SAMPLE

Building the Foundation Models of Tomorrow

A Practical Guide to the Models Powering the AI Revolution

Aiman JoeHanz

Free sample: Introduction & Chapter 1

putrabooks.github.io

Buy the full book:

Carousell carousell.sg/p/building-the-foundation-models-of-tomorrow-ebook-pdf-1465657186

Gumroad

Ko-fi

ElevenReader

Payhip

Contents

Introduction

Chapter 1: The Current State of Foundation Models

Chapter 2: Limitations and Failure Modes of Foundation Models

Chapter 3: Common Mistakes in Foundation Model Projects

Chapter 4: Foundational Principles of Foundation Model Design

Chapter 5: Building on Foundation Models, Step by Step

Chapter 6: Implementation Strategies for Production Foundation Model Systems

Chapter 7: Technology Deep Dive: Architecture, Training, and Inference

Chapter 8: Case Studies: Real-World Foundation Model Deployments

Chapter 9: Evaluating and Testing Foundation Models

Chapter 10: The Future of Foundation Models

Chapter 11: Action Plan: Building Your Foundation Model Practice

Chapter 12: Advanced Topics: Alignment, Safety, and Governance

Chapter 13: Building a Foundation Model-Ready Organization

Frequently Asked Questions

Glossary

Bibliography

Appendix A: Model Selection Guide for Foundation Models

Appendix B: Fine-Tuning Techniques for Foundation Models

Appendix C: Production Architecture for Foundation Model Systems

Appendix D: Extended FAQ

Appendix E: Cost Optimization Guide for Foundation Model Systems

Appendix F: Evaluation and Testing of Foundation Models

Appendix G: Deployment Checklist for Foundation Model Systems

Appendix H: RAG Implementation Deep Dive

Appendix I: Glossary Supplement

Appendix J: Production Case Studies

Appendix K: Best Practices Summary

Appendix L: The Foundation Model Practitioner's Checklist Library

This free sample contains the Introduction and Chapter 1. The complete book contains the full contents above.

Introduction

Foundation models are large-scale AI models trained on massive datasets that can be adapted to a wide range of downstream tasks. They are the engines behind the AI revolution — the GPTs, Claudes, Geminis, LLaMAs, and Stable Diffusions that have transformed how we work, create, and interact with technology.

The term "foundation model" was coined by Stanford's CRFM in 2021 to describe the shift from task-specific AI models to general-purpose models that serve as a foundation for many applications. This shift has fundamentally changed how AI systems are built, deployed, and used. Before foundation models, building an AI application required collecting labeled data, training a model from scratch, and deploying it for a single task. Today, you start with a foundation model and adapt it — through prompting, retrieval, fine-tuning, or some combination — to your specific use case.

This book is for people who need to understand foundation models as a technology and a discipline. AI engineers building systems on top of foundation models. Product managers developing AI-powered products. Business leaders evaluating AI investments. Students learning about the field. Anyone who wants to understand what foundation models are, how they work, and where they are going.

This book covers foundation models as an engineering discipline — from the current state of the technology to its limitations, from step-by-step guides to building on foundation models to implementation strategies for production systems, from technology deep dives to case studies and the future of the field.

Part I (Chapters 1-3) covers the current state, limitations, and common mistakes. Part II (Chapters 4-11) covers solutions — from principles and step-by-step guides to implementation, deep dives, case studies, evaluation, future directions, and an action plan. Part III (Chapters 12-13) covers advanced topics and organizational adoption.

This book is written for several audiences:

- AI engineers who are building systems on top of foundation models and need a practical guide to adaptation, evaluation, and deployment - Product managers who are developing AI-powered products and need to understand the technology well enough to make informed decisions - Technical leaders who are evaluating AI investments and need to understand the capabilities, limitations, and costs - Students and researchers who are learning about the

field and need a comprehensive overview - Business leaders who want to understand what foundation models mean for their organization

This book assumes basic familiarity with software engineering concepts. You do not need a machine learning background — the book explains the necessary ML concepts as they arise. However, you should be comfortable with programming, APIs, and production software systems.

If you are new to AI, start with Chapters 1-3 for an overview. If you are already building with foundation models, start with Chapters 4-7 for design principles and implementation. If you are evaluating or governing AI, start with Chapters 9 and 12.

