs become feasible at all



An ESP32 handles classic TinyML; above it sit the UNO Q, then Pis and phones

The region beginning around 1 GB of memory is where SLMs and small VLMs become possible.

THE GOAL

Not the biggest model the board will tolerate.

It is to land on the smallest, cheapest device that still does the job well.

STAGE 4 OF 7

Language models: LLMs and SLMs

Neural networks built on the Transformer architecture.

4

WHY THE TRANSFORMER WAS A LEAP

Two reasons it beat earlier sequence models

LONG-RANGE DEPENDENCIES

It can look far back

Earlier sequence models like RNNs degraded over distance. Transformers handle long-range relationships in text far better.

PARALLELISM

Tokens processed together

Not strictly one after another. That parallelism is a large part of why these models could scale the way they did — you can throw far more hardware at training them.

The architecture was published in 2017. It did not produce one model — it set off an explosion of them.

THE LANDSCAPE

Open, open-from-China, and closed

Group	Models	Weights
Open — China	GLM, Kimi, Qwen, DeepSeek	Downloadable
Open	Phi (Microsoft), Gemma (Google), Llama/Muse (Meta)	Downloadable
Closed	Gemini, GPT, Claude, Grok	API only

The closed models are excellent and irrelevant here: you cannot download them, so you cannot run them on a board with no network connection.

WORTH STATING PLAINLY

Local inference exists because part of the frontier released weights.

Qwen — a Chinese open-weights model — is what every generative project in this book actually runs. A striking share of that openness currently comes out of China.

