Maxim Bogdanow

Evaluating Local and Small Language Models

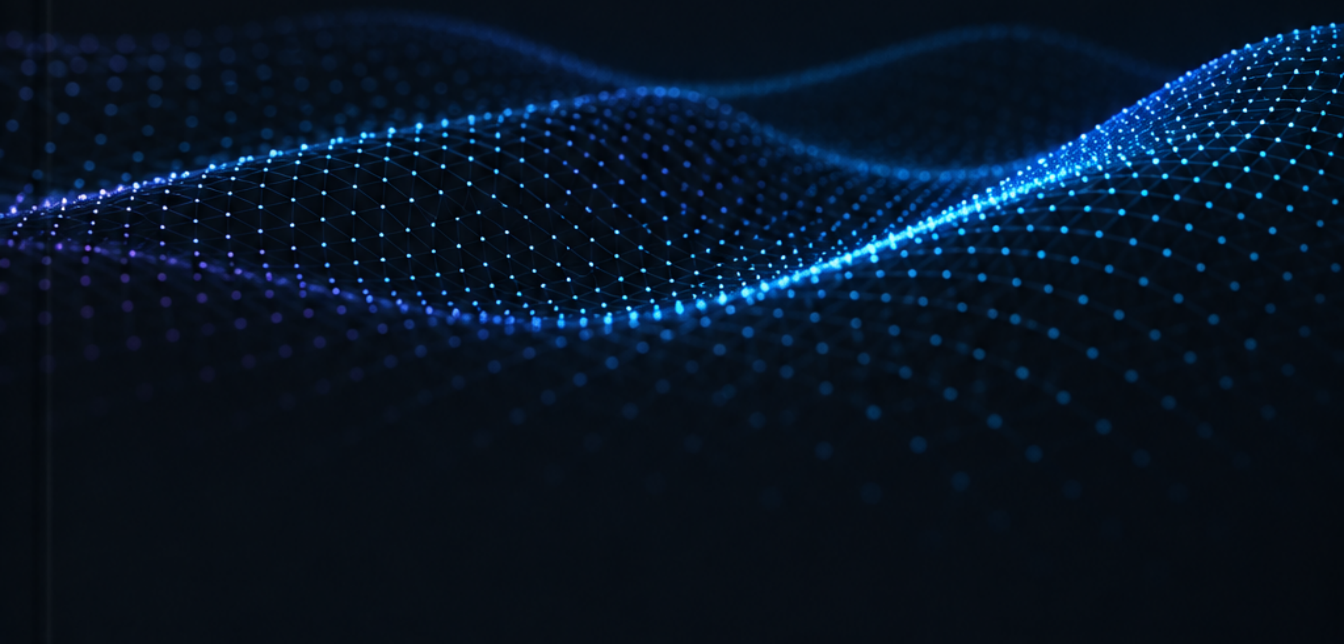
A Practical Engineering Guide to Benchmarking, Optimization, and Production Deployment


Benchmarking


Optimization


Hardware & Runtimes


Production Deployment



Evaluating Local and Small Language Models

A Practical Engineering Guide to Benchmarking, Optimization, and Production Deployment

Steve Publications

This book is available at
https://leanpub.com/evaluatinglocalandsmalllanguagemodels

This version was published on 2026-09-07

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

A Practical Engineering Guide to Benchmarking, Optimization, and Production Deployment

This book teaches engineers how to build reliable evaluation systems for small and locally deployed language models. You will learn to measure performance accurately, compare models across hardware and runtimes, interpret results without being misled by benchmarks, and make evidence-based deployment decisions for real-world workloads. The guide combines rigorous experimental methodology with example code so you can independently establish an evaluation laboratory and produce trustworthy, reproducible results.

Introduction

You run a small language model on a laptop for a demo. It generates text at eight tokens per second and seems adequate. You then move to production with a different quantization, a different runtime, and a different workload, and the numbers change so drastically that the demo performance becomes meaningless. This is a common experience in the world of local language model deployment.