Chapter 1: The Current State of Foundation Models

Foundation models share key characteristics that distinguish them from traditional AI models:

- Pre-trained at scale: Trained on massive datasets (trillions of tokens, billions of images) using self-supervised learning objectives. A single model may consume the equivalent of the entire public internet during training. - General-purpose: Can be adapted to many tasks without task-specific training. A single language model can summarize, translate, classify, extract, and generate — all without being explicitly trained for any of these tasks. - Adapted downstream: Fine-tuned, prompted, or used as feature extractors for specific applications. The adaptation methods range from simple prompting (zero cost, zero training) to full fine-tuning (high cost, significant training). - Expensive to train: Training costs range from thousands of dollars for small models to tens of millions for frontier models. The compute, data, and engineering effort required to train a frontier model is comparable to a major infrastructure project.

- Emergent capabilities: Show capabilities not explicitly trained for. Models trained only on next-token prediction develop the ability to write code, answer questions, reason step by step, and use tools — abilities that emerge from scale rather than explicit programming.

The key insight is that foundation models represent a paradigm shift. Traditional AI was a collection of narrow, specialized models. Foundation models are general-purpose platforms that can be adapted to many tasks. This has changed not just how AI is built, but how AI products are designed, how AI teams are organized, and how AI investments are evaluated.

The foundation model ecosystem spans multiple modalities and use cases:

- Language models: GPT-5, Claude 4, Gemini 2, LLaMA 4, Mistral — text understanding and generation. These are the most widely deployed foundation models and power chatbots, search, content creation, code generation, and enterprise applications. - Vision models: SAM (Segment Anything Model), CLIP, DINOv2 — image understanding and segmentation. These models can classify images, detect objects, segment scenes, and understand visual content without task-specific training. - Multimodal models: GPT-4V, Gemini, LLaVA, Claude 3.5 — text, image, audio, video. These models can process and reason across modalities — describing images, answering questions about videos, and generating text from visual input. - Generative models: Stable Diffusion, DALL-

E, Sora, Midjourney — image and video generation.

These models generate new content rather than just understanding existing content. - Code models: Codex, Code LLaMA, StarCoder — code generation and understanding. These models are trained on code and can write, review, debug, and document software. - Audio models: Whisper, ElevenLabs — speech recognition and synthesis. These models transcribe speech to text and generate speech from text, enabling voice interfaces and content creation. - Embedding models: text-embedding-3, BGE, E5 — semantic representation. These models convert text into dense vectors that capture semantic meaning, enabling search, retrieval, and similarity matching.

Each category has its own training pipeline, evaluation metrics, and production considerations. Many production systems use models from multiple categories — a language model for generation, an embedding model for retrieval, a vision model for image understanding.

Foundation model training has several distinct stages, each with its own engineering challenges:

- Data collection: Gather massive datasets (web text, code, images, video). The quality and diversity of training data is the single most important factor in model quality. Data collection involves crawling the web, licensing content, and synthesizing data. - Pre-processing: Clean, filter, deduplicate, and tokenize data. Pre-processing removes low-quality content, personally identifiable information, and duplicates. Tokenization converts text into the discrete units the model processes. - Pre-training: Train on self-supervised objectives (next-token prediction, masked modeling). This is the most compute-intensive stage, requiring thousands of GPUs running for weeks or months. The pre-training objective determines what the model learns to do. - Alignment: Align with human preferences (RLHF, DPO, Constitutional AI).

Alignment makes the model helpful, harmless, and honest. Without alignment, models produce fluent but unhelpful, biased, or harmful output. - Evaluation: Benchmark on standard and custom tests. Evaluation measures quality, safety, and capability across dimensions. Standard benchmarks enable comparison between models; custom benchmarks measure performance on specific use cases. - Deployment: Release via API, open weights, or both. Deployment decisions involve trade-offs between accessibility, control, safety, and business model. API deployment gives the provider control; open-weight deployment gives users freedom.

Each stage involves different teams, tools, and infrastructure. Pre-training is the most expensive and technically demanding stage. Alignment and

evaluation are the most safety-critical. Deployment is the most product-impacting.

Scaling laws describe how model quality improves with three factors:

parameters, data, and compute. The Kaplan scaling laws (2020) showed that model quality follows a power-law relationship with scale — doubling any factor produces a predictable improvement. The Chinchilla scaling laws (2022) refined this, showing that the original laws underestimated the optimal amount of data per parameter.