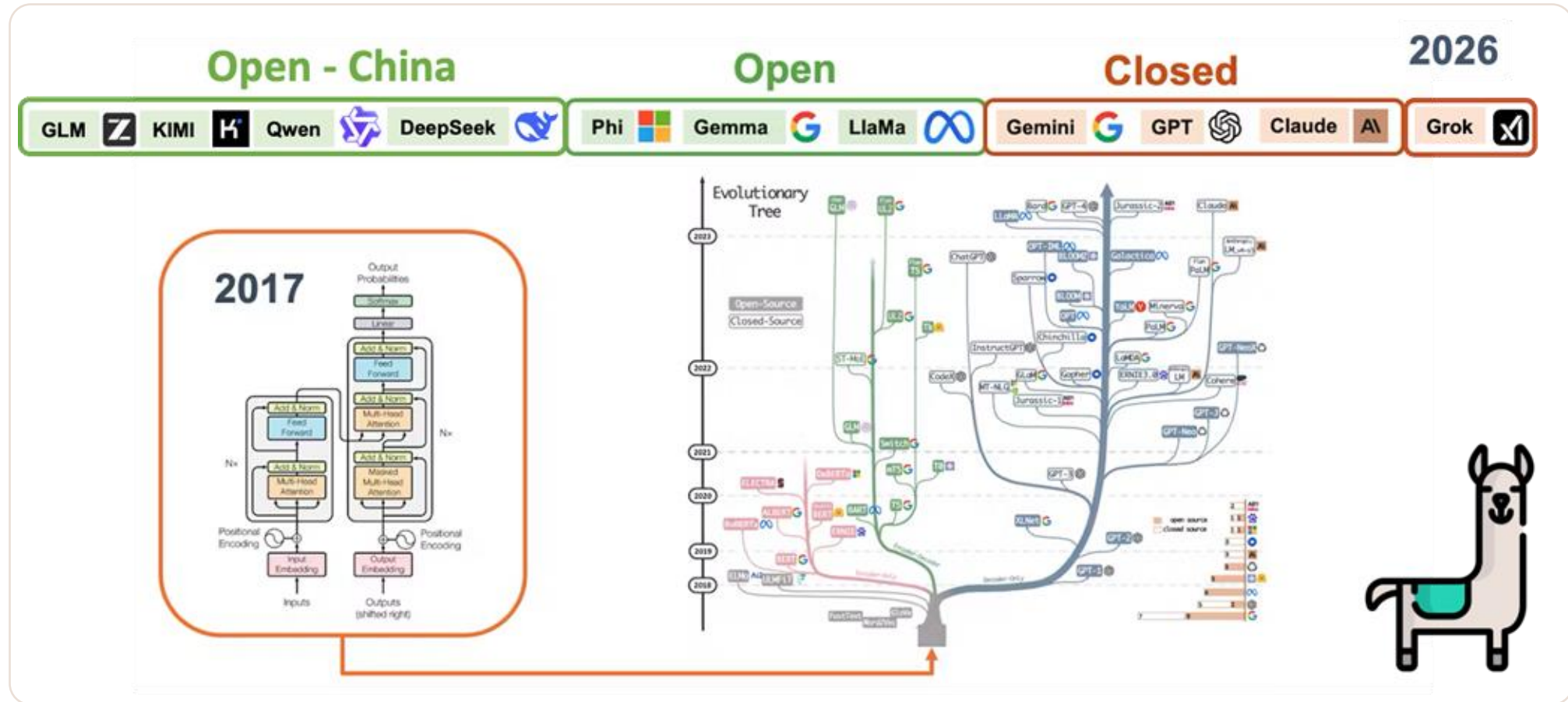
ONE CAUTION

Open does not mean small.

Kimi K3 is open-weights and Chinese — and it is a terabyte-scale model that needs a data centre. Downloadable and runnable on your hardware are separate questions, and a model has to satisfy both.

THE EXPLOSION

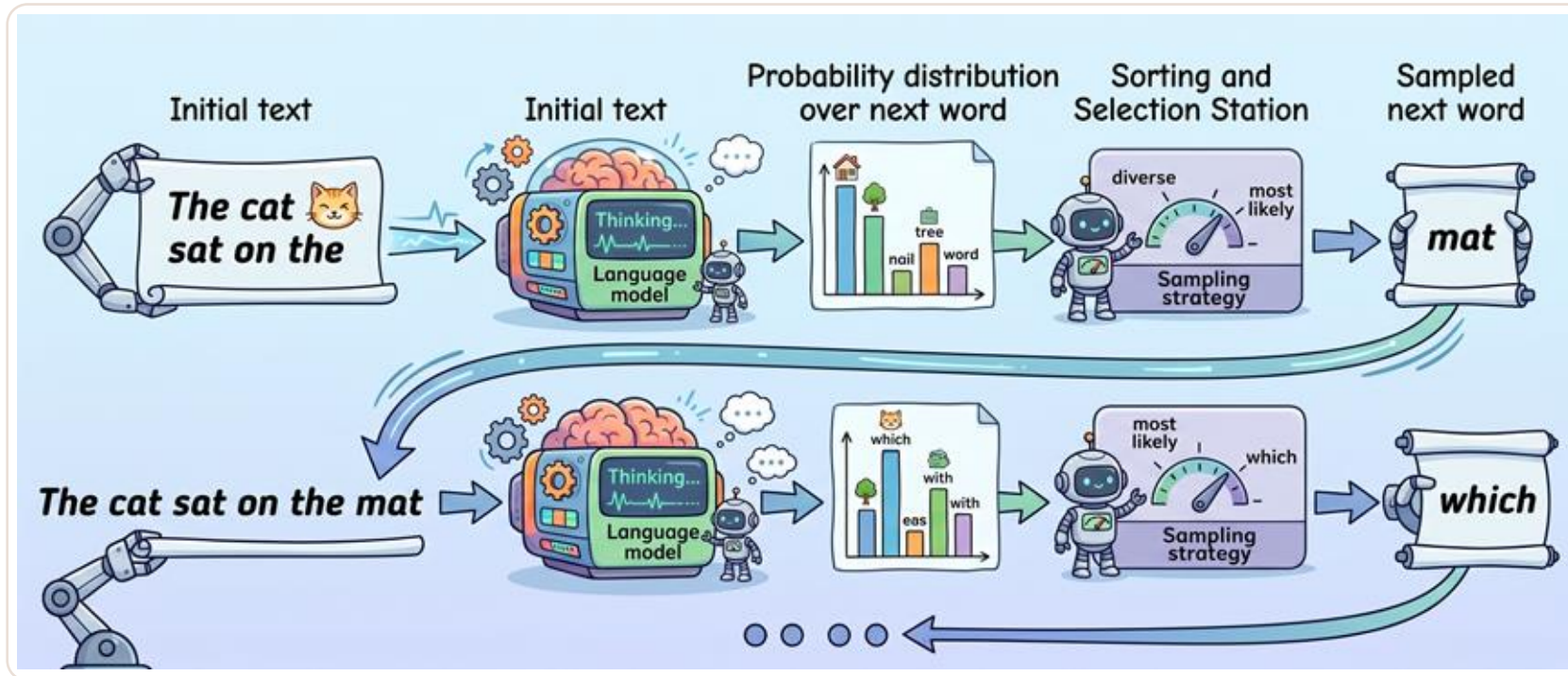
One architecture, an entire landscape



From the 2017 paper to the model landscape of 2026

TOKENS COME FIRST

Text is broken into sub-word pieces, not words



'Understanding' might be two or three tokens; a rare technical term might be five

Your context window — and therefore your memory budget — is measured in tokens. This is why later chapters tell you to keep context at 512–1024.

TRY IT YOURSELF

Three tools that build more intuition than any diagram

1

The OpenAI tokenizer

platform.openai.com/tokenizer — type a sentence and watch it split. Five minutes here is worth more than any explanation I can write.

2

TensorFlow Embedding Projector

projector.tensorflow.org — rotate through the embedding space and watch related words cluster together.

3

The Transformer Explainer

poloclub.github.io/transformer-explainer — walks you through attention layer by layer, on a live model, in your browser.

If you are teaching from this book, these are worth ten minutes of class time.

EMBEDDINGS

A vector whose position encodes meaning.

Each token becomes a numerical vector placed in a high-dimensional space, so that words with similar meanings sit near each other. Embeddings are the bridge between human language and the pure arithmetic the model actually performs.

HOW THESE MODELS ARE TRAINED

Two phases, and what each one buys

PHASE 1

Pretraining

The model digests an enormous amount of text and learns language structure and world knowledge.

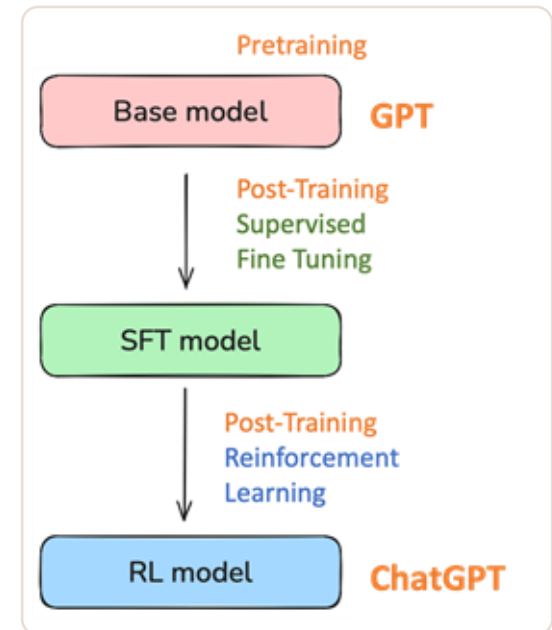
Expensive, done once, by organisations with large budgets.

PHASE 2

Post-training

Supervised fine-tuning teaches it to follow instructions.

Reinforcement learning from human feedback shapes it toward being helpful and aligned.



Pretraining is what makes a model knowledgeable. Post-training is what makes it usable.

WHY THE FILENAME MATTERS

-Instruct and -it mean post-trained.

The model files you download in later chapters carry that suffix. The raw pretrained base model would not hold a conversation at all.

STAGE 5 OF 7

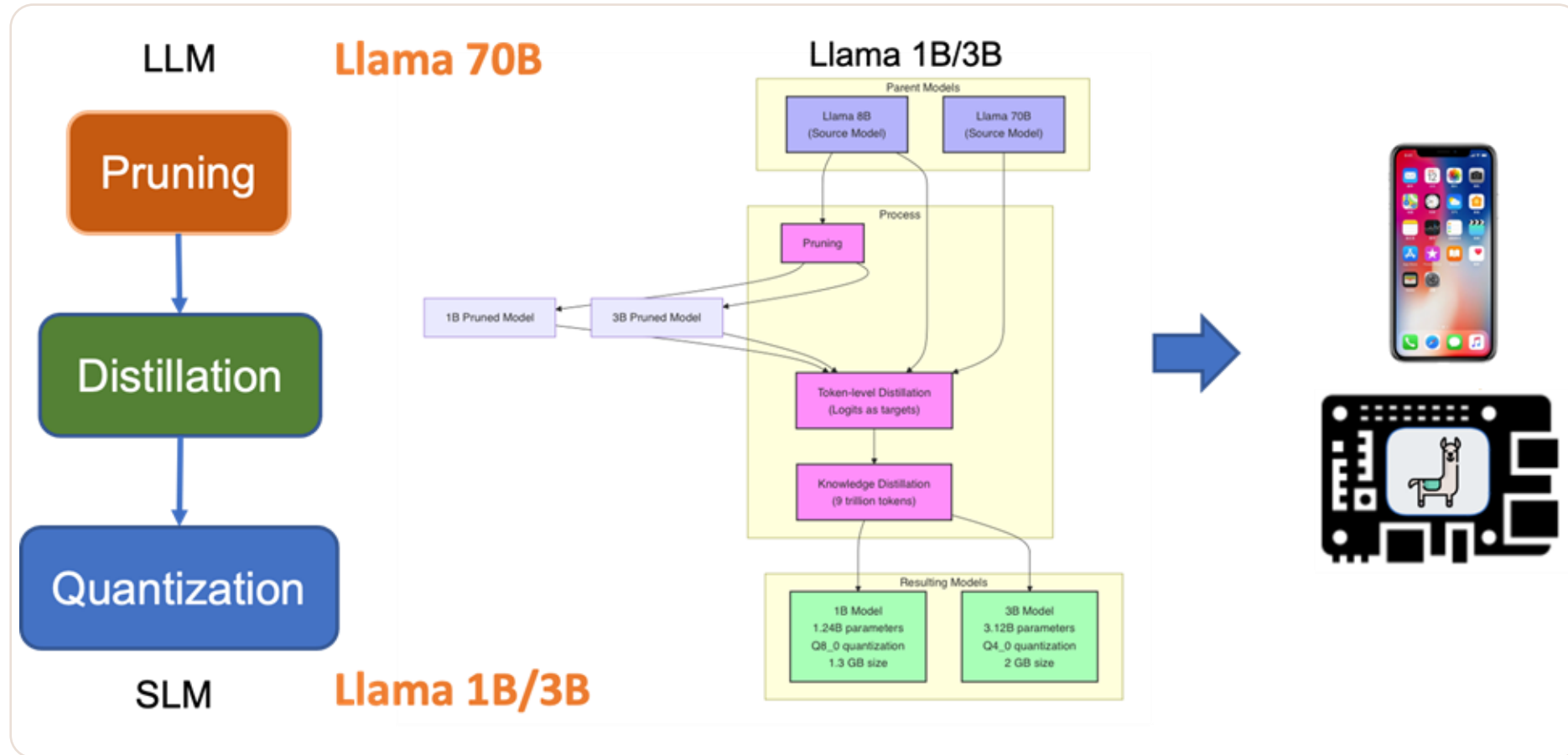
How big models become small

5

Three techniques that make edge deployment possible at all.

THREE TECHNIQUES

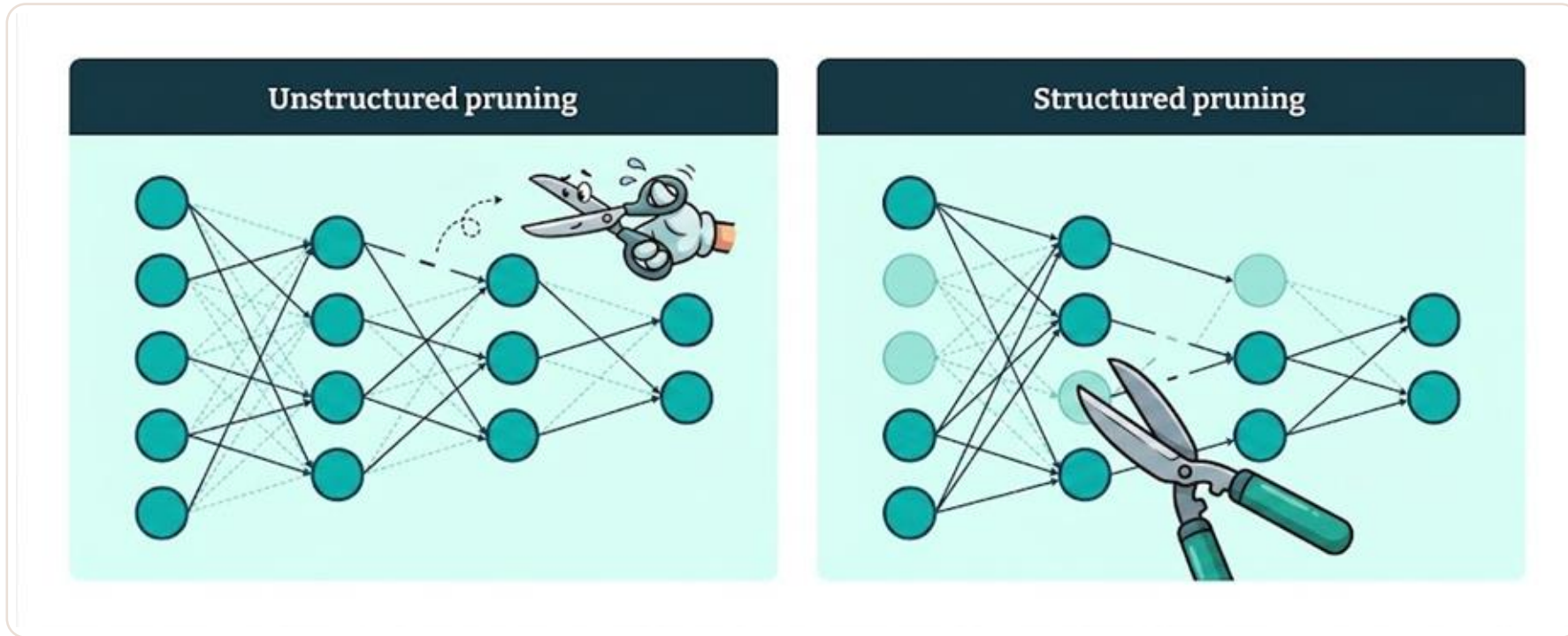
From 70B to something that fits



Pruning · knowledge distillation · quantization

PRUNING

Remove the weights that contribute almost nothing

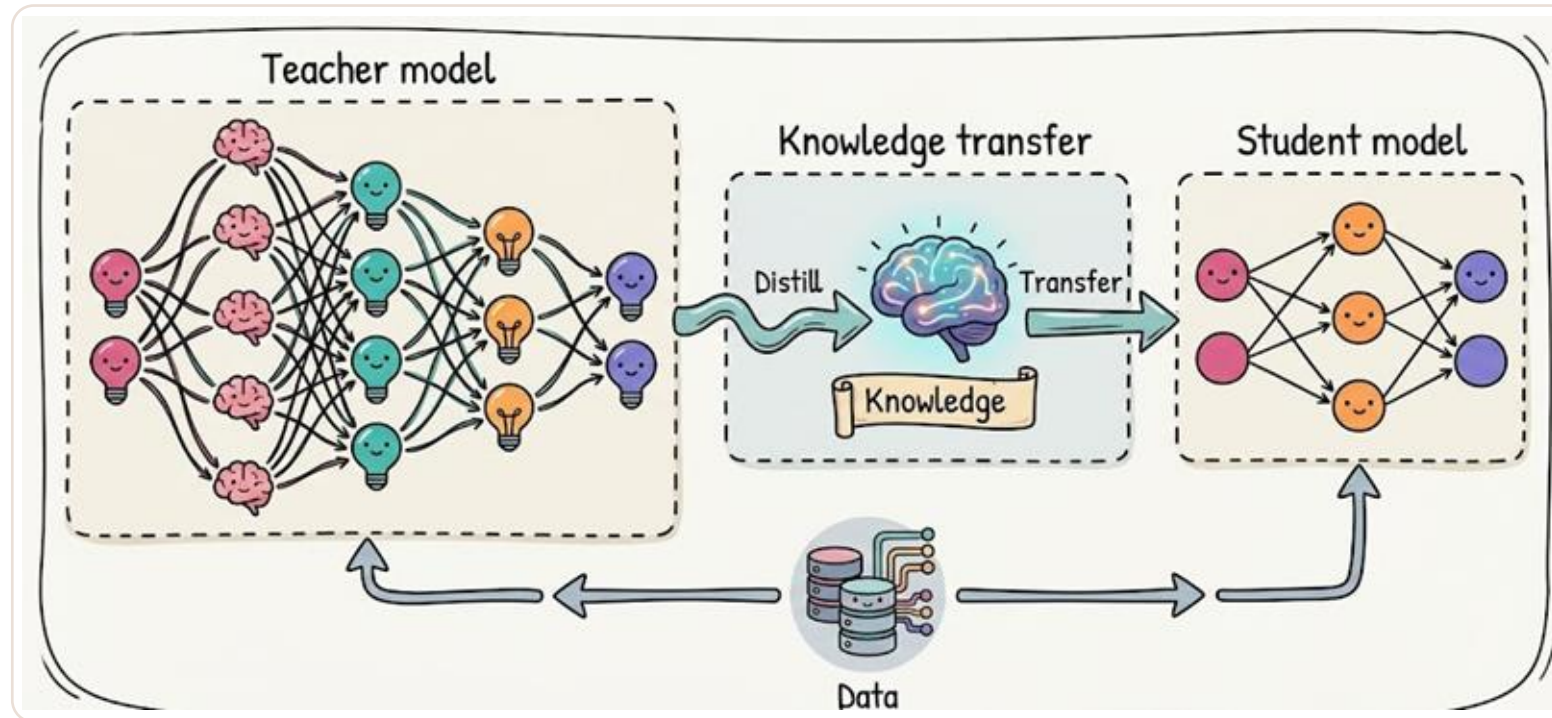


A trained network carries many weights that barely affect its output

Done carefully, you can cut a substantial fraction of the parameters for a small accuracy cost — and the leaner model occasionally generalises slightly better.

KNOWLEDGE DISTILLATION

A teacher trains a smaller student



The student learns from the teacher's full output distribution, not just the labels

That distribution carries far richer information than a single correct label: it shows what the teacher considered, and how confident it was.

WHY THIS MATTERS SO MUCH

A 1B model in 2026 beats a 7B model from 2023.

Most of the reason is better training data and distillation from stronger teachers. The result is a compact model that punches well above its parameter count.

QUANTIZATION

Store the weights at lower precision.

8-bit or 4-bit integers instead of 16-bit floats. A model at 4-bit takes roughly a quarter of the memory — and runs faster, because moving less data through memory is most of what inference costs on a CPU.

THE TRADE-OFF IS NOT LINEAR

Small models tolerate quantization badly.

Larger models handle aggressive quantization well; very small models do not, because they have less redundancy to spare. Choosing Q4 versus Q8 on a sub-1B model is worked through with real measurements in Chapter 2.

STAGE 6 OF 7

From assistant to agent

6

It is worth stepping back to see how fast the shape of these systems has changed.

THE SHIFT

Two generations of system

THEN

Assistants

They augmented human knowledge. You asked, they answered, the interaction ended.

Text in, text out, nothing else.