The problem is not that small or local models are unreliable. The problem is that evaluation is often done with the wrong level of rigor. Engineers rely on single benchmark scores, anecdotal observations, or vendor claims that do not match their actual workload. The result is poor model selection, unexpected performance in production, and wasted money on hardware or cloud spend.

This book exists to fix that gap. It treats local and small language model evaluation as an engineering discipline that requires systematic measurement, statistical rigor, workload-specific methodology, and reproducibility. You will not find vague advice here. Instead you will find complete evaluation procedures, working code, and decision frameworks grounded in real system behavior.

Why evaluation matters more for local models

When you call a cloud API, evaluation largely means choosing among well-documented options and trusting the provider. When you deploy locally, the surface area explodes. You now choose the model family, the parameter size, the quantization format, the inference runtime, the hardware configuration, the threading settings, the memory layout, the batch size, the context window, the concurrency level, and the temperature. Each choice interacts with every other choice. The configuration space is large enough that intuition fails.

Three factors make rigorous evaluation essential for local models. First, the gap between advertised capability and actual performance can be wide. A model that scores well on a public benchmark may perform poorly on your domain, especially if the benchmark was contaminated during training.

Second, system performance is just as important as model quality. A slower model running on better hardware may serve users more responsively than a faster model on worse hardware. Third, resource constraints force trade-offs that must be measured, not guessed. You need to know whether the quality loss from INT4 quantization matters for your task, whether a 3 billion parameter model is good enough, or whether adding a second GPU improves throughput enough to justify the cost.

What you will be able to do after reading this book

By the end of this book you will be able to do the following things:

Design and run statistically sound evaluations that produce results you can trust. You will know how to choose appropriate test sets, how many samples you need, how to control for variance, and how to report results with confidence intervals instead of single numbers that hide uncertainty.

Measure system performance correctly. You will understand the difference between time-to-first-token and inter-token latency, how prompt length affects each, how batching changes throughput, and how to identify whether your system is compute-bound or memory-bound.

Compare models and configurations without being misled. You will know how public benchmarks can deceive you, how to detect contamination, why aggregate scores are dangerous, and how to build custom evaluations that reflect your actual workload.

Make deployment decisions with evidence. You will have frameworks for choosing among quantization levels, model sizes, hardware options, and runtimes based on measured trade-offs between quality, latency, throughput, memory, and cost.

Detect false claims. Whether from vendors, blog posts, or benchmark leaderboards, you will know what questions to ask and what experiments to run to verify performance assertions.

Build reproducible evaluation pipelines. You will have complete code for downloading models, running inference, measuring latency, collecting telemetry, validating outputs, comparing results, and generating reports that another engineer can reproduce.

Who this book is for

This book assumes you are comfortable programming and working with systems. It is written for software engineers, ML engineers, AI developers, DevOps engineers, researchers, and technical decision-makers who need to work with local or small language models. You should know how to run a Python script, what a GPU is, and how to use command-line tools. You do not need deep theory knowledge of transformers, though the book explains what you need.

If you are a manager evaluating whether to deploy models on-premise, you will find the decision frameworks, cost models, and risk analysis useful. If you are an engineer building a product around local models, you will find the implementation guidance and code essential. If you are a researcher comparing model capabilities, you will find the statistical methodology and benchmarking analysis important.

How this book is organized

The book is structured as a progression from foundations to mastery. Part I establishes the landscape: what small and local models are, why they matter, and the technical foundations you need. Part II covers the evaluation methodology: how to design tests, measure performance, and apply statistical rigor. Part III dives into the tooling: inference runtimes, model formats, quantization, and hardware considerations. Part IV addresses practical evaluation across different workload types. Part V covers advanced techniques and optimization. Part VI brings everything together with decision frameworks and production guidance.

You can read this book sequentially, which is recommended if you are new to local model evaluation. If you are already experienced, you can jump to specific chapