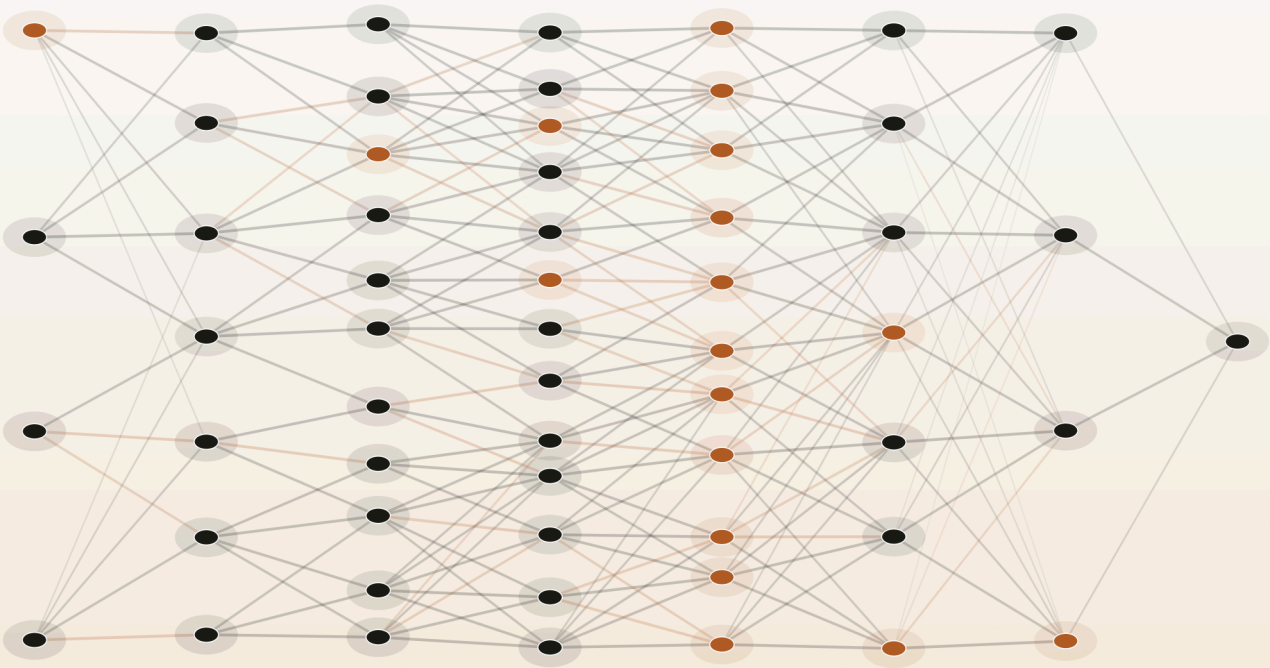
A PRACTICAL FIELD GUIDE

Build Your Own *AI*, in 2026

An honest look at the two real paths — bare metal from total zero, and multi-model fusion — with prices, tradeoffs, and a Hebrew guitar teacher as a running example.



BY *Iddo*

2026 · 05 · 04

Contents

PART ONE

I

Two paths, honestly described

> *"Pretending you're on the bare-metal road when you're actually doing fusion is the most common mistake in this whole field."*
>
> FROM CHAPTER TWO

CHAPTER 1

What "build your own AI" actually means in 2026

Two genuine paths. Neither is wrong. Most real builders are taking the second one and pretending it's the first.

If you walked into a room of people who say they "built their own AI" in 2026, you would find two very different kinds of work being done under that single phrase.

The first kind is the bare-metal path. You write the autograd loop yourself, you implement attention from scratch, you collect a dataset, you watch the loss curve descend over hours or weeks of GPU time, and at the end of it you have a model whose every weight you understand because you put it there. This is the path of *sovereignty and education*. It is also the path of stopping at "toy" scale, because pretraining a 7-billion-parameter model from random initialisation costs between fifty thousand and five hundred thousand US dollars in compute — not a thing you do at a desk in Tel Aviv.

The second kind is multi-model fusion. You take the open-weight models that the frontier labs released last quarter — Llama 3.3, Qwen 3, DeepSeek-V3, Whisper, Stable Diffusion, MediaPipe — and you fuse, fine-tune, chain, prompt, crop and recombine them into something that fits a use case the original authors never imagined. This is the path of *practicality and shipping*. It is what almost every successful indie AI product in 2026 actually is, beneath the marketing.

> *The honest sentence: in 2026, "I built my own AI" almost always means "I assembled and fine-tuned a stack of open weights for my specific problem." That is not a lesser thing. It is a different and equally legitimate thing.*

§ Why this guide exists

The internet is full of two failure modes on this topic. The first is the YouTube-tutorial path: "build a transformer in 200 lines of PyTorch!" — technically true, useless for anything you would actually deploy. The second is the consultant path: "just call the OpenAI API" — which is fine until you need Hebrew, or offline, or your data, or you don't want to pay forty cents per long conversation forever.

This guide picks an honest middle. It treats the two paths as siblings — one to learn deeply and one to ship usefully — and walks the same example, a Hebrew-speaking guitar teacher AI, down both. It puts real 2026 Israel-landed prices on the hardware. It does the three-year math on owning a workstation versus renting cloud GPUs. And it admits, where appropriate, that the popular answer is wrong.

§ The running example: a Hebrew guitar teacher

Across this guide we will refer to the same project: an AI that watches a beginner play guitar through a webcam, listens through a microphone, and gives spoken feedback in Hebrew. Concretely, it must:

- Understand the student's spoken Hebrew questions ("why does this chord buzz?")
- Recognise hand position on the fretboard from video
- Hear what the student is playing and identify the pitches and timing
- Reason pedagogically — "you're flat on the G string because your finger is too far from the fret"
- Speak back in natural Hebrew, with terminology a beginner can follow

This is a real, specific, achievable project for a single person in 2026. It also has the property that no single off-the-shelf model can do all of it. You must combine. That is what makes it the right teaching example.

§ What you'll find ahead

Chapter 2 lays out the two paths side by side, with the honest cost and effort numbers. Chapters 3 and 4 walk the guitar teacher down each path concretely. Chapter 5 puts real 2026 prices on three home-lab hardware tiers. Chapter 6 compares those to cloud GPU rental and computes the break-even point. Chapter 7 covers the things people forget — electricity, depreciation, your time. Chapter 8 talks about what your data actually means in each option. Chapter 9 is one page of a recommended starting kit for someone in your exact position.

We will not be precious. Where one path is clearly better, we'll say so. Where the popular answer is wrong, we'll say that too.

CHAPTER 2

The two paths, side by side

Bare metal from total zero, or fuse what already exists. The honest table.

Before we walk the guitar teacher down each path, let us look at what each path actually is, what it costs, and what it gives back. The numbers in this chapter are not aspirational — they reflect what a single person, working from a desk in Israel in 2026, can actually do.

Path A — Bare metal from zero

- You write the model architecture in PyTorch or JAX
- You assemble or pretrain on your own dataset
- You own every weight and every design decision
- Realistic ceiling at home: ~100M–1B parameter models, or fine-tuning of larger ones
- Best for: deep learning, custom architectures, sovereignty over a niche model
- Worst for: shipping a useful product this quarter

Path B — Multi-model fusion

- You take open-weight models from Hugging Face
- You fine-tune the parts that need to be domain-specific (LoRA / QLoRA)
- You chain them together in your own glue code — the orchestration is yours
- Realistic ceiling at home: production-grade Hebrew assistants, multimodal pipelines
- Best for: shipping in weeks not years, with frontier-grade quality
- Worst for: claiming you "built the model" — you didn't, you assembled it

§ The honest comparison

One picture is worth a paragraph here. The chart on the left is a log scale — bare-metal pretraining is roughly two orders of magnitude more effort than fusion for the same shipped capability. The chart on the right shows what each path is actually *good* for. They are not redundant; they win in different dimensions.

