FREE SAMPLE — PREFACE + CHAPTERS 1–2

The biggest model wins the demo.
The smallest model wins the invoice.

Small Models Win

Why Fine-Tuned Small Models Outperform General Models

Yahya Laraki

September 2026 edition

Preface

Imagine a bank that classifies two million customer messages a month — complaints, fraud flags, wire questions. The team prototypes with the flagship model API. It works beautifully. Then finance sees the bill, engineering sees the latency at peak hours, and legal sees customer financial data flowing to a third party. The prototype was a triumph. The production system is a problem.

That bank is hypothetical, but the pattern is real, and I have watched versions of it play out across industries. When a company builds something with AI today, the first question is almost always the same: which model should we use? And the answer is almost always the same too: the biggest one. The newest flagship. The model with the most impressive demo.

This book argues that this default is wrong — expensively, measurably wrong — for a large and growing class of real work. On narrow, well-defined tasks, a smaller model that has been specialized — fine-tuned, distilled, or trained on carefully curated data — routinely beats the giant generalist. Not sometimes. Routinely. And it does so while costing a fraction to run, answering far faster, and keeping your data inside your own walls.

That is the bet this book makes. Let me state it precisely, because the sloppy version of this claim is false and the precise version is one of the most useful things you can know about AI in 2026.

The naive claim: *small models are better than big models.* That's not true. A 7-billion-parameter model will not out-reason the frontier on a task nobody prepared it for. Scale is real, and the frontier labs keep proving it. This book spends two full chapters (8 and 9) on exactly where scale wins, because the conditional thesis is only valuable if you know its borders.

The conditional claim — the one this book defends with evidence — is this: **a specialized small model outperforms a larger general model on narrow, well-defined, in-distribution tasks, when the specialization is done well, the training data is high quality, and the evaluation matches the task.** Every one of those qualifiers matters. Narrow. Well-defined. In-distribution. Done well. Each chapter that follows will show you exactly what they mean, with real numbers from real papers.

This book is written for two people at once, and I want to be explicit about the contract with each. If you're an engineer, you'll get mechanisms and numbers: how fine-tuning actually works, what LoRA and distillation change under the hood, and exact benchmark scores with their sources — every one of them traceable. You will not be asked to take anything on faith, and you will not find hand-waving where a number should be. If you're a founder, product leader, or executive, you'll get the business consequence of every idea: what it costs, how fast it answers, where it fails, what can go wrong in production, and how to decide between fine-tuning, prompting, retrieval, or just buying the API. Every technical idea is paired with the question you actually care about: so what should we do differently on Monday?

The register stays in the middle — technical ideas explained plainly, business ideas grounded in evidence. No heavy math anywhere in this book. Terms are defined the first time they appear, in plain language, and collected in a glossary at the back. If an analogy appears, it's because it earned its place.

What this book is not: it is not a training manual. You won't find step-by-step code for running fine-tuning jobs here — the tooling changes every quarter, and a printed recipe would be stale before the ink dried. Chapter 5 explains how the machinery works so you can evaluate claims and vendors; Chapter 10 gives you the decision framework. For implementation details, the cited papers and their code repositories are the right next stop.

A word on honesty, because this subject attracts hype. Every number in this book traces to a cited source — a paper, a company publication, or a public leaderboard. Papers that haven't been peer-reviewed are labeled as preprints. Company blog posts are the authors' own reported experiments, and I'll say so when I use them. Some results I found are promising but unverified; those are either omitted or labeled explicitly as unverified leads. To give you a concrete example of this discipline: a vendor later claimed a legal AI variant reached roughly 0.77 on a legal benchmark, described as "GPT-4 class." The claim is unverified, so it appears nowhere in this book's evidence. You'll see the verified 7B legal result it was based on — and you'll see me say plainly what it does and doesn't prove.

The research behind this book also surfaced results that cut *against* the thesis — a fine-tuned model collapsing outside its training distribution, a general model beating a medical specialist — and those are in the book too, because a

thesis that can't survive its own counterexamples isn't worth your time. Chapter 8 and Chapter 9 are devoted entirely to where the thesis fails and where raw scale still wins. I would rather you trust this book's limits than admire its wins.

One more discipline this book keeps, because most "small beats big" writing doesn't: there are three different mechanisms hiding inside that phrase, and they are not the same thing.

1. **Supervised fine-tuning.** Take an existing model and continue training it on examples of your task — your support tickets, your contracts, your financial headlines. The model's weights shift toward your distribution.
2. **Targeted distillation.** Use a big model as a teacher to generate training data — reasoning traces, synthetic examples, worked solutions — and train a small student on it. The small model inherits the teacher's behavior on the tasks you care about.
3. **Curated domain pretraining.** Train (or continue training) a small model from scratch on better, narrower data — textbook-quality text, a mountain of code, a curated math corpus. The model is small but its data was excellent.

These mechanisms overlap in practice, and I'll label which one each result uses. Conflating them is how people end up claiming "fine-tuning did this" when really "great data did this." The distinction matters because each mechanism has different costs, different data requirements, and different failure modes — and Chapter 10, the playbook, will ask you to choose between them.

A note on the word "win" in the title. Winning here means outperforming the larger general model *on the task* — higher accuracy, better F1, stronger benchmark scores — plus the economic advantages that come free with smallness: lower cost, lower latency, deployability on your own hardware, data privacy. It does not mean the small model is "better" in general. Nobody in this book claims a 7-billion-parameter model is smarter than the frontier. The claim is narrower and, I'd argue, more interesting: on the work that actually fills production systems, the specialist wins. The book's job is to show you exactly where that sentence is true, where it breaks, and how to tell the difference for your own use case.

Finally, a word about why this book exists now. The results it reports are mostly from 2023 to 2026 — the years when specialization stopped being a research-lab luxury and became something a small team could do in an afternoon. That shift is still underappreciated. Most of the industry's mental models about "build vs. buy" in AI were formed in the era when training anything meant a supercomputer. Those models are stale. The teams that have updated them — the ones running fine-tuned 7B models where their competitors rent flagship APIs — are quietly enjoying better margins, faster products, and data architectures their lawyers approve of. This book is my attempt to make that quiet advantage legible, with the receipts attached. Every claim you can check; every number has a source; every limit is stated. What you do with the advantage is up to you.

And one request, offered sincerely: argue with this book. Check a citation. Reproduce a comparison on your own data. Try to break the thesis in Chapter 8's terms and see if it holds. A book about evidence should be read evidentially — and if you find a result that changes the picture, that's not a failure of the book. That's the book working as designed.

How to read this book: Chapters 1 through 4 make the case with evidence — medicine, law, finance, code, math, reasoning, language, retrieval, speech. Each is self-contained enough to read on its own; together they form an argument no single result could carry. Chapters 5 and 6 explain *why* it works — the machinery of fine-tuning and the data story underneath it. These are the chapters that turn "interesting results" into a mental model you can apply to your own problems. Chapters 7 through 9 are the money and the limits — the economics of small models (Chapter 7 is the one your CFO can read alone), when fine-tuning fails, and where scale still wins. Chapter 10 turns everything into a decision framework you can actually use, including the cases where the right answer is "don't fine-tune."

A note on navigation for the two readers. If you're the engineer, you'll probably read front to back — the evidence chapters give you the results, the mechanism chapters give you the understanding, and the playbook gives you the plan. If you're the executive, here's a fast path with no shame in it: Chapter 1 for the thesis, Chapter 2 for the proof it works in serious domains, Chapter 7 for the money, Chapter 8's opening for what can go wrong, and Chapter 10 for the decision. The key takeaways at the end of every chapter are written to stand alone, so even a skim leaves you with the load-bearing facts.

If you only have an hour, read Chapters 1, 7, and 10. If you only have twenty minutes, read Chapter 1 and the key takeaways at the end of each chapter — they're written to stand alone. If you're the skeptical type, read Chapters 8 and 9 first; if the thesis survives your skepticism there, the rest of the book will read as the opportunity it is.

The evidence that follows changed how I think about building with AI. I hope it changes how you build too.

Key takeaways

- The default of "use the biggest model" is often wrong — expensively wrong — for narrow, well-defined tasks.
- The book's thesis is conditional, not universal: specialized small models beat larger generalists **on narrow, in-distribution tasks, with good data and task-aligned evaluation**.
- Three distinct mechanisms — supervised fine-tuning, targeted distillation, curated domain pretraining — are kept separate throughout.
- Every number is sourced; preprints are labeled; counterexamples are included, not hidden.
- Chapters 1–4 present evidence, 5–6 explain mechanisms, 7–9 cover economics and limits, 10 is the decision playbook.

Chapter 1: The Big Bet: Small Models Are Beating Giants

Every engineering team building with AI in the last few years has had the same meeting. Someone demos the newest flagship model — it writes a poem, debugs a script, explains a merger agreement, all in one session — and the room converges on the obvious conclusion: we'll just use the big one. The API is right there. It handles everything. Why would we do anything else?

That meeting is the subject of this book, because that conclusion is wrong more often than anyone in the room realizes.

The gravity of the biggest model

The pull toward the biggest model is understandable. A general-purpose frontier model is the most capable single artifact most teams have ever had access to. It really can do the poem and the script and the merger agreement. It needs no training, no data pipeline, no machine-learning expertise — just an API key and a prompt. For a prototype, it is unbeatable. For a demo, it is magic.

And demos are the problem. A demo shows you the model at its best, on tasks chosen to flatter it, judged by vibes. The presenter picked the examples. Nobody in the room ran the model's ten-thousandth answer through a grader. Psychologists would call this the availability heuristic wearing a tuxedo: the impressive thing you just saw crowds out the boring question of what the thing does on *your* data, at *your* volume, under *your* constraints. Call it the demo effect. The model that dazzles in a thirty-minute meeting is not necessarily the model that classifies your ten millionth support ticket correctly, cheaply, and in under a second.

There's a second force at work: the fear of leaving capability on the table. Nobody gets fired for choosing the biggest model. If the flagship fails, it's the model's fault; if a small fine-tuned model fails, it's *your* fault for being clever. So teams default to scale the way previous generations defaulted to the most expensive consultant — as insurance against blame, not as an engineering decision.

But insurance has a premium, and this one is steep. The biggest models are the most expensive to run per token, the slowest to answer, the hardest to keep private, and the least predictable to control. You are renting a genius to do a clerk's job, and paying genius rates for every token.