- Parameters: More parameters -> better quality (up to a point). Models range from 1B parameters (small, efficient) to 1T+ parameters (frontier). The relationship is sublinear — doubling parameters improves quality, but by less than 2x. - Data: More training data -> better quality. Chinchilla showed that optimal training requires approximately 20 tokens per parameter. Many large models are under-trained relative to this ratio. - Compute: More compute -> better quality. Compute is the binding constraint — you can always add more parameters or data if you have more compute. Compute-optimal training balances parameters, data, and compute. - Chinchilla scaling: Optimal training requires a specific ratio of parameters to data. A 70B parameter model should be trained on approximately 1.4 trillion tokens. Many models in the wild violate this ratio.

The scaling laws have been remarkably consistent, driving the trend toward larger models. However, diminishing returns, data scarcity, and cost are raising questions about how long scaling can continue. The field is increasingly focused on efficiency — getting better quality from the same compute, rather than just adding more compute.

Foundation models exhibit emergent capabilities — abilities that appear at scale without being explicitly trained. These capabilities are not present in small models and emerge predictably as model size increases:

- In-context learning: Learning from examples in the prompt. Models can perform new tasks given a few examples, without any weight updates. This is the basis of few-shot prompting. - Chain-of-thought reasoning: Step-by- step reasoning for complex problems. When prompted to "think step by step," models break down problems and solve them incrementally, improving accuracy on math and logic tasks. - Instruction following: Following instructions without specific training. Models trained on instruction-response pairs can follow novel instructions they have never seen before. - Code generation: Writing code from natural language. Language models trained on code can generate working code, explain code, and debug code — capabilities not

explicitly trained but emerging from the code in training data. - Tool use: Using tools and APIs to complete tasks.

Models can learn to call external tools — search engines, calculators, code interpreters — to extend their capabilities beyond text generation. - Multilingual transfer: Applying knowledge across languages. A model trained primarily on English can still answer questions in French or Japanese, transferring knowledge across languages.

The emergence of these capabilities has practical implications. It means that a foundation model can do things it was never trained to do. This is powerful — but also unpredictable. You cannot know in advance which capabilities will emerge at which scale, which makes safety evaluation challenging.

Foundation models are adapted to specific tasks through several methods, each with different cost, effort, and quality trade-offs:

- Prompting: Providing instructions in the input. Zero cost, zero training, but quality depends heavily on prompt quality. Prompting is the starting point for all foundation model applications. - Few-shot learning: Providing examples in the input. Low cost, no training, improves quality by showing the model what good output looks like. Few-shot learning is the fastest way to improve quality. - Fine-tuning: Updating model weights on task-specific data. Higher cost (requires GPU training), but produces the best quality for specific tasks. Full fine-tuning updates all parameters; LoRA/QLoRA updates a small subset. - LoRA/QLoRA: Parameter-efficient fine-tuning.

Trains small adapter modules (typically 1-10% of parameters) while keeping the base model frozen. LoRA is the dominant fine-tuning method in production because it is efficient, composable, and produces quality close to full fine-tuning. - RAG: Retrieving relevant information to augment the prompt. RAG grounds the model's responses in factual, up-to-date information from a knowledge base. It is the dominant production pattern because it reduces hallucination and enables domain-specific knowledge. - Adapter layers: Adding small task-specific modules. Adapters are similar to LoRA but are inserted between transformer layers. They enable multi-task learning with a shared base model.

Most production systems combine multiple adaptation methods. A typical system uses RAG for knowledge, few-shot prompting for quality, and fine-tuning for domain adaptation. The art is in choosing the right combination for your task, budget, and quality requirements.

The current foundation model landscape includes several tiers:

- Frontier proprietary: GPT-5 (OpenAI), Claude 4 (Anthropic), Gemini 2 (Google) — the most capable models, available only through APIs. These models set the state of the art on benchmarks and are updated every few months. - Open weights: LLaMA 4 (Meta), Mistral/Mixtral, Qwen (Alibaba), Phi (Microsoft), Gemma (Google) — models with freely available weights that can be downloaded, self-hosted, and modified. Open models are closing the gap with frontier models. - Specialized: Code LLaMA (code), Med-PaLM (medical), BloombergGPT (finance) — models fine-tuned or pre-trained for specific domains. These models outperform general-purpose models on their target domain. - Small/efficient: Phi-3, Gemma-2B, LLaMA-3.2-1B — models designed for edge deployment. These models run on phones, laptops, and embedded devices, enabling on-device AI. - Multimodal: GPT-4V, Gemini, Claude 3.5 Sonnet, LLaVA — models that process text, images, and sometimes audio and video. Multimodal models are becoming the default as users expect to interact with AI through multiple modalities.

The landscape is changing rapidly. New models are released every few weeks. Open models are getting better faster than closed models. The gap between frontier and open is narrowing. This creates both opportunity (more choices, lower costs) and challenge (harder to keep up, harder to choose).

Training and running foundation models requires significant infrastructure:

- Training clusters: Thousands of GPUs (A100, H100, H200, B200) connected by high-speed networking (InfiniBand, NVLink). Training a frontier model requires a cluster of 10,000+ GPUs running for months. - Inference infrastructure: GPU servers for model serving. Inference is less compute- intensive than training but must be done continuously. Inference infrastructure must handle variable load, ensure low latency, and manage cost. - Cloud providers: AWS, Google Cloud, Azure, CoreWeave, Lambda Labs. Cloud providers offer GPU instances, managed model serving, and AI APIs. They are the primary way most organizations access foundation models. - Orchestration: Kubernetes, Ray, Slurm for distributed training. Orchestration manages the distribution of training across thousands of GPUs, handles failures, and coordinates checkpoints. - Model serving: vLLM, TGI, TensorRT- LLM, Triton. Serving frameworks optimize inference for throughput and latency. vLLM's PagedAttention, for example, improved inference throughput by 2-4x over naive serving.

The infrastructure layer is a significant cost factor. Training a frontier model can cost $10M-$100M in GPU time. Running inference at scale can cost $100K-$1M+ per month. Infrastructure optimization — choosing the right GPUs, serving framework, and deployment strategy — is as important as model

selection.

Foundation models are trained on massive datasets, and data quality is increasingly the bottleneck:

- Text: Common Crawl, Wikipedia, books, code, scientific papers. Text data is the most abundant but also the most variable in quality. Data cleaning — removing low-quality content, duplicates, and PII — is a critical engineering task. - Images: LAION, ImageNet, COCO, web-crawled images. Image datasets are large but noisy. Web-crawled image-text pairs often have mismatched or low-quality captions. - Code: GitHub, Stack Overflow, programming tutorials. Code data is high-quality (it compiles or it does not) and teaches models logic and structure. - Video: Web video, YouTube. Video data is increasingly important for multimodal models but is expensive to process and store. - Audio: LibriSpeech, Common Voice. Audio data is smaller than text or image data, which limits audio model quality.

Data quality, diversity, and scale are critical to model quality. The data layer is increasingly the bottleneck, as high-quality data becomes scarce. Synthetic data — generated by models to train other models — is emerging as a solution, but raises questions about quality, bias, and model collapse.

Foundation models power applications across every industry:

- Text: Chatbots, search, content creation, translation, summarization. Text applications are the most mature and widely deployed. - Code: Code generation, code review, debugging, documentation. Code applications have transformed software development workflows.

- Vision: Image analysis, medical imaging, autonomous driving. Vision applications are high-value but require domain-specific evaluation and safety. - Audio: Speech recognition, text-to-speech, music generation. Audio applications enable voice interfaces and accessibility. - Creative: Image generation, video generation, creative writing. Creative applications are growing rapidly but raise copyright and quality questions. - Enterprise: Document processing, knowledge management, workflow automation. Enterprise applications focus on productivity and cost reduction. - Research: Literature review, data analysis, hypothesis generation. Research applications accelerate scientific discovery.

Each application category has different quality, safety, and cost requirements. A chatbot needs low latency and helpful responses. A medical imaging system needs high accuracy and regulatory compliance. A code generation tool needs correctness and security. The foundation model is the same — the system around it differs.

Foundation models have created a massive economic ecosystem:

- Model providers: OpenAI, Anthropic, Google, Meta — billions in revenue from API access and enterprise contracts. The model provider market is consolidating around a few large players. - Cloud providers: GPU rentals and AI services — tens of billions. Cloud providers are the primary distribution channel for foundation models. - Application companies: AI-powered products — emerging market. Application companies build on foundation models to create products for specific use cases. - Hardware providers: NVIDIA, AMD, TSMC — trillions in market value. Hardware providers are the ultimate beneficiaries of the foundation model boom. - Developer tools: LangChain, Pinecone, Hugging Face — growing ecosystem. Developer tools make it easier to build on foundation models.

The economic impact extends beyond the AI industry. Foundation models are transforming every industry that uses language, code, images, or data — which is all of them. The question is not whether foundation models will affect your industry, but when and how much.

Foundation models face increasing regulation:

- EU AI Act: Foundation model-specific provisions (GPAI models). The EU AI Act requires transparency about training data, capability documentation, and safety evaluations for large models. - US executive orders: Safety reporting requirements for large models. The US approach is less prescriptive than the EU but requires reporting for models above a compute threshold. - Transparency: Training data disclosure, capability documentation. Transparency requirements aim to give regulators and users information about what models can do and how they were trained. - Safety: Red teaming requirements, safety evaluations. Safety requirements ensure models are tested for harmful capabilities before deployment. - Open vs. closed: Debate about open vs. closed foundation models. Regulators are grappling with whether open models should face the same requirements as closed models.

The debate over open vs. closed foundation models is one of the most contentious in AI:

- Open: LLaMA, Mistral, Qwen — available for anyone to download and use. Open models enable customization, privacy, research, and cost control. - Closed: GPT-5, Claude, Gemini — available only through APIs. Closed models offer higher quality, safety guarantees, and ease of use. - Arguments for open:

Democratization of AI, customization for specific use cases, privacy (no data leaves your infrastructure), research and education, cost control (inference cost only). - Arguments for closed: Safety (providers can monitor and revoke

access), quality (frontier models are closed), accountability (providers are responsible for output), ease of use (no infrastructure needed). - Hybrid: Open weights with API access (many providers offer both). Hybrid models give users the choice between self-hosting and API access.

The foundation model competition is intense across multiple dimensions:

- Quality race: Each provider pushing for better benchmarks. The quality gap between providers is narrowing — models from different providers are increasingly interchangeable for most tasks. - Efficiency race: Smaller models with comparable performance. Phi-3 (3B parameters) approaches GPT-3.5 quality. Efficiency improvements reduce cost and enable edge deployment. - Multimodal race: Models that handle text, image, audio, video. Multimodal models are becoming the default — users expect to interact through whatever modality is natural. - Price race: Lowering API costs to attract developers. API prices have dropped by 10-100x in two years. Price competition is brutal and benefits users. - Context race: Longer context windows for more complex tasks. Context windows have grown from 2K to 1M+ tokens. Long context enables processing entire books and codebases. - Speed race: Faster inference for real-time applications. Latency is a critical product metric — users will not wait 30 seconds for a response.

Foundation models are improving at an extraordinary pace:

- New models every few months from multiple providers - Performance doubling every 6-12 months on many benchmarks - New capabilities emerging with each generation - Costs decreasing while quality improves

This pace creates both opportunity and challenge — organizations must continuously evaluate and adopt new developments while maintaining stable production systems. The half-life of a "best model" is now 3-6 months. Systems must be designed for model portability — the ability to swap models without rewriting the application.

PRACTICAL INSIGHT The foundation model era has fundamentally changed AI development. Five years ago, building an AI application required training a model from scratch. Today, you adapt a foundation model. This has democratized AI — anyone can build AI applications — but it has also created new challenges: model selection, cost management, evaluation, and safety.

The key skill is no longer training models but adapting and deploying them effectively.

KEY TAKEAWAYS

- Foundation models are pre-trained at scale and adapted downstream through

prompting, RAG, and fine-tuning - The model landscape includes frontier proprietary models, open weights, specialized, and efficient models - Scaling laws predict quality from parameters, data, and compute — but efficiency is now the focus - Emergent capabilities appear at scale without explicit training, making models powerful but unpredictable - The pace of innovation requires designing for model portability — expect to swap models every 3-6 months

End of free sample. The full book is available at putrabooks.github.io

ABOUT THE AUTHOR

Aiman JoeHanz writes about the technologies that are quietly reshaping the world. His Building of Tomorrow series examines the infrastructure behind artificial intelligence -- from blockchain and cloud computing to agentic systems and token optimization -- and asks how these technologies will evolve over the coming decade.

His books are built on technical research and engineering documentation, written for readers who want to understand systems rather than trends. He covers what is being built today and what it will take to build what comes next.

He believes that understanding infrastructure is the difference between watching the future happen and helping to create it.