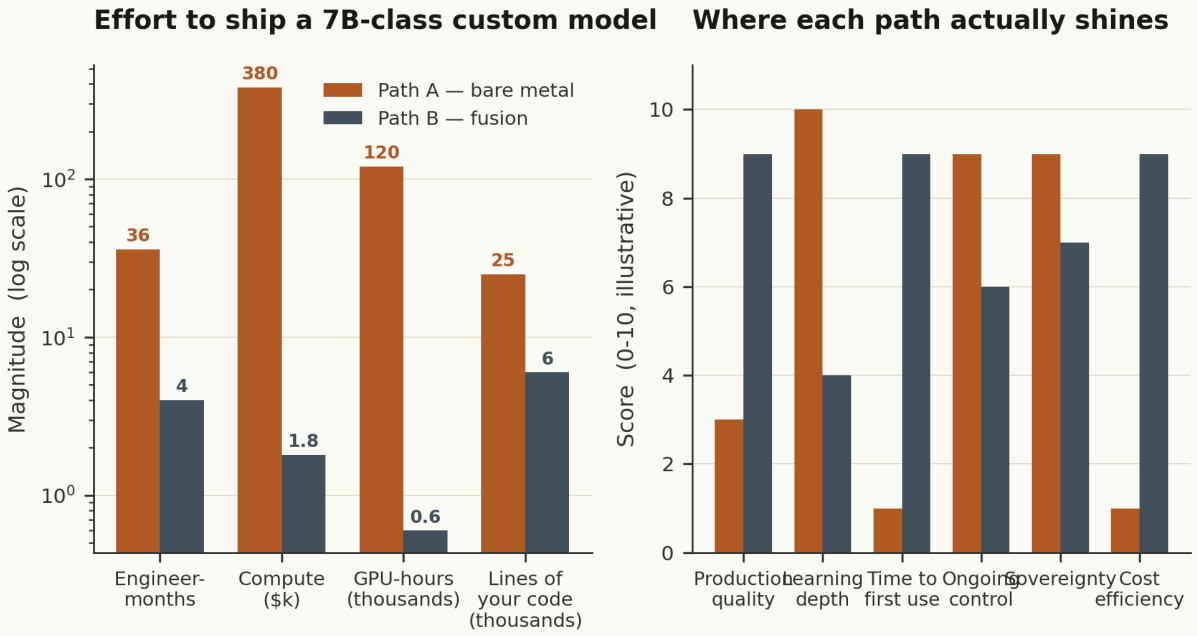


FIGURE 2.1 Left: order-of-magnitude difference in resources to ship a 7B-class custom model from scratch versus from open weights. Right: where each path actually wins. Numbers are illustrative for a single-developer project; full pretraining costs are based on published Llama-3 and DeepSeek-V3 compute figures, scaled to 7B.

§ What "from scratch" really means in 2026

This is the place to deflate a particular fantasy. Pretraining a useful general-purpose 7B language model from random initialisation requires roughly 1.5 trillion tokens of training data and 200,000 H100-hours of compute. At current cloud prices that is between 250,000 and 500,000 US dollars. At home electricity costs alone for the same wallclock would dwarf your hardware bill. **You will not pretrain a frontier-quality 7B from scratch at home in 2026. Nobody will.**

So when this guide says "Path A — bare metal from zero," it means one of three honest things, in descending order of how often you will actually do them:

1. **Toy models for understanding.** A 10M-parameter character-level transformer trained on a public-domain corpus, in two hours on your 4070. Educational gold. Not a product.
2. **Small custom architectures for narrow tasks.** A 50M-parameter encoder for Hebrew chord-symbol parsing. Trains overnight on a 4090. Useful as a component inside a larger fusion stack.
3. **Heavy fine-tuning of open weights as if it were from-scratch.** Take Llama-3.3-8B, run continued pretraining on 50 GB of Hebrew text and your own corpus, then instruction-tune. Several days on a Tier-2 box. Most of what people in 2026 mean by "trained my own model" is this.

◆ THE UNROMANTIC TRUTH

Item 3 above is what gets shipped. Item 1 is what gets blogged about. Pretending item 1 is item 3 is the most common mistake in this whole field. Honest builders know which one they are doing today and don't confuse the two.

§ So which one should you pick?

The right answer for almost everyone reading this is **both, in sequence**. Spend two weeks on Path A — build a small transformer from scratch following the Karpathy "nanoGPT" or Sebastian Raschka tradition — not because the resulting model is useful, but because nothing else gives you the same intuition for what is happening inside the open weights you'll spend the next two years fine-tuning. Then switch to Path B for everything you actually want to deploy.

If you ignore Path A entirely, you will end up cargo-culting fine-tune recipes you don't understand and your debugging will be miserable. If you live on Path A forever, you will spend five years on what should have been a six-week project. The boundary between them is the most useful place to live.

CHAPTER 3

Path A — the guitar teacher, from zero

If you walked the bare-metal road end to end, what would it look like? An honest sketch of the months ahead.

Let's pretend, for the length of this chapter, that you are stubborn. You want to build the Hebrew guitar teacher with as little dependence on other people's pretrained models as possible. What does the project plan actually look like?

§ What you'd have to build, by hand

Five real subsystems, each its own multi-month sub-project:

1. **A Hebrew speech recogniser.** A custom acoustic model + language model trained on the few hundred hours of public Hebrew speech that exist (ivrit.ai corpus, Mozilla Common Voice he, your own voice memos). This alone is a six-month research project for one person, and the result will be markedly worse than the off-the-shelf Whisper-large-v3 + ivrit.ai fine-tune that you can download tonight.
2. **A vision model that finds fingers on a fretboard.** You'd label thousands of frames of guitar video, train a convolutional or vision-transformer model from scratch, and build the fretboard-coordinate logic from raw pixel coordinates. MediaPipe Hands does the hand-landmark part for free; you'd be reinventing it.
3. **A pitch / chord recogniser.** A small CNN over CQT spectrograms. This one is genuinely doable from scratch in a few weeks — the field is mature enough that there are good academic tutorials.
4. **A pedagogical reasoning model.** A small Hebrew-speaking language model that knows enough about guitar pedagogy to give useful feedback. From scratch: infeasible. (See chapter 2 on pretraining costs.) Even fine-tuning a 1B model from random init on a hand-curated corpus would not match a 30-minute LoRA on top of Llama-3.3.
5. **A Hebrew text-to-speech synthesiser.** Roughly the same shape of project as the speech recogniser — corpus-bound, six months minimum, won't beat Coqui XTTS-v2 with a Hebrew fine-tune.

§ The honest order-of-magnitude

If you are a competent ML engineer working alone, full-time, with access to a Tier-2 home rig (RTX 4090, 64 GB RAM): **three to four years of work, and the result will be measurably worse on every axis than what you can ship via Path B in ten weekends**. The hardware would not be your bottleneck. The data would.

> **! THIS IS NOT PESSIMISM**
>
> Frontier labs spend nine-figure budgets and employ hundreds of people to produce the open weights you can download for free. The reason Path A loses is not that you are not smart enough. It is that the open weights have absorbed so much of the world's compute already that it is irrational to redo that work.

§ So why would anyone pick Path A?

Three legitimate reasons:

To learn deeply.
Implementing attention from raw matrix multiplications, watching the loss curve descend on your own toy GPT, debugging a from-scratch LoRA — this is how you build the intuition that lets you debug Path B work. Every serious AI engineer should do this once.

To research a genuinely new idea.
If you have a novel architecture (a different attention mechanism, a different positional encoding, a non-transformer alternative), the only way to test it is from scratch. Open weights bake in particular architectural choices.

For sovereignty over a tiny, narrow model.
If you want a model that recognises Hebrew place-names from holocaust testimonies, or classifies Talmudic citation styles, no one will release pretrained weights for that. A small from-scratch model on your own data is the right answer.

§ The minimum-viable Path A learning week

Here is what we recommend for the "do Path A once" rite of passage. Block off seven days. Take Karpathy's *nanoGPT* or Sebastian Raschka's *Build a Large Language Model From Scratch* as your spine. By the end of the week you will have:

- A 10M-parameter character-level transformer trained on the Hebrew Bible (public domain, 1.2 MB), generating syntactically correct nonsense Hebrew
- A working understanding of: attention, residual streams, layer norm, the autograd graph, the optimiser inner loop, the training/validation split, gradient accumulation, mixed precision
- The same intuition that makes you, on day eight when you switch to Path B, immediately understand why a LoRA rank of 16 is different from a rank of 64

That week of investment is the highest-leverage learning in the entire field. It is not a substitute for Path B. It is the foundation that makes Path B not feel like magic.

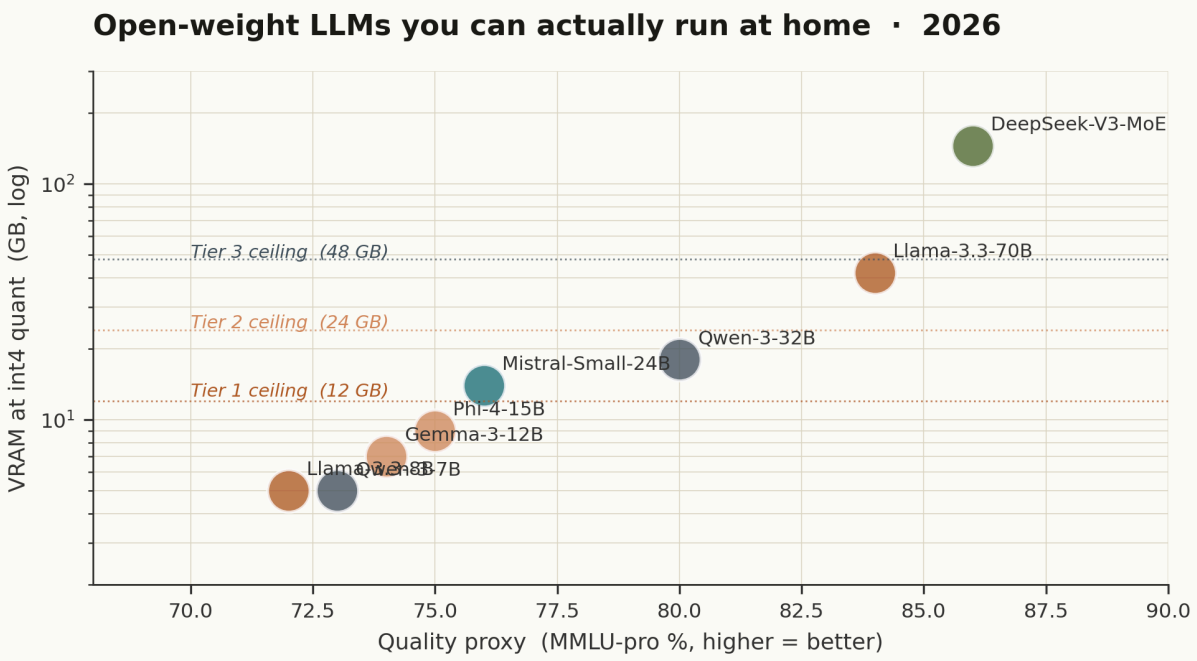


FIGURE 3.1 The reason Path A loses on production work: every dot here represents tens of millions of dollars of compute that you would otherwise be re-spending. Quality proxy is approximate MMLU-pro percentage; VRAM is at int4 quantisation. Sources: official model cards on Hugging Face, lmsys.org leaderboard snapshots Q1 2026.

CHAPTER 4

Path B — the guitar teacher, by fusion

What it actually looks like to ship the same project by combining open weights. Concrete components, real toolchains, ten-weekend timeline.

Now the honest version. Here is what an indie developer in Tel Aviv would actually build over the next two months to ship version one of the Hebrew guitar teacher. Every component named here is real, free, and downloadable today.

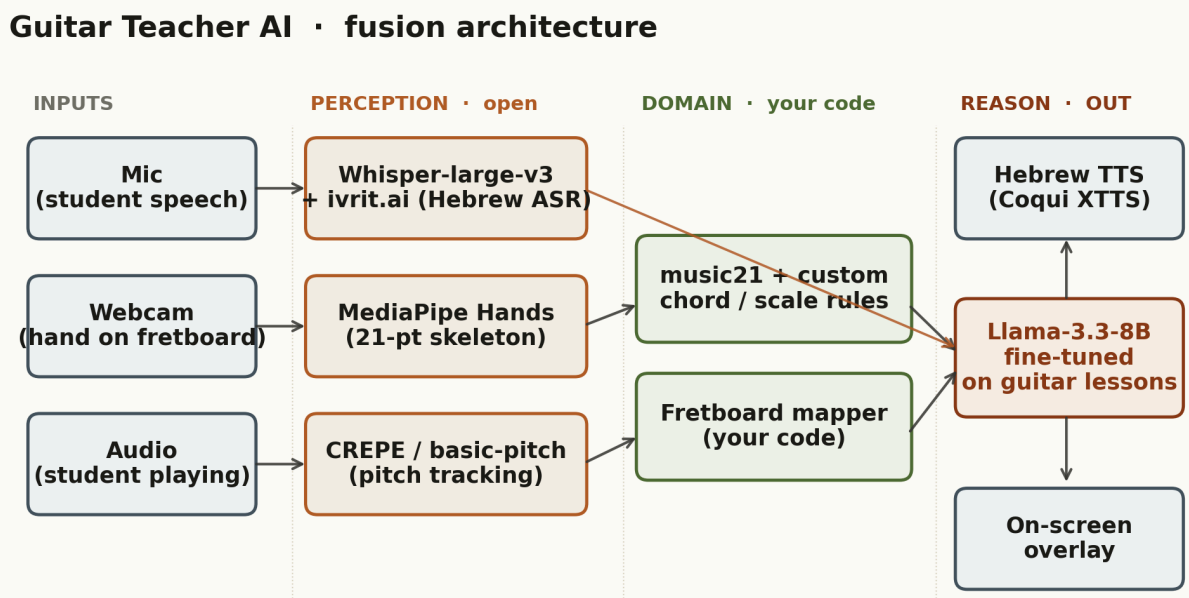


FIGURE 4.1 The guitar-teacher fusion stack. Three input streams (mic, webcam, audio) flow through three open-weight perception models, then through your own domain code (the only part you actually have to write), then into a fine-tuned Llama-3.3 for pedagogical reasoning, and back out as Hebrew speech and screen overlay.

§ The components, named

Working through the diagram from input to output:

OPEN-WEIGHT COMPONENTS FOR THE HEBREW GUITAR TEACHER, 2026

SUBSYSTEM	OPEN-WEIGHT MODEL	VRAM (INT8)	WHERE IT LIVES
Hebrew speech recognition	Whisper-large-v3 + ivrit.ai LoRA	3 GB	Hugging Face `ivrit-ai/whisper-large-v3-tuned`
Hand landmark detection	MediaPipe Hands (TFLite)	CPU only	`mediapipe` pip package
Pitch and chord tracking	basic-pitch (Spotify) + CREPE	2 GB	pip `basic-pitch`
Music theory engine	music21 (your code on top)	CPU only	pip `music21` + ~600 lines of yours
Pedagogical reasoning	Llama-3.3-8B-Instruct + LoRA	8 GB	Local via `ollama` or `llama.cpp`
Hebrew speech synthesis	Coqui XTTS-v2 (Hebrew fine-tune)	4 GB	Local; cloned voice optional

Total VRAM if everything is loaded simultaneously: roughly 17 GB at int8 — comfortably inside a single RTX 4090 (24 GB) or even a 4080 Super (16 GB) if you swap models in and out. On a 4070 (12 GB) you would run the LLM at int4 and unload Whisper between utterances.

§ What you actually write yourself

This is the part Path-B sceptics underestimate. Yes, the heavy weights came from elsewhere. But the system is still *yours*, and these are the parts where you spend your time:

1. **The fretboard mapper.** Take MediaPipe's 21 hand landmarks, calibrate against a one-time fretboard photo, and produce `{string: 1-6, fret: 1-19, finger: index/middle/ring/pinky}` for each frame. ~400 lines of Python.
2. **The music-theory bridge.** Combine the chord/pitch output from basic-pitch with the fretboard map to produce a structured *"the student is trying to play E-minor at the seventh fret with their pinky on the wrong string"* JSON event. ~600 lines using `music21`.
3. **The pedagogical prompt template.** Convert that JSON event into a system-prompted Hebrew query to the LLM: *"You are a patient guitar teacher. The student just attempted <event>. Their hand position was <coords>. Reply in friendly Hebrew, two sentences, with one specific tip."* ~150 lines plus a YAML of teaching styles.
4. **The orchestration loop.** Twenty-frames-per-second video in, audio chunks at 100ms, debounce, route, render overlay, queue TTS. ~800 lines of asyncio.
5. **The LoRA fine-tune.** Fine-tune Llama-3.3-8B on 5,000 hand-curated Hebrew teacher-student dialogues from your own lesson recordings. Six hours on a 4090 using Unsloth, 200 MB of LoRA weights at the end. This is what makes the model sound like a Hebrew guitar teacher and not a generic chatbot.

Total code you write: roughly 2,000 lines, plus a 5,000-example fine-tune dataset that you spent two weekends preparing. The result is yours, runs entirely on your own hardware, never sends a byte to anyone, and is competitive with anything a frontier lab could ship at the same cost.

§ Toolchains worth knowing in 2026

Unsloth
: The fastest LoRA / QLoRA fine-tuning library on consumer GPUs in 2026. Roughly 2x faster than vanilla Hugging Face `peft` for the same recipe, and includes good defaults for Llama, Qwen, Gemma, Mistral.

Axolotl
: Heavier-weight fine-tuning framework, YAML-driven, more configurable than Unsloth. Use this when you outgrow Unsloth's defaults — usually around the time you start doing full-precision fine-tunes on more than one GPU.

llama.cpp / Ollama
: The two ways most people actually run open-weight LLMs locally in 2026. Ollama is a friendlier wrapper; llama.cpp is the engine. Either one runs Llama-3.3, Qwen-3, Gemma-3 with one-line install.

vLLM
: Higher-throughput inference server for when one user becomes ten. Not for a single hobbyist; useful when the guitar teacher gets twenty concurrent students.

Hugging Face `transformers` + `peft`
: Still the lingua franca for everything model-loading. You'll touch this no matter which higher-level toolchain you use.

LangGraph or plain asyncio
: For orchestration. Honestly, plain asyncio + a handful of typed dataclasses beats most "agent frameworks" for a project of this size. Reach for LangGraph only when the state machine has more than ten nodes.

§ The honest timeline

Working solo, evenings and weekends, on a Tier-2 rig, building only this project:

10	~₪200	2,000	0
weekends to a working v1 demo	total spend, mostly electricity	lines of your own code	bytes sent to any cloud

This is not a thought experiment. This is what builders are actually shipping in 2026. The frontier-lab work has democratised so much that the bottleneck has moved upward — from *can the model exist* to *can you find a worthwhile use of the model*. The guitar teacher is one such use.

PART TWO

II

The hardware you'd actually buy

> *"Most hobbyists overestimate their utilisation by five to ten times. Track your real GPU minutes for one month before committing to Tier 3."*
>
> FROM CHAPTER SIX

CHAPTER 5

Three home-lab tiers, with real Israel prices

Bootstrap, indie lab, serious. What each one can do, what each one costs landed in Israel, what each one can't.

All prices are 2026, in NIS, landed in Israel — after 17% VAT, customs, and shipping. Sourced from KSP, Bug, Plonter, Ivory and Amazon IL as of late April 2026 (single-line citation under each table). For US-import parts we add ~12% over US sticker, the empirical 2025–2026 rate.

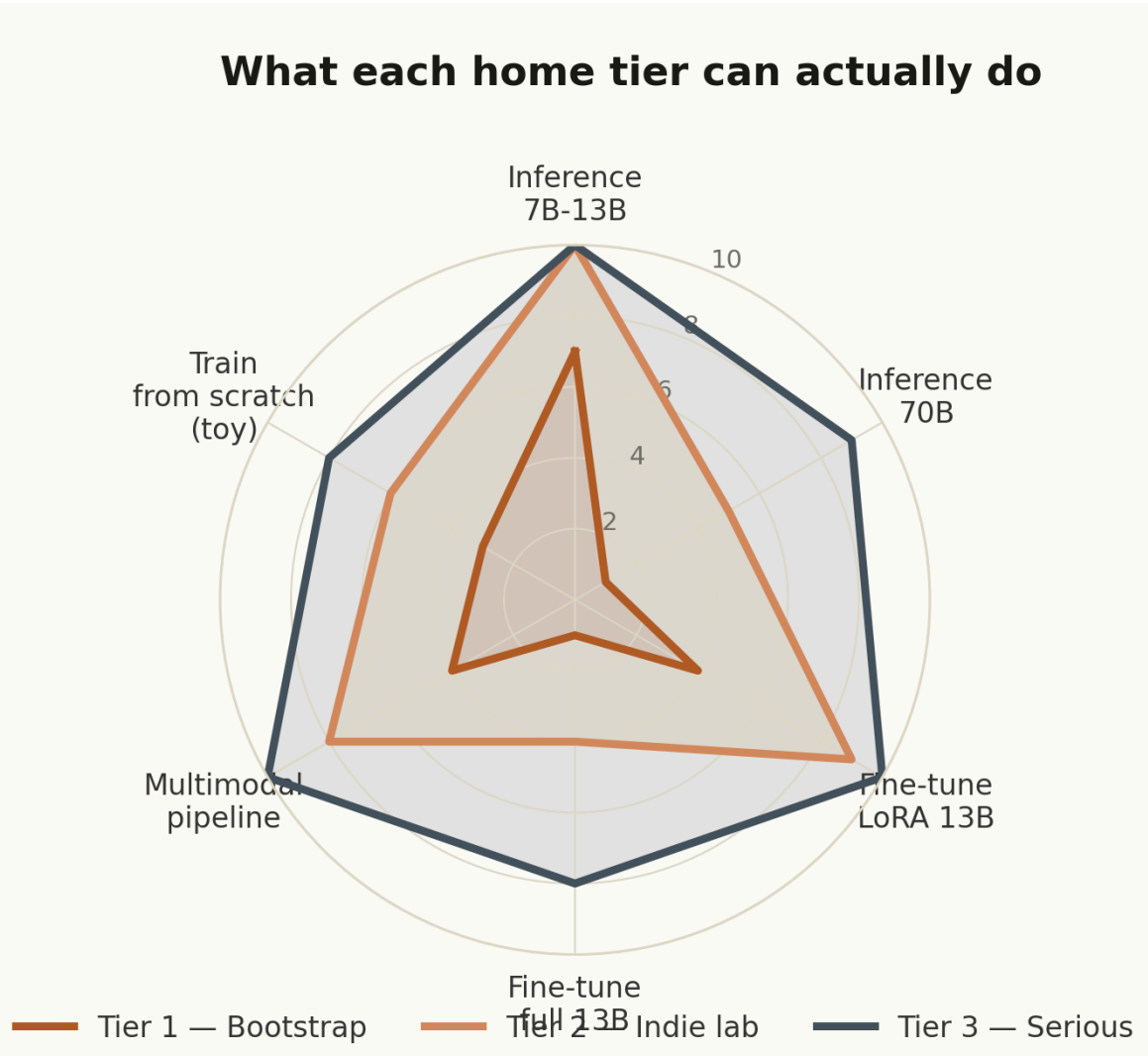


FIGURE 5.1 What each tier can actually do, scored 0–10. Notice how Tier 1 is competent at small-model inference but useless for fine-tuning anything serious; Tier 2 is the sweet spot for solo developers; Tier 3 starts to look like a small lab.

§ Tier 1 — Bootstrap (₪3–8k extra spend)

You already have a desktop PC — maybe a 2-year-old gaming rig. You add (or own) a current-generation 12–16 GB GPU. That is the entire AI capex. Most readers are already here without realising.

TIER 1 — BOOTSTRAP CONFIGURATION, ISRAEL-LANDED 2026 PRICES

COMPONENT	CHOICE	ISRAEL PRICE (NIS)	NOTES
GPU	RTX 4070 Super 12 GB	2,950–3,400	KSP/Bug, mid-range tier in 2026
GPU (alt)	RTX 4070 Ti Super 16 GB	3,800–4,400	The 16 GB makes a real difference for 13B at int4
RAM upgrade	64 GB DDR5-5600	700–900	If your existing build only had 32 GB
NVMe upgrade	2 TB Gen 4 NVMe	450–600	For model weights; you'll fill it
Total extra spend		**4,100–5,300**	Assumes the rest of the PC exists

Sources: ksp.co.il, bug.co.il, plonter.co.il listings sampled 28 April 2026. Prices include 17% VAT.

What Tier 1 actually runs

- Llama-3.3-8B at int8: smooth (~25 tok/s on the 4070 Super)
- Qwen-3-7B fine-tunes at int4: smooth
- Llama-3.3-70B: forget it. Maybe at int4 partial offload, painfully slow
- LoRA fine-tuning of 7B-class models: yes, in 6–10 hour overnight runs
- Whisper-large-v3 + Stable Diffusion 1.5 + LLM concurrently: tight but possible if you swap
- Pretraining anything from scratch: only toy models (<100M params)

> ◆ HONEST VERDICT ON TIER 1
>
> This is genuinely enough for the guitar teacher project end-to-end. It is also enough for 80% of indie AI projects in 2026. The reason to upgrade is not capability per se — it is iteration speed. A fine-tune that takes 8 hours on a 4070 Super takes 90 minutes on a 4090, and that compounds across a hundred experiments.

§ Tier 2 — Indie lab (₪16–32k all-in)

A dedicated AI workstation, not gaming-that-also-does-AI. The 2026 decision: 4090 vs 5090 — the 5090 has 32 GB GDDR7 and ~1.7x throughput, but costs nearly 2x landed in Israel.

TIER 2 — INDIE LAB CONFIGURATION, ISRAEL-LANDED 2026 PRICES

COMPONENT	CHOICE	ISRAEL PRICE (NIS)	NOTES
GPU (primary)	RTX 4090 24 GB	7,800–9,200	2026 stock from KSP, Plonter; gaming card supply restored
GPU (premium alt)	RTX 5090 32 GB	12,500–15,000	2025 launch; Israel availability stabilised by Q1 2026
CPU	AMD Ryzen 9 9900X (12c)	1,650–1,900	Or Intel Core Ultra 9 285K, similar
Motherboard	X870 (PCIe 5.0, 2× M.2)	1,150–1,500	Headroom for second GPU later
RAM	128 GB DDR5-5600 (4×32)	1,800–2,200	Mandatory for any model-merging or full-precision work
NVMe	2× 4 TB Gen 4 NVMe	1,400–1,800	One for OS+models, one for datasets
PSU	1000 W Platinum	650–850	Headroom for second GPU
Case + cooling	Fractal Torrent / Lian Li, AIO 360 mm	1,400–1,700	Airflow matters more than aesthetics
Total (4090 build)		**15,850–19,150**	Conservative bracket
Total (5090 build)		**20,550–24,950**	Premium bracket

Sources: ksp.co.il, plonter.co.il, bug.co.il, ivory.co.il listings sampled 28 April 2026; cross-checked with TechSpot 2025 launch coverage and zap.co.il price-history. Prices include 17% VAT.