Consider an illustrative example. Imagine a company routing 500,000 customer support messages a month through a flagship API at a few dollars per million tokens. The monthly bill lands in the low five figures — noticeable but tolerable. Then the product grows 10×, as successful products do. The bill is now six figures a month, every month, forever, scaling linearly with success. Meanwhile a fine-tuned 8-billion-parameter model — which, as you'll see in this book, routinely *matches or beats* the flagship on classification-style tasks — would run that same workload for a small fraction of the cost, answer faster, and keep every message inside the company's own infrastructure. The team that chose the biggest model in that first meeting didn't make a technical decision. They signed a variable-rate mortgage on their own growth.

There's a third force, and it's worth naming because it shapes everything you'll read in the AI press: the companies selling you the models make more money when you use the biggest one. Every token billed at flagship rates is revenue. No API vendor has ever published a guide titled "you'd be better off with a smaller model" — while several have published benchmarks showing their flagship winning at everything. The information environment around model selection is not neutral. It is a sales funnel with the most expensive product at the bottom. This book is an attempt to give you the independent evaluation the vendors won't.

The specialist intuition

Now consider the rest of engineering — and the rest of human life. We do not hire a Nobel laureate to do our bookkeeping. We do not use a Formula 1 car to deliver groceries. Specialization beating generality on a narrow task is not a surprising claim; it is the default assumption of every mature field. The general practitioner refers you to the specialist. The Swiss Army knife loses to the chef's knife in every kitchen on earth.

Machine learning has known this for decades. A spam filter trained on spam beats a general text classifier at detecting spam. A fraud model trained on your

transactions beats a generic anomaly detector on your fraud. Nobody found this controversial, because the tasks were narrow and the models were small and the comparison was honest.

What changed is that large language models made generality *feel* free. One API call handles everything, so the specialist's advantage — which was always about the match between the model and the task distribution — got buried under convenience. This book is an excavation project. The specialist's advantage didn't go away. It got stronger, because now the specialist can start from a strong base model and be *taught* the specialty, rather than built from nothing.

There's a historical echo worth hearing. In the 2010s, the same debate played out in computer vision. For years, the default was the biggest ImageNet model you could afford to run — until EfficientNet showed in 2019 that a 5.3-million-parameter model could beat a 26-million-parameter ResNet on ImageNet top-1 accuracy (77.1% vs 76.0%) through better design rather than more scale (Tan & Le, 2019). The pattern — small wins via focus and craft, not via mass — predates large language models by years. LLMs just made the stakes enormously larger, because now the "big model" costs dollars per million tokens instead of milliseconds of GPU time.

That teaching has a name, or rather three names. You can fine-tune a model on examples of your task (supervised fine-tuning). You can have a big model generate training data for a small one (distillation). Or you can train a small model on superb, narrowly curated data from the start (domain pretraining). All three produce the same economic shape: a small, fast, cheap model that beats the giant on the task it was built for. You'll see all three in this book, carefully labeled, because they fail in different ways and cost different amounts.

Why this is happening now

A fair objection: if specialization is so powerful, why isn't everyone already doing it? The answer is that until recently, specialization was a rich lab's game. Fully fine-tuning a large model meant owning racks of GPUs and employing people who could keep a distributed training run alive. The economics only worked for the largest companies.

That changed with a pair of techniques you'll meet properly in Chapter 5: LoRA and QLoRA. The short version is that researchers figured out how to specialize a model by training a tiny fraction of its parameters — roughly 8 million out of 7 billion, about a tenth of one percent — while leaving the rest frozen. Suddenly a fine-tuning job that needed a data center fit on a single high-end GPU. The cost of producing a specialist collapsed by orders of magnitude, and the results in the following chapters — most of them from 2023 onward — are what that collapse made possible.

This timing matters for the thesis. The evidence in this book isn't a collection of historical curiosities; it's the output of a capability that became widely accessible only recently, and whose results are still compounding. The teams winning with small specialists today are early, not late.

The question this book answers

Strip away the hype and the entire book reduces to one practical question: **for a given piece of work, should you rent the giant or train the specialist?**

Most teams answer it once, early, by default — and then never revisit it. The flagship API becomes load-bearing infrastructure. Prompt engineering becomes a full-time job. The monthly bill becomes a fact of nature. Nobody re-runs the comparison that would have been run at the start if the team had known then what the literature now shows: that for narrow, high-volume, well-specified work, the specialist usually wins on quality and always wins on cost.

This book exists to make that comparison unavoidable. By the time you finish Chapter 4, you will have seen small specialists beat larger generalists in medicine, finance, law, code, math, reasoning, translation, retrieval, and speech — each result sourced, each caveat stated. By Chapter 7, you will have real prices and a worked cost comparison. By Chapter 10, you will have a decision framework that tells you, for your specific task, which path to take and what evidence would change your mind.

You don't have to accept the thesis in advance. In fact, I'd prefer you didn't — the skeptical reader gets more from this book than the convinced one, because the counterexamples in Chapters 2, 8, and 9 are doing half the argumentative work. Just bring one narrow task from your own world as you read, and keep

asking: would the specialist win here? By the end, you'll know how to answer that question with data instead of vibes. That's the whole game.

A tour of what's coming

The evidence in Chapters 2 through 4 comes from published results, each with its source, each with its caveats. A preview:

In medicine, a 7-billion-parameter model trained on medical textbooks and GPT-4-generated reasoning traces scored 74.3% on the MedQA benchmark — beating a 70-billion-parameter medical model that scored 70.2%. Ten times the parameters, worse at the specialty.

In finance, a 7-billion-parameter model instruction-tuned on 136,609 financial examples beat GPT-4 at financial sentiment classification — and also beat BloombergGPT, a 50-billion-parameter model pretrained specifically on finance. The specialist beat the generalist *and* the bigger specialist.