NOW

Agents

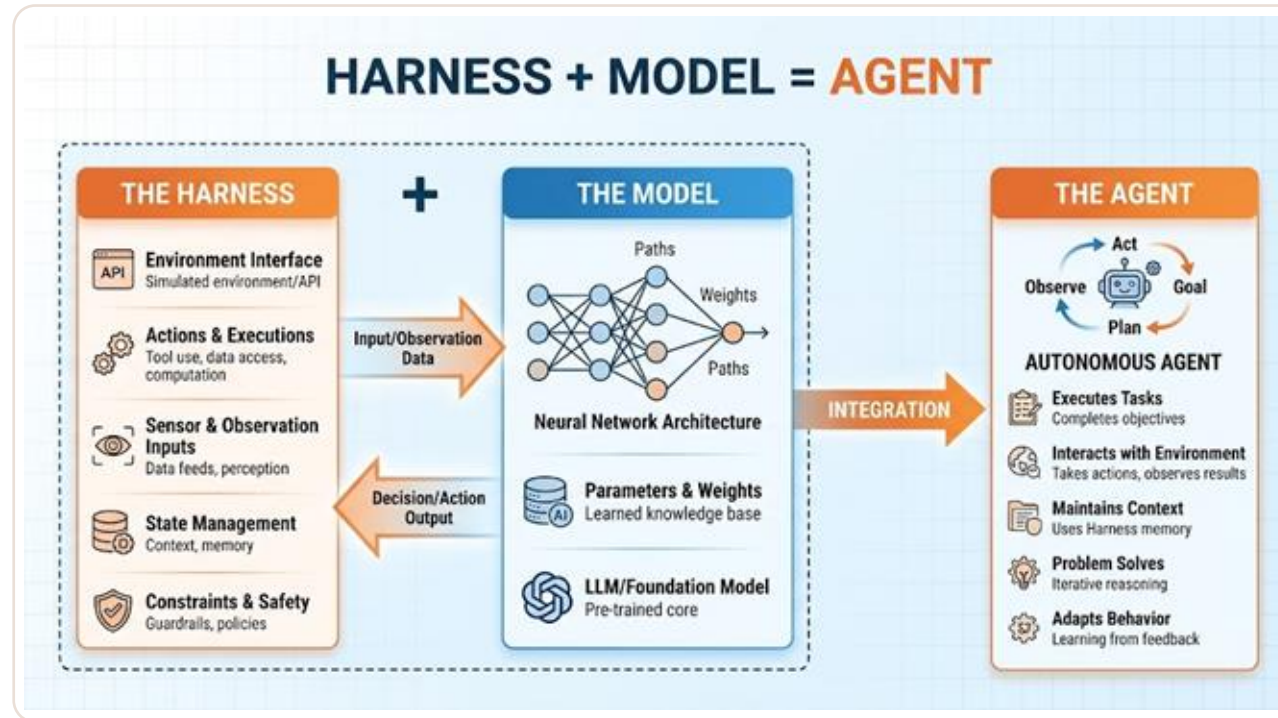
They work through tool-calling APIs. They reach into the world, use its infrastructure, and take actions with consequences.

The interaction is a loop, not an exchange.

That word — harness — will come back. Chapter 5 is entirely about building one.

THE HARNESS

The software around the model that makes this possible



The layer that lets a model call functions, use tools, and operate in a loop

At the edge, the harness is what turns a text generator into something useful — it reads a sensor, decides, and triggers an action.

STAGE 7 OF 7

Four ways to run an SLM at the edge

Which tool you choose depends mostly on your hardware.



THE OPTIONS

Four tools, and where each one fits

Tool	What it is	Where it fits
Ollama	Framework with a model registry and REST API	Easiest entry point. Excellent on Raspberry Pi. Does not run easily on the UNO Q
llama.cpp	Lightweight C++ engine for quantized models	Tested on the UNO Q. Preferred for control, compact size, low memory use
HF Transformers	Python library, broadest model coverage	Well supported on Raspberry Pi; heavy for very constrained boards
LiteRT-LM	Google AI Edge solution aimed at production	Optimised for efficient deployment across edge devices

You will meet Ollama and LM Studio in the next chapter, on your own computer, where the constraints are looser and the comparison is instructive.

FOR THIS BOOK

llama.cpp is the tool of choice.

It is the one that fits, the one that gives the most control over memory, and the one with a server mode we build several projects on.

WHERE THIS GOES

The vocabulary you now have

SECTION 1–2

generative
SLM / VLM
at the edge

SECTION 3

CloudML
EdgeML
TinyML
data → model → device

SECTION 4

tokens
embeddings
attention
pretraining
post-training

SECTION 5

pruning
distillation
quantization

SECTION 6

assistant → agent
the harness
tool-calling

SECTION 7

llama.cpp
Ollama
HF Transformers
LiteRT-LM

Everything from here is practice.

WHAT COMES NEXT

The path from here

1

The next chapter

Runs a model on the computer you already own, so the concepts land before any board is involved.

2

Then the UNO Q

First setting it up, then compiling llama.cpp on it.

3

Then capability

Giving the model vision, then sensors, then tools.

Nothing that follows requires you to remember the mathematics of attention.

TWO IDEAS TO CARRY

Why 4-bit differs from 8-bit. Why context costs memory.

Nothing that follows requires the mathematics of attention. It does require these two — and they come up in every chapter of Part 1.

Questions?

Prof. Marcelo J. Rovai

rovai@unifei.edu.br

UNIFEI — Federal University of Itajubá, Brazil

AiEng4D — Academic Network Co-Chair

EdgeAIP — Academia-Industry Partnership Co-Chair