What Tier 2 actually runs

- Any 7–13B model at fp16, smoothly
- Llama-3.3-70B at int4: ~12–18 tok/s on the 4090, ~25 on the 5090. Usable.
- QLoRA fine-tunes of 13B in under 2 hours
- Full-precision LoRA of 7B in under 1 hour
- Real multimodal pipelines: Whisper + LLM + TTS + vision all loaded concurrently with room to spare
- Pretraining: still only toys, but bigger toys (300–500M params)

◆ HONEST VERDICT ON TIER 2

This is the right tier for "I am taking AI seriously as my craft and want one machine that does not bottleneck me for the next three years." The 4090 build is the rational pick; the 5090 is for people whose time is worth the marginal speed. Neither one will pretrain anything frontier-quality, and both will run any open-weight LLM short of the 405B-class.

§ Tier 3 — Serious (₪70–200k)

Past hobbyist territory. Pretrain meaningful small models, fine-tune 70B in a day, run multimodal pipelines for paying customers. Two paths: dual consumer GPUs (cheaper, complicated cooling, no NVLink so model parallelism is harder), or single workstation/data-centre card (expensive, drops in cleanly, well-supported).

TIER 3 — SERIOUS CONFIGURATIONS, ISRAEL-LANDED 2026 PRICES

COMPONENT	BUILD A: DUAL 4090	BUILD B: RTX 6000 ADA / A6000	BUILD C: H100 PCIE (USED)
GPU(s)	2× RTX 4090 (~₪16k)	1× RTX 6000 Ada 48 GB (~₪28k)	1× H100 80 GB used (~₪95k)
VRAM total	48 GB (2×24)	48 GB	80 GB
NVLink	No (PCIe only)	n/a (single)	n/a
Workstation board (TR / Xeon)	~₪7,000	~₪7,000	~₪9,000
RAM 256 GB ECC DDR5	~₪7,500	~₪7,500	~₪7,500
NVMe (8 TB total)	~₪3,000	~₪3,000	~₪3,000
PSU 1600 W	~₪1,400	~₪1,200	~₪1,400
Server case + cooling	~₪3,500	~₪2,500	~₪4,000
Total estimate	**~₪38–45k**	**~₪49–55k**	**~₪120–135k**

Build A & B prices sourced from ksp.co.il, plonter.co.il and Israeli system integrators (BlackWidow, Buy.co.il) sampled 28–29 April 2026. Build C: H100 80 GB PCIe is not stocked in Israel for end users; the figure represents a personal import from US server-resellers (vendors such as ServerMonkey, Bargain Hardware) including ~12% landed-in-Israel uplift for shipping and customs. *[2026 range, based on 2025–2026 trends — H100 used market is volatile; verify within 30 days of purchase.]*

What Tier 3 actually runs

- Llama-3.3-70B at fp16: comfortable (build B/C); painful but possible (build A)
- Full-precision fine-tune of 70B in 1–3 days
- Pretraining of 1–3B-parameter models from scratch in 1–4 weeks — a real research workflow
- Multimodal production pipelines for 5–50 concurrent users

- Holding the entire DeepSeek-V3 MoE in int4 across two 4090s with offload (build A)

> **! THE UNSEXY REALITIES OF TIER 3**
>
> Sustained 1.6 kW draw for hours during fine-tunes. Your home electrical circuit may need attention — a single 16-amp Israeli circuit handles ~3.5 kW total, and your AC + workstation + monitor can saturate it. Heat: a Tier 3 box in a closed room raises ambient by 4–6 °C in summer. Noise: 50+ dB under load — not a desk machine, a closet machine. None of this is a dealbreaker, but plan for it before you spend ₪50k.

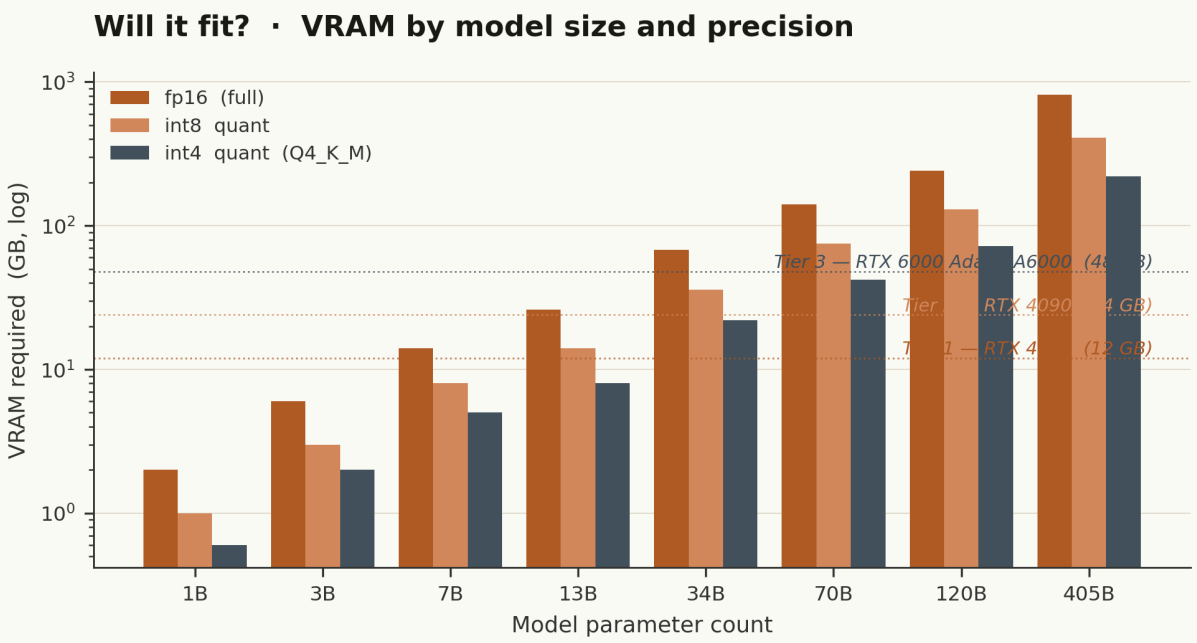


FIGURE 5.2 The deciding factor for "will it fit?" Each model size has three bars (fp16, int8, int4). Horizontal lines show the three tiers' VRAM ceilings. A 70B at int4 just clears 24 GB — this is why the 4090 is the sweet spot. A 405B model is out of scope for any single-GPU home build.

CHAPTER 6

Cloud, honestly

Six providers, real per-hour rates, the break-even hours/month for every home tier. The math that decides "buy or rent."

The instinct that cloud is "expensive" or "for big companies" is wrong both ways in 2026. For low-utilisation users, cloud is dramatically cheaper than owning. For high-utilisation users, owning pays back fast. The question is where the crossover is for *you*.

§ The 2026 menu of GPU rentals

CLOUD GPU RATES, SAMPLED LATE APRIL 2026

PROVIDER	GPU	VRAM	USD / HR	NIS / HR	NOTES
RunPod (community)	RTX 4090	24 GB	$0.34	₪1.26	Cheapest at this tier; spot-style, can interrupt
RunPod (secure)	RTX 4090	24 GB	$0.69	₪2.55	Stable instance
RunPod	A100 80 GB	80 GB	$1.64	₪6.07	Workhorse for 70B fine-tunes
Lambda Labs	A100 80 GB	80 GB	$1.29	₪4.77	On-demand; cheapest reputable A100
Lambda Labs	H100 80 GB	80 GB	$2.49	₪9.21	For when you really need it fast
Lambda Labs	8×H100 cluster	640 GB	$22.40	₪82.88	Multi-node; hourly
Vast.ai	RTX 4090 (peer)	24 GB	$0.28–0.50	₪1.04–1.85	Variable; check seller score
Together AI	Llama-3.3-70B inference	n/a	$0.88 / 1M tok	₪3.26 / 1M tok	Per-token, not per-hour
Anthropic API	Claude Opus 4.7	n/a	$15 in / $75 out / 1M tok	₪55 / ₪278	Frontier; not self-hosted equivalent
GCP Vertex AI	A100 40 GB	40 GB	$3.67	₪13.58	Hyperscaler premium
AWS SageMaker	A100 40 GB (p4d)	40 GB	$4.10	₪15.17	Hyperscaler premium

Sources: runpod.io, lambdalabs.com, vast.ai, together.ai, anthropic.com, cloud.google.com, aws.amazon.com pricing pages, accessed 28 April 2026. Conversion at USD 1 = NIS 3.7. Hyperscaler rates exclude egress, storage and orchestration overhead.