In code, a 1.3-billion-parameter model trained on textbook-quality synthetic data beat models ten times its size at Python code generation. A 15-billion-parameter code model beat a 540-billion-parameter generalist by a wide margin.

In reasoning, a 13-billion-parameter model taught reasoning strategies by GPT-4 beat a 70-billion-parameter chat model by over nine points on average across six benchmarks.

And these are not cherry-picked miracles. They are the predictable output of a mechanism: when you concentrate training on the task distribution, the model spends its capacity where it matters instead of spreading it across everything. A useful mental image: the generalist is a library containing every book ever written, and answering your question requires finding the right shelf. The specialist is a single well-organized manual for exactly your job. For the job the manual covers, the library's vastness isn't an advantage — it's search overhead. The specialist doesn't know more. It knows where things are.

Each of these results will get its full treatment — methods, numbers, caveats, business consequences — in the chapters ahead. The preview's job is just to calibrate your expectations: the thesis of this book is not a contrarian hot take. It's the summary of a literature.

The honest boundaries

A book that only showed wins would be a sales pitch, and this isn't one. The same research that found these victories found their boundaries, and you'll meet them early — Chapter 2 includes them alongside the wins, because they sharpen the thesis rather than weaken it.

The financial specialist that beat GPT-4 at sentiment classification collapsed at numerical finance questions in the very same paper — the fine-tuning won inside its training distribution and fell apart outside it. A general GPT-4, with no medical specialization at all, beat an early medical specialist at medical questions; specialization is not destiny, and the lead changes hands as methods improve. Distillation wins turn out to be distribution-dependent: the student beats the teacher on the distilled distribution and loses elsewhere.

These boundaries are the point. "Small beats big" is false. The true claim has conditions, and the conditions are knowable: narrow task, well-defined evaluation, in-distribution data, high-quality training examples. When those hold, the small specialist wins — on accuracy, on cost, on latency, on privacy. When they don't, the generalist's breadth reasserts itself. Chapters 8 and 9 are devoted to the failure modes, because knowing exactly where the thesis breaks is what makes it usable.

There's one more boundary worth stating early, because it protects you from the most common misreading of this book. The thesis is about *tasks*, not about *intelligence*. Nothing in the following chapters implies that a specialized 7B model is "smarter" than a frontier model, or that scale doesn't matter, or that the frontier labs are wasting their money. The frontier's generality is what makes the specialist possible — every fine-tuned model in this book starts from a base model that only exists because someone trained at scale. The relationship is symbiotic, not adversarial: scale creates the raw capability, specialization aims it. The book's argument is about where to spend your *next* dollar — and for narrow production work, the evidence says the aimed dollar beats the raw one.

The conditional thesis, stated precisely

Here is the claim this book defends, in its full form:

On narrow, well-defined, in-distribution tasks, a small model that has been specialized — by supervised fine-tuning, targeted distillation, or curated domain pretraining — outperforms a larger general-purpose model, provided the specialization uses high-quality data and the evaluation measures the actual task.

Read the qualifiers as a checklist, because Chapter 10 will turn them into one. To make each concrete, here's an illustrative example of a task that passes the full checklist — and one that fails it.

Passes: routing incoming customer emails into twelve fixed categories (billing, cancellation, technical issue, and so on) for a software company. Narrow — twelve buckets, nothing else. Well-defined — every email belongs in exactly one bucket, and misroutes are countable. In-distribution — next month's emails look like last month's. High-quality data — the company has three years of emails already labeled by its support team. Task-aligned evaluation — accuracy on a held-out sample of real emails. This is the kind of task where a fine-tuned small model should beat the flagship, and the literature says it will.

Fails: "help our analysts think through novel market situations." Not narrow, not well-defined, no stable distribution, no ground truth to train on, no way to evaluate except vibes. This is flagship territory — the generalist's breadth is the whole point. Anyone proposing to fine-tune a small model for this is buying trouble, and Chapter 8 will show you the wreckage of similar attempts.

The five qualifiers:

- **Narrow:** the task has a bounded scope — classify these headlines, answer these medical questions, extract these entities. Not "be smart about everything."
- **Well-defined:** you can say what a right answer looks like, and measure it. Benchmarks, graded evals, human review with rubrics.
- **In-distribution:** the inputs at deployment resemble the inputs in training. This is the qualifier that kills most careless fine-tuning projects.
- **High-quality data:** the specialization data is clean, correct, and representative. A thousand excellent examples beat a million noisy ones — you'll see the study in Chapter 6.
- **Task-aligned evaluation:** you're measuring the actual job, not a proxy that flatters someone. More on the many ways benchmarks lie in Chapter 8.

When the checklist holds, the economics follow automatically: the small model costs a fraction per token, answers several times faster, can run on your own hardware or even on a phone, and never sends your data to a third party. Chapter 7 puts real numbers on all of this.

The bet of this book is that a large fraction of production AI work — the unglamorous, high-volume, well-specified work that actually makes money — satisfies this checklist. If that's right, then the industry's default of reaching for the biggest model is not just suboptimal. It's the most expensive habit in software.

One final framing before the evidence begins. This book is structured as a case: the claim (this chapter), the evidence (Chapters 2–4), the explanation (Chapters 5–6), the economics and the limits (Chapters 7–9), and the verdict you can act on (Chapter 10). But it's also structured as a bet you can test yourself. Every result in the evidence chapters comes with its source, its benchmark, and its caveats — enough for a competent team to reproduce the comparison on their own data. If the thesis is right, the most valuable thing you can do after reading this book is also the cheapest: take one narrow, well-defined task in your own operation, fine-tune a small model on it, and measure. The book makes the case; your data delivers the verdict.

Key takeaways

- Teams default to the biggest model because of the demo effect and blame-avoidance — not because it's the best engineering choice for their task.
- Specialization beating generality on narrow tasks is the oldest pattern in engineering; LLMs buried it under the convenience of one API, but didn't erase it.
- Three mechanisms produce the specialist advantage: supervised fine-tuning, targeted distillation, and curated domain pretraining.
- The wins are real and published: small specialists beating models 10× their size in medicine, finance, code, and reasoning.
- The thesis is conditional — narrow task, well-defined eval, in-distribution data, high-quality examples — and the book is explicit about where it breaks.

Sources

- Mingxing Tan and Quoc V. Le, Google, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* (ICML 2019). https://arxiv.org/pdf/1905.11946
- Benchmark landscape and model references throughout this book draw on the September 2026 LLM benchmark survey in the book's research files, and on the individual papers cited in each chapter's Sources section.

Chapter 2: Medicine, Law, Finance: Specialists at Work

Professions are the natural habitat of the specialist. A doctor, a lawyer, and a financial analyst do narrow, high-stakes, well-defined work — exactly the conditions under which a specialized small model should beat a generalist. So the professional domains are where the thesis faces its first real test. The results are striking, and so are the failures. Both belong here.

How to read the numbers in this chapter

Before the results, a brief user's guide to the scoreboards, since this book asks you to trust numbers and you should know what they mean.

A **benchmark** is a fixed set of test questions with known right answers. **MedQA** uses USMLE-style medical questions — multiple choice, one right answer, scored as a percentage. **F1 score** is the standard accuracy measure for classification tasks: it balances catching the positives against avoiding false alarms, on a 0-to-1 scale where higher is better. When a paper reports "F1 0.86 vs 0.78," the first model is meaningfully better, not marginally.

Two honesty notes I'll repeat wherever they apply. First, most of the papers in this chapter are **preprints** — published on arXiv or OpenReview but not yet peer-reviewed. Their numbers are the authors' own reported experiments, and I treat them as evidence, not settled fact. Second, a benchmark score is only as good as the benchmark: a model can win on a test set that resembles its training data and lose in the real world. That caveat is not a footnote in this book — it's a recurring character, starting with the FinMA story below.

Medicine: the 7B model that beat the 70B medical model

Medical question answering is one of the hardest narrow tasks in AI — and one of the most consequential, because the same technology that answers exam questions is being piloted for clinical decision support, triage, and patient

communication. The MedQA benchmark uses questions in the style of the USMLE — the American medical licensing exam. Answering them requires real domain knowledge, not just fluent text. It's also a benchmark with a meaningful human reference point: these are the questions we ask human doctors to answer before we let them practice.

In 2024, a team released Meerkat-7B: a 7-billion-parameter model built on top of Mistral-7B, continually trained on medical instructions and chain-of-thought reasoning data generated by GPT-4 from medical textbooks. Think of it as a capable medical student who was tutored, step by step, by the best available teacher, using the actual textbooks. That's mechanism (2) — targeted distillation — with a dash of continued training.

On MedQA, Meerkat-7B scored **74.3%**. Its headline comparison was against MediTron-70B — a 70-billion-parameter model specialized for medicine — which scored **70.2%** (Kim et al., preprint 2024, https://arxiv.org/html/2404.00376v1; the work was subsequently peer-reviewed and published in *npj Digital Medicine*, 2025). Ten times the parameters, the same specialty, worse score. The small model didn't just beat a generalist; it beat a *bigger specialist*.

Why did the textbook-and-tutor recipe work? Because medical QA is a knowledge-and-reasoning task with a stable distribution: diseases don't change their symptoms to fool your model, and the USMLE format is fixed. The distillation concentrated the small model's capacity exactly where the test measures. And note what the recipe required: not a data center, but textbooks and a teacher model's reasoning traces — an investment a single research team could afford. That's the democratizing half of the thesis: the wins aren't reserved for labs with frontier budgets.

A second medical result sharpens the picture. Llama-3-Meditron-8B, an 8-billion-parameter medical model, scored **69.88** on MedQA against Flan-PaLM's **67.60**, and averaged **70.06** across MedMCQA, MedQA, and PubMedQA versus Flan-PaLM's **68.07** (Sallinen et al., AAAI 2025 workshop, https://openreview.net/pdf?id=ZcD35zKujO). One important qualifier: these scores were achieved with the paper's **MediTree inference pipeline** — a Tree-of-Thoughts sampling strategy — not by the base model alone, which scored 63.00 on MedQA (65.88 average) without it. Part of the win belongs to a specialized *inference procedure*, not just specialized weights — Chapter 4's Self-RAG is the same idea, and the book labels it whenever it appears. That's a

small specialist (plus a smart pipeline) beating a larger generalist. But the same paper reports the honest limits: Meditron-8B did *not* beat the MediTron-2 70B average of **70.17**, and GPT-4 scored **74.50**. Specialization lifted the small model past the big generalist, but not past the biggest specialist or the frontier. The thesis holds conditionally — and the conditions are visible in the numbers.

For a hospital administrator, the business consequence is concrete. An 8-billion-parameter model can run on a single GPU inside the hospital's own data center. Patient data never leaves the building — a requirement, not a preference, under health privacy law. The 70-billion-parameter alternative needs a rack of expensive hardware or a third-party API contract. When the smaller model also scores higher on the medical benchmark, the decision stops being a trade-off and becomes obvious. And when it scores slightly lower — as Meditron did against the biggest models — the decision becomes a genuine cost-benefit analysis rather than an automatic win for scale. That honesty is what makes the framework useful: sometimes the answer really is the bigger model, and you'll know why.

The counterexample: when the generalist won

Intellectual honesty requires the other side of the medical story, because it happened and it's instructive.

In March 2023, Microsoft researchers tested GPT-4 — a general model with no medical specialization — on medical challenge problems (Nori et al., 2023). GPT-4 exceeded the USMLE passing score by more than 20 points and **outperformed Med-PaLM, a medical specialist built on a 540-billion-parameter model** (https://www.microsoft.com/en-us/research/uploads/prod/2023/03/GPT-4_medical_benchmarks.pdf). At that moment, raw general scale beat domain specialization. The specialists later caught up — Med-PaLM 2 reached a reported 86.5% on MedQA (Singhal et al., preprint May 2023, later in Nature Medicine 2024) — but the lesson stands: specialization is not destiny. The lead changes hands as methods improve, and a big enough leap in general capability can swamp a specialist's head start.

Why did the generalist win that round? Because Med-PaLM's specialization was built on an older base with an older recipe, while GPT-4 represented a generational jump in pretraining scale and quality. Specialization multiplies the

base model's capability; it doesn't replace it. A 2× specialization gain on a base that's 5× behind still loses. This gives the thesis a dynamic form that the static version misses: the question is never "specialist or generalist?" in the abstract. It's "this specialist, built on this base with this data, versus this generalist, on this task, today?" The honest practitioner re-asks that question every model generation — which, in 2026, means roughly every quarter.

This is exactly why the thesis is conditional. In early 2023, the condition "the specialization is done well enough to matter" wasn't met relative to what GPT-4 brought. By 2024, with better distillation recipes, the 7B specialists were beating 70B medical models. The frontier moves; the specialist's job is to move with it, cheaply.

Finance: the 7B model that beat GPT-4 and BloombergGPT

If medicine tests knowledge, finance tests judgment under noise — and it produced one of the cleanest wins in the literature.

FinMA is a financial language model instruction-tuned on FIT, a dataset of 136,609 financial instruction examples — mechanism (1), supervised fine-tuning, at serious scale. To put that number in perspective: it's roughly the size of a large company's historical archive of analyst notes, earnings summaries, and labeled headlines. An organization sitting on that kind of data already owns most of what it needs to build its own specialist. The paper behind it, PIXIU (Xie et al., preprint, 2023, https://arxiv.org/pdf/2306.05443), evaluated it on the FLARE benchmark suite.

On FPB, a financial sentiment classification task, FinMA-7B achieved an F1 score of **0.86** (FinMA-30B: **0.88**), against GPT-4's **0.78** and BloombergGPT's **0.51**. On financial headline classification, FinMA-7B averaged **0.98** F1 versus GPT-4's **0.86** and BloombergGPT's **0.82**.

Pause on that BloombergGPT number. BloombergGPT is a 50-billion-parameter model pretrained specifically on finance — by Bloomberg, with Bloomberg's data. It is the canonical "big specialist." And a 7-billion-parameter instruction-tuned model beat it decisively on these tasks, while also beating the

frontier generalist. This is the thesis in its purest form: targeted fine-tuning on the task distribution beats both generality *and* brute-force scale.

Why did instruction tuning work so well here? Financial sentiment and headline classification are pattern-recognition tasks over a stable vocabulary — earnings language, analyst phrasing, market terminology. The 136,609 instruction examples taught the model the *shape* of financial text: what "headwinds" means in an earnings call versus a weather report. The generalist knows both meanings; the specialist learned which one matters and stopped hedging. That certainty is what the F1 scores measure.

The business translation: sentiment classification of financial text is a high-volume, well-defined, in-distribution task — exactly the kind of work banks and funds run millions of times a day. A 7B model doing it better than the flagship API, at a small fraction of the cost per token, on hardware you own, is not a research curiosity. It's a budget line. And unlike the flagship API, its behavior is *stable*: the same input gets the same classification every time, with no silent model updates from a vendor changing your results overnight. For regulated financial work, that determinism is worth as much as the accuracy.

The counterexample inside the victory

But the FinMA paper contains its own refutation of the naive thesis, and it's the most instructive failure in this chapter. The same FinMA models that dominated sentiment classification collapsed on numerical finance question answering — a task outside their fine-tuning distribution.

On FinQA (exact match): FinMA-7B scored **0.06**, FinMA-30B **0.11**, against GPT-4's **0.63**. On ConvFinQA: FinMA-7B **0.25**, FinMA-30B **0.40**, against GPT-4's **0.76** (Xie et al., preprint, 2023).

The same model. The same paper. Dominant in-distribution, helpless out-of-distribution. Fine-tuning is a deal with the task distribution: it concentrates the model's capacity where the training data points, and the price is paid everywhere else. This is the "in-distribution" qualifier from Chapter 1 made quantitative — a 0.86-to-0.06 cliff. Any team fine-tuning a model needs this image burned into memory: specialization is not a rising tide. It's a spotlight. Everything outside the beam goes dark.

It's worth understanding *why* the collapse was so total, because the mechanism generalizes. FinMA's instruction tuning taught it the surface patterns of financial text — sentiment-bearing phrases, headline structures, the vocabulary of markets. Numerical QA requires something different: multi-step arithmetic reasoning over tables, chaining quantities through calculations. The tuning data contained almost none of that, so the model never learned it — and worse, the tuning may have actively suppressed the general reasoning the base model had, by rewarding confident pattern-matching over careful computation. This is a milder form of the catastrophic forgetting you'll meet in Chapter 8: the specialist didn't just fail to learn the new skill, it may have unlearned the old one. The practical lesson: before fine-tuning, inventory the skills your task needs and check whether your tuning data teaches all of them. The ones it doesn't teach are the ones that will break in production, silently, at 2 a.m.

Law: specialization wins, honestly scoped

Legal NLP gives us a different shape of result — a specialization win at matched scale, which is worth including precisely because it *isn't* a smaller-beats-larger story.

SaulLM-7B-Instruct, a 7-billion-parameter model specialized for law, scored **0.61** on average on LegalBench-Instruct against roughly **0.55** for the Mistral-7B-Instruct baseline — a gain of 6 absolute points, about 11% relative (preprint, 2024, https://arxiv.org/html/2403.03883v1). Same size, same task family, better performance through legal specialization — mechanisms (1) and (3) working together: instruction tuning on legal tasks plus continued exposure to legal text.

Why does law respond to specialization? Legal language is a dialect — archaic constructions, terms of art, citation formats, and reasoning patterns that barely appear in general web text. A generalist model has seen some of it; a legal specialist has *lived* in it. The 11% relative gain is the measurable value of that immersion. For a law firm, an 11% improvement in instruction-following on legal tasks — drafting assistance, clause review, research triage — compounds across every matter the firm handles.

The honest scoping matters here. This is a 7B model beating a 7B baseline, not a small model beating a giant. It demonstrates that specialization adds real

capability, but it doesn't demonstrate the David-and-Goliath upset. (A later vendor claim put a SaulLM variant at roughly 0.77, described as "GPT-4 class" — that claim is unverified, and this book doesn't use it.) Law firms evaluating AI should read this result correctly: domain adaptation measurably helps, but the legal small-beats-large upset hasn't been rigorously published yet. The gap in the literature is itself information — it tells you where the next wins are likely to come from, and that anyone selling you one today owes you a benchmark.

The gaps are data too

A brief note on what this chapter doesn't contain, because absences carry information. The research behind this book found no rigorous published result of a small fine-tuned model beating a large generalist at customer-support dialogue — only anecdotes, synthetic demos, and vendor claims. That doesn't mean it's impossible; support triage is narrow and well-defined enough that the thesis predicts it *should* work. It means nobody has published the clean experiment yet. Similarly, text classification beyond finance (the FinBERT and CopBERT results against GPT-4) looked promising, but the exact score tables couldn't be verified from primary sources, so they're excluded.

I mention this because the pattern of the missing evidence matters: the wins cluster where benchmarks are mature and public (medicine, finance, code), and the gaps cluster where benchmarks are proprietary or fuzzy (support dialogue, legal practice). If your domain lacks a public benchmark, that doesn't refute the thesis — but it does mean you'll have to build the evaluation yourself before you can claim the win. Chapter 10 covers how.

How fragile are these wins?

A careful reader should now be asking the uncomfortable question: how much should I trust these numbers? It's the right question, and the answers vary by result — which is why this book labels every source.

The medical and finance results come from preprints — arXiv and OpenReview papers that haven't completed peer review. Their numbers are the authors' own reported experiments. That's meaningful evidence, but it's one

team reporting on its own work, and independent replication is the gold standard that turns a result into a fact. Treat the exact margins (74.3 vs 70.2, 0.86 vs 0.78) as the authors' measurements, and the *pattern* — small specialists beating bigger models across independent teams, domains, and years — as the finding that matters. One preprint could be wrong about its margin. Sixteen results across medicine, finance, code, math, and language don't all point the same way by accident.

There's a subtler risk worth naming: benchmark contamination. If a model's training data included the benchmark's test questions, its score measures memorization, not capability. The research behind this book found genuine contamination concerns in the broader literature — models recovering masked answers from famous benchmarks at rates that suggest the tests leaked into training. The honest response isn't to discard every number; it's to weight results by how hard they'd be to game. A fine-tuned 7B model beating GPT-4 on a public classification benchmark is hard to explain by contamination alone, because contamination would help the *bigger* model too — it saw more of the internet. When the small model wins anyway, the specialization explanation survives. Chapter 8 goes deeper on evaluation hygiene; for now, note the principle: trust the *comparisons*, and trust them most when the deck seems stacked against the winner.

Finally, notice what none of these results claim. Nobody claims the 7B medical model is a better doctor than the frontier model in general. Nobody claims FinMA understands finance the way an analyst does. The claims are narrow — higher scores on specific, well-defined benchmarks — and the book's thesis is only as broad as those claims. That's a feature. Narrow claims are checkable, and checkable claims are the ones you can build a business on.

What the professional domains teach

Three patterns emerge from medicine, law, and finance — and they generalize to every chapter that follows.