§ Break-even — the chart that decides

The single most useful question is: *at how many GPU-hours per month does a home tier pay for itself versus renting an equivalent cloud GPU?* Below is the answer for the most common comparison — a Tier 2 home box (₪25k capex) versus a Lambda A100 ($1.29/hr, similar real-world throughput on most workloads).

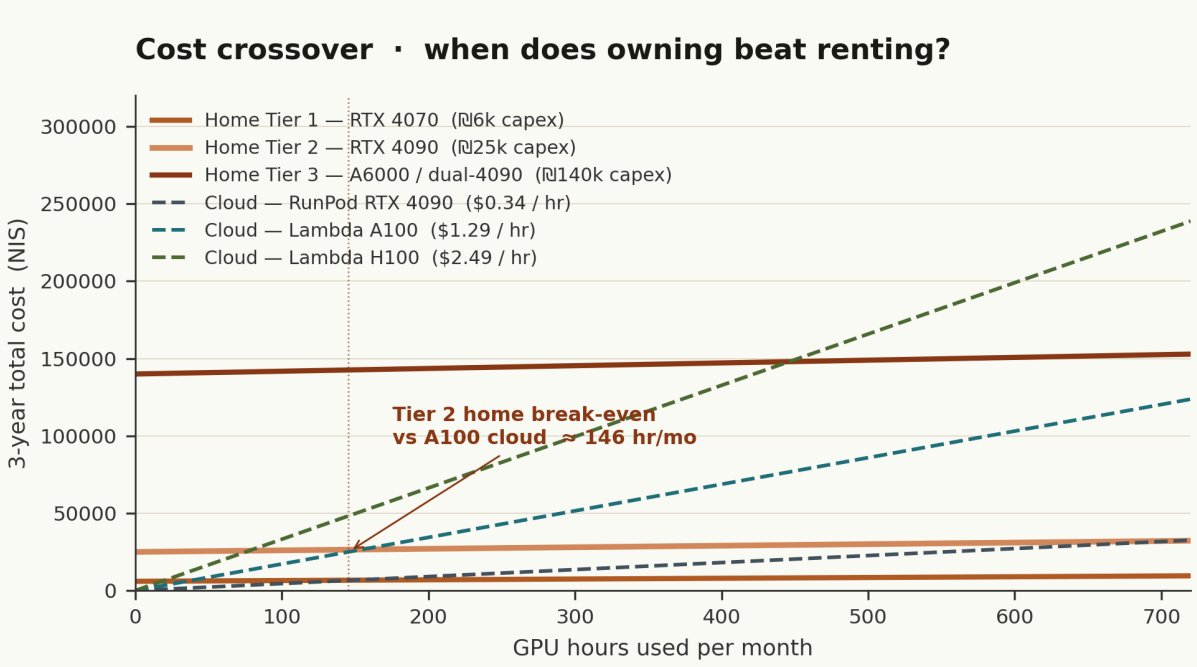


FIGURE 6.1 Three-year cumulative cost as a function of GPU hours used per month. The dotted line marks the Tier 2 home break-even versus Lambda A100 cloud. Below ~155 hr/month (about 5 hr/day), cloud is cheaper. Above it, owning wins — and the gap widens fast.

§ The break-even table you should print and keep

BREAK-EVEN GPU-HOURS PER MONTH — IF YOU'LL USE MORE, OWNING IS CHEAPER (3-YR HORIZON)

HOME TIER	VS RUNPOD 4090 ($0.34/HR)	VS LAMBDA A100 ($1.29/HR)	VS LAMBDA H100 ($2.49/HR)
Tier 1 (₪6k all-in)	133 hr/mo	36 hr/mo	19 hr/mo
Tier 2 (₪25k all-in)	555 hr/mo	~155 hr/mo	79 hr/mo
Tier 3 (₪140k all-in)	3,100 hr/mo	~840 hr/mo	440 hr/mo

3-year horizon, electricity at ₪0.62/kWh Israeli residential rate, GPU drawing average rated TDP under load. Cloud rates locked at April 2026 levels (will likely fall, narrowing the home advantage). 24×30 = 720 hr/month is the absolute ceiling.

How to read this

If you will use a GPU more than 5 hours a day on average, a Tier 2 home build pays back versus a Lambda A100 within three years. If you will use it less than 1 hour a day, you should not buy it — rent. The Tier 3 numbers look intimidating, and they should: a single H100 in your closet only beats Lambda H100 cloud rental at ~14 hours a day of sustained use, which is approximately what a small startup running a paid product needs.

◆ THE MOST USEFUL OPERATIONAL FACT

Most hobbyists overestimate their utilisation by 5–10x. They picture themselves "training all weekend" but actually use the GPU for 4 hours on a Saturday and then nothing for two weeks. **Track your real GPU minutes for one month before committing to Tier 3.** `nvidia-smi --query-gpu=utilization.gpu --format=csv -l 60 >> util.log` — that one line, run continuously, will tell you whether you are a 30-hour-per-month user or a 300-hour-per-month user.

§ The hybrid that almost everyone ends up with

Few real builders are pure-cloud or pure-home. The honest 2026 pattern is:

- **Inference and small experiments at home** (Tier 1 or Tier 2). Always-on, low marginal cost, your data stays local.
- **Big training runs in the cloud** (RunPod, Lambda). Spin up an 8×A100 cluster for a 12-hour run, pay $200, shut it down. Your home box would have taken three weeks for the same job.
- **Frontier API calls for things only the frontier can do** (Anthropic, OpenAI). When you need GPT-class reasoning for a one-off task, paying $15 for a million input tokens beats spending a month trying to match it locally.

The hybrid is not a compromise — it is the optimal allocation of your money and time. The mistake is religious commitment to one side.

CHAPTER 7

The hidden costs of home

Electricity, depreciation, your time, the failure modes nobody warns you about. The full three-year picture.

"Home GPU is cheaper than cloud" is true in narrow ways and misleading in broader ones. Capex is the headline; three other costs follow you for three years and most online comparisons ignore them.

§ Electricity, in Israel

IEC residential rate, January 2026: **₪0.62/kWh** regular, ~₪0.42 nighttime. A heavily-used Tier 2 box draws ~600 W under load (450 W GPU + 150 W rest), idling at ~80 W. Training 4 hr/day on average:

~₪900/yr
Tier 2, 4 hr/day load

~₪1,800/yr
Tier 3, 8 hr/day load

~₪260/yr
Tier 1, 2 hr/day load

None of these numbers will change your decision on their own. But over three years, Tier 3 electricity is ₪5,500 — roughly the cost of an extra NVMe drive or two months of cloud rental. And it is paid in monthly bills you cannot un-incur.

§ Depreciation, honestly

The market value of a GPU at year three of ownership, based on 2024–2026 second-hand patterns:

RESIDUAL VALUE AT 36 MONTHS — RECENT NVIDIA CARDS

CARD	ORIGINAL ISRAEL PRICE	YEAR-3 SECOND-HAND PRICE	EFFECTIVE DEPRECIATION
RTX 3090 (2020)	~₪7,000 launch	~₪2,800 in 2023	60%
RTX 4090 (2022)	~₪10,500 launch	~₪5,500 in 2025	48%
RTX 4070 (2023)	~₪3,400 launch	~₪2,000 in 2026	41%

Sources: zap.co.il price-history archives, yad2.co.il / facebook-marketplace IL listings sampled across 2023–2026. Trends approximate.

The honest interpretation: a 4090 you buy today for ₪8k is worth roughly ₪4k in 2029. That ₪4k difference is your real GPU cost over three years — not the full sticker. Cloud comparisons that use the full sticker as "home capex" overstate the home cost.

§ Your time — the cost everyone forgets

Owning a workstation costs time. Specifically:

- **Initial build:** 4–8 hours assembly + OS + driver setup, plus a weekend of CUDA / Python / fine-tuning toolchain shakedown
- **Ongoing maintenance:** driver updates that break things (~3 incidents/year, ~2 hours each), thermal cleaning (1 hr / 6 months), the occasional component failure (~1 incident per 18 months, half a day to diagnose & replace)
- **Power management:** deciding whether to leave it on overnight, configuring sleep, the inevitable 3am realisation that you forgot to start a training run

If you value your time at ₪150/hour (a conservative figure for an Israeli software developer in 2026), the time cost of owning Tier 2 over three years is roughly **₪3,500** — about the same as an extra GPU. Cloud users do not pay this; they trade it for a different set of frictions (waiting for instances, tearing down resources you forgot, surprise egress bills).

§ The full 3-year TCO chart

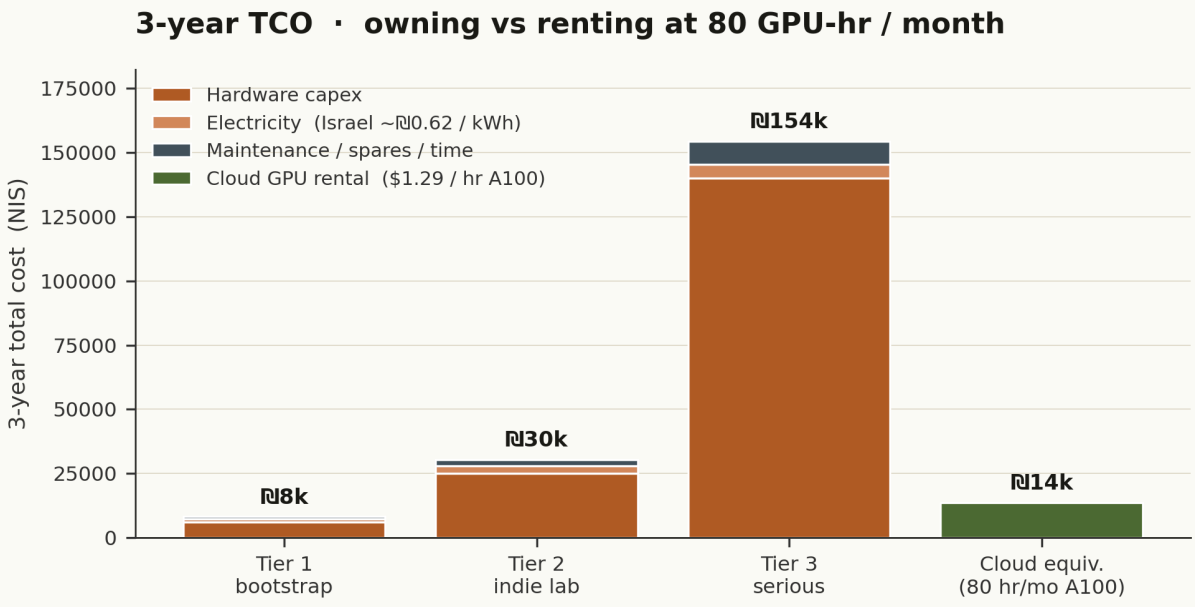


FIGURE 7.1 Three-year total cost of ownership, all components included, for a moderate-use scenario (80 GPU-hours/month, roughly 2.7 hr/day average). At this utilisation, Tier 1 home is the clear winner; Tier 3 is overkill; cloud is competitive only because the rate assumed is the cheap A100 ($1.29/hr). Switch the cloud line to a hyperscaler ($3.67/hr GCP) and the cloud bar becomes ₪32k.

§ What the cloud crowd undercounts

Now the other side. Cloud users have their own hidden costs that on-prem evangelists rarely mention:

Egress charges.
Hyperscalers charge ₪0.30–0.50 per GB to move data *out* of their cloud. If you train a model and want the weights on your laptop — that's 15 GB of egress for a 7B fine-tune, or about ₪7. Tiny per occurrence; large at scale, and the bill arrives a month later.

Spin-up time.
Cold-starting a new RunPod instance takes 3–6 minutes; a fresh AWS p4d, 8–12 minutes including AMI pull. For ten experiments a day, that is an hour of your life staring at "Provisioning..."

Forgotten resources.
Every cloud user has at least one war story about a forgotten GPU instance running for a weekend at $5/hour. That's $240. The home alternative is your electricity bill complaining mildly.

Lock-in by data gravity.
Once you have 200 GB of training data, model checkpoints and logs sitting in S3, moving providers is genuinely painful. You have not chosen freely on day 365 even if you chose freely on day 1.

Compliance and data sovereignty.
If your project involves Israeli citizens' personal data, GDPR-equivalent Israeli privacy law, or anything you cannot legally store outside Israel, large parts of the cloud menu are off-limits. (See chapter 8.)

§ The three-year answer

For a single developer doing the guitar-teacher style of project — mostly inference, occasional fine-tuning, ~50–100 GPU-hr/month — the honest ranking is:

1. **Tier 1 home** (~₪10k all-in over 3 years, electricity included). Best total cost. Fully sovereign.
2. **Tier 2 home** (~₪31k over 3 years). Better iteration speed; pays back if you'll do many fine-tune experiments.
3. **Cloud-only** (~₪14k at 80 hr/mo on RunPod 4090). Surprisingly cheap if your usage is low and bursty; loses on data sovereignty.
4. **Hybrid** (~₪14k home Tier 1 + ~₪3k/yr selective cloud bursts). The honest answer for almost everyone.
5. **Tier 3 home**. Only if you are running paid product workloads or doing real research at home.

If you remember one thing: track your actual GPU utilisation for a month before you spend anything above Tier 1. That single data point is worth more than every benchmark blog you'll read.

PART THREE

III

What you actually own when you're done

> *"The real cost of cloud is not the bill — it is what the bill prevents you from doing."*
>
> FROM CHAPTER EIGHT

CHAPTER 8

What "your data" actually means

Sovereignty, privacy, lock-in. The honest framing — not "the cloud is evil" but "here is what each option costs you in data terms."

This chapter exists because the discourse around AI privacy is unusually bad. One camp says "the cloud is fine, you have nothing to hide." The other camp says "any cloud use is data colonialism." Both are missing the actual question, which is more boring and more useful: *what specifically happens to your data in each option, and is that something you care about?*

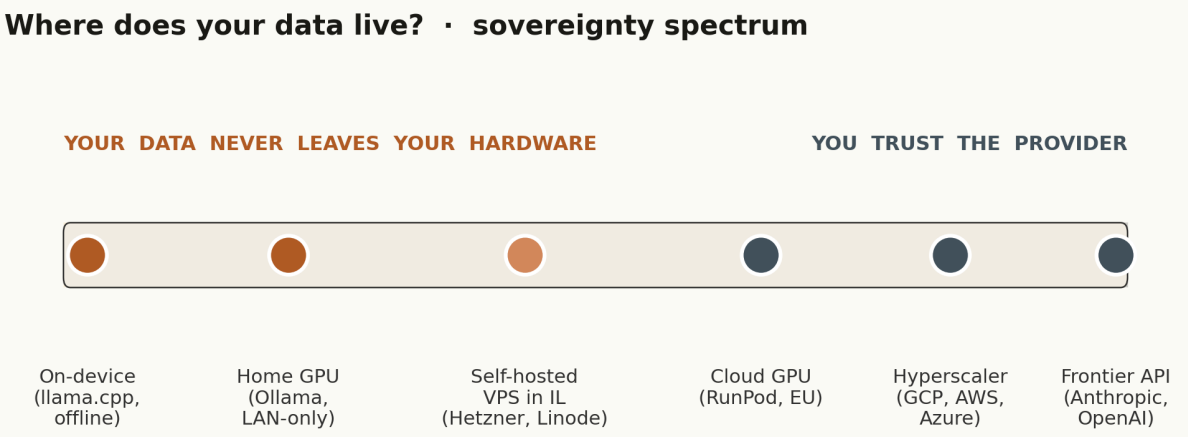


FIGURE 8.1 The honest spectrum. Each dot represents a real option for running an AI workload in 2026. Moving rightward gives you better quality and easier setup; moving leftward gives you more control over where your data sits.

§ The six honest options

1. On-device, fully offline (e.g. `llama.cpp` on a laptop with no network)

Your data never leaves the machine you typed it on. The model never sees the internet during inference. Even if you are wrong about a security setting, there is no attack surface. **Tradeoff:** you are limited to whatever fits in your laptop's RAM, and you are responsible for every update.

2. Home GPU, LAN-only (Ollama running on your workstation, accessed from your other devices over WiFi)

Same sovereignty as option 1, with better hardware. The model is local; your prompts cross your home WiFi but never your modem. **Tradeoff:** you have to maintain the box.

3. Self-hosted on a VPS in Israel (Hetzner Falkenstein has Israel-jurisdiction options; Linode Tel Aviv POP)

Your data is on a machine you exclusively rent, in a jurisdiction you trust, on a provider's hardware. Better uptime than home; less sovereignty (the host can in principle read disk if compelled by court order). **Tradeoff:** ₪200–800/month for a usable GPU VPS, often less GPU per shekel than home.

4. Cloud GPU rental (RunPod, Lambda, Vast)

You spin up a Linux box with a GPU, do your work, tear it down. Your data sits on shared infrastructure; the provider's policies say they don't look but they technically can. The model weights you upload are on someone else's disk for the duration. **Tradeoff:** excellent cost per GPU-hour, real but bounded data exposure.

5. Hyperscaler managed AI (GCP Vertex, AWS SageMaker, Azure ML)

Same as option 4 plus deep integration with the rest of the hyperscaler ecosystem and stronger SLAs — and stronger lock-in. Your data is governed by the hyperscaler's terms, which are extensive and change. **Tradeoff:** 3–5x more expensive per GPU-hour, much better tooling, large vendor lock-in.

6. Frontier API (Anthropic, OpenAI)

You send your prompt, you get a completion. The frontier lab sees every prompt. They have written policies (Anthropic and OpenAI both default to *not* training on API data, but the policies are policies, not laws). **Tradeoff:** you get the best models in the world, you get a per-token bill, and you're trusting a US company with your prompts.

§ Where Iddo's data lives, in particular

This guide is for a builder who cares about specific things — family photos of his late sister, voice recordings of family members, Hebrew-language content created by people whose consent he has but who never agreed to be in OpenAI's training set, religious texts in personal annotations. For data of that character, the right answer is rarely "the cheapest option." It is closer to "the option where the answer to *'who else can in principle see this?'* is shortest."

> ◆ THE HONEST SOVEREIGNTY HEURISTIC
>
> If the data on your hard drive is something you would not want to read in a leaked corporate database five years from now, do the work locally. If the data is something you'd happily post on a public blog tomorrow, the cheapest cloud option is fine. Most projects have *both* kinds of data — do the sensitive parts locally, the rest in the cloud.

§ The hybrid sovereignty pattern

For the guitar teacher, the natural split is:

- **Local only:** the student's voice recordings, video, and any biometrics from the webcam. These never leave the home machine. Whisper and MediaPipe both run locally.
- **Local only:** the LoRA fine-tune dataset of your own teacher-student dialogues. This is your IP; cloud upload would be foolish.
- **Cloud OK:** the initial Llama-3.3-8B base model download (it's public, you're just fetching). The Unsloth library updates. The python packages.
- **Cloud OK:** aggregated, anonymised metrics ("this lesson took N seconds, M chord errors detected"). Useful for product improvement, not personally identifying.

- **Cloud burst, sometimes:** if you ever do a full-precision fine-tune at 70B scale, you will rent an A100 cluster for a day. The dataset you upload should be sanitised first. The trained weights you download are then yours forever — the cloud only ever held them in transit.

§ What lock-in looks like in practice

The real cost of cloud is not the bill — it is what the bill prevents you from doing.

If you build the guitar teacher entirely on Anthropic's API, the project is dead the day Anthropic deprecates the model you built it on, or changes the pricing, or restricts the use case. (All three have happened to someone you know in the past 24 months.) If you build it on Llama-3.3-8B local + 200 MB of LoRA you trained yourself, your project is alive for as long as your hard drive is alive. That difference is hard to put a price on, but it is real and it should weigh in the decision.

Conversely, if you religiously refuse cloud entirely, you spend three months building Hebrew TTS from scratch when an XTTS-v2 fine-tune from a one-day RunPod run would have done it in an afternoon. The honest ledger has costs on both sides.

CHAPTER 9

Recommended starting kit, for someone in your position

One opinionated answer. The hardware, the software, the first ten weekends of work.

Compressed advice for the reader who has read the previous eight chapters and now wants to be told what to do. This chapter is opinionated. There are other valid answers. This is the one we'd give a friend.

§ If you have a recent gaming PC: stay there

Add a 16 GB GPU (RTX 4070 Ti Super, ₪3.8–4.4k) and 64 GB of RAM if you don't have it. Total spend ₪4–5k. This is enough for the entire guitar teacher project, and enough for almost any indie AI work in 2026. Do not buy more hardware until you have used what you have.

§ If you have nothing and want to start cleanly: build Tier 2

RTX 4090 build at the conservative end of the bracket: ~₪16k. Add a Lambda Labs account ($0 to register, you pay only what you use) for occasional H100 bursts on jobs that don't fit. This combination handles 95% of what a serious indie builder will do for the next three years.

§ The first ten weekends — a syllabus

AN OPINIONATED TEN-WEEKEND CURRICULUM

WEEKEND	WHAT TO DO	WHAT YOU'LL HAVE AT THE END
1	Install Python 3.12, PyTorch 2.6, CUDA 12.6, Ollama. Run Llama-3.3-8B locally. Talk to it in Hebrew.	A working local LLM. Working environment.
2	Karpathy nanoGPT or Raschka chapter 1–3. Train a 10M-param transformer on Tanakh.	A toy LLM, your own. The intuition that matters.
3	Whisper-large-v3 + ivrit.ai, transcribe a long Hebrew podcast. Get word-error-rate honest.	A real perception module. Confidence in audio pipeline.
4	MediaPipe Hands tutorial. Calibrate against your guitar. Output finger-on-fret JSON.	The vision module of the guitar teacher.
5	basic-pitch + music21. Detect chords from your own recordings.	The audio analysis module.
6	Wire 3-5 together. End-to-end demo: webcam in, JSON event out.	The orchestration backbone.
7	Hand-curate 200 Hebrew teacher-student dialogues. Format for fine-tuning.	The training dataset.
8	Unsloth LoRA fine-tune of Llama-3.3-8B on the dataset. Two epochs, overnight.	A guitar-teacher-flavoured LLM.
9	Coqui XTTS-v2 with Hebrew. Wire to LLM output.	Spoken Hebrew responses.
10	Polish, error handling, simple Tk GUI window. Show it to one real student.	v1 product, runs entirely on your machine.

§ What to read along the way

- **Sebastian Raschka, *Build a Large Language Model From Scratch*** — the best single text for Path A intuition. Read first three chapters in weekend 2.
- **Andrej Karpathy's "neural networks: zero to hero" YouTube series** — free, deep, still the gold standard.
- **Hugging Face's *NLP Course*** — the practical bridge into Path B. Free.
- **Unsloth's GitHub README** — the actual fine-tuning recipe you'll use in weekend 8. Frequently updated; check the version against your installed Llama.
- **Lilian Weng's blog** (lilianweng.github.io) — for when you want to understand what you just used.

§ One last honest paragraph

The single biggest predictor of whether you ship the guitar teacher is not which GPU you bought. It is whether you do weekend 2 before weekend 8. Most people skip weekend 2 (Path A), go straight to weekend 8 (LoRA fine-tune), don't understand why their loss is exploding, give up. The week of toy-model work is what makes the rest of the syllabus tractable. Do not skip it.

Beyond that — the project is yours. The hardware is bought. The frontier labs have done the expensive part. Ten weekends from this Saturday, you can have something running on your desk that, ten years ago, would have required a research lab.

— Iddo, Tel Aviv, 2026

BIBLIOGRAPHY

Further reading

Sebastian Raschka · *Build a Large Language Model From Scratch*

Manning, 2024 — the best single text for Path A intuition.

Andrej Karpathy · *Neural Networks · Zero to Hero (YouTube)*

Free, deep, still the gold standard. youtube.com/karpathy

Hugging Face · *NLP Course*

The practical bridge into Path B. huggingface.co/learn/nlp-course

Unsloth team · *Unsloth GitHub README*

The actual fine-tuning recipe you'll use in weekend 8. github.com/unslothai/unsloth

Lilian Weng · *lilianweng.github.io*

For when you want to understand what you just used.

ivrit.ai project · *ivrit-ai/whisper-large-v3-tuned*

Hebrew ASR LoRA on Hugging Face. huggingface.co/ivrit-ai

Anthropic · *Claude Models Documentation*

Reference for the frontier-API column of the comparison. docs.anthropic.com

Colophon

This second edition of *Build Your Own AI 2026* was designed and typeset in May 2026 in Tel Aviv. The body is set in **EB Garamond**, an open-source revival by Georg Duffner of Claude Garamond's 1592 Egenolff–Berner specimen. Headings and figure captions are **Inter** by Rasmus Andersson. Code is set in **JetBrains Mono**. The Hebrew edition is set in **Frank Ruhl Libre** with display weights from the Heebo family.

The cover art is a stylised neural-network cross-section rendered programmatically with matplotlib (see `scripts/cover.py`). All in-text figures are matplotlib at 300 DPI (see `scripts/figs_premium.py`). Page assembly is done in raw HTML/CSS and rendered to PDF through Chromium's headless print engine.

Version 2 · May 2026 · Iddo, with Claude Opus 4.7. In loving memory of Alma ז״ל.