First, **the wins are real and large**. A 7B medical model beating a 70B medical model by 4 points. A 7B financial model beating GPT-4 by 8–12 points on classification F1 while also beating a 50B finance-pretrained model. These

aren't rounding errors or prompt tricks. They're the predictable result of concentrating training on the task.

Second, **the wins are bounded by the distribution**. FinMA's collapse on numerical QA is the clearest demonstration in the literature that fine-tuning trades breadth for depth. The specialist wins where it was trained and loses where it wasn't — sometimes catastrophically. Deployment planning has to treat the training distribution as the model's territory and everything else as foreign soil.

Third, **the frontier gets a vote**. GPT-4 beating Med-PaLM in 2023 is the permanent reminder that "specialized" is relative to the generalist of the moment. A specialist built on last year's base can be overtaken by this year's generalist. The strategic response isn't to abandon specialization — it's that specialization is *cheap*. Re-tuning a 7B model on a new base costs orders of magnitude less than training a frontier model, so the specialist can re-specialize every generation. The generalist's lead is rented; the specialist's economics compound.

Fourth, **the mechanism matters less than the match**. Medicine's biggest win came from distillation, finance's from supervised fine-tuning, law's from a mix. What they share isn't a technique — it's the fit between the training and the task. Teams sometimes ask "should we fine-tune or distill?" as if one were universally better. The evidence says: pick the mechanism that gets the most task-representative signal into the model for the least cost. Sometimes that's your own labeled data (fine-tuning). Sometimes it's a teacher's reasoning traces (distillation). Sometimes it's a better corpus (domain pretraining). The checklist in Chapter 1 doesn't mention mechanisms at all — deliberately. Mechanisms are implementation; the distribution match is the strategy.

For the non-technical reader, the takeaway is a decision rule: if your work looks like these professions' benchmark tasks — narrow, high-volume, well-specified, with answers you can check — the evidence says a specialized small model will likely beat the big API on quality while winning overwhelmingly on cost, speed, and privacy. If your work looks like novel reasoning across unfamiliar territory, keep reading: Chapters 8 and 9 are about exactly where this rule stops.

And for the technical reader, the takeaway is a research posture: reproduce the wins before you believe them, check the eval setup before you cite them, and

always ask what the model does *outside* the reported distribution. The FinMA cliff — 0.86 to 0.06 — should be your mental reference for every fine-tuning proposal you'll ever review. Ask for the out-of-distribution numbers. If nobody measured them, the win isn't finished.

Key takeaways

- Meerkat-7B (74.3%) beat the 10×-larger MediTron-70B (70.2%) on MedQA via distillation of GPT-4-generated medical reasoning traces.
- Llama-3-Meditron-8B (with the MediTree inference pipeline) beat the larger generalist Flan-PaLM on medical QA — but not the biggest medical specialist or GPT-4, showing the thesis's conditional boundaries.
- FinMA-7B beat GPT-4 and the 50B finance-pretrained BloombergGPT on financial classification (F1 0.86–0.98), via instruction fine-tuning on 136,609 financial examples.
- The same FinMA models collapsed on numerical finance QA (0.06 vs GPT-4's 0.63) — specialization is a spotlight, not a rising tide.
- GPT-4 beat the medical specialist Med-PaLM in 2023: general scale can swamp specialization, so specialists must re-specialize as the frontier advances.
- SaulLM-7B's legal win (+6 points on LegalBench-Instruct) is a genuine specialization gain at matched 7B scale — not a small-beats-large upset.

Sources

- Meerkat-7B: Hyunjae Kim et al., *Small Language Models Learn Enhanced Reasoning Skills from Medical Textbooks* (preprint, 2024; peer-reviewed publication: *npj Digital Medicine*, 2025, https://doi.org/10.1038/s41746-025-01653-8). https://arxiv.org/html/2404.00376v1
- Llama-3-Meditron-8B with the MediTree inference pipeline: Alexandre Sallinen et al., *Llama-3-Meditron: An Open-Weight Suite of Medical LLMs Based on Llama-3.1* (AAAI 2025 GenAI4Health workshop). https://openreview.net/pdf?id=ZcD35zKujO — "MediTree" is the paper's Tree-of-Thoughts inference pipeline, not the model; the base 8B model alone scored 63.00 on MedQA (65.88 avg).

- GPT-4 vs Med-PaLM: Harsha Nori et al., Microsoft, *Capabilities of GPT-4 on Medical Challenge Problems* (March 2023). https://www.microsoft.com/en-us/research/uploads/prod/2023/03/GPT-4_medical_benchmarks.pdf
- Med-PaLM 2 (86.5% on MedQA): Karan Singhal et al., Google (preprint, May 2023; published in Nature Medicine, 2024).
- FinMA / PIXIU: Qianqian Xie et al., *PIXIU: A Large Language Model, Instruction Data and Evaluation Benchmark for Finance* (preprint, 2023). https://arxiv.org/pdf/2306.05443
- SaulLM-7B: *SaulLM-7B: A Pioneering Large Language Model for Law* (preprint, 2024). https://arxiv.org/html/2403.03883v1

Keep reading

This was the free sample — the preface and the first two chapters of *Small Models Win*. The full book is 10 chapters and a glossary: code, math, and reasoning wins; how fine-tuning actually works; the economics chapter with the full invoice math; when fine-tuning fails; where scale still wins; and the migration playbook.

Get the full ebook at aifrontierpost.com/book/small-models-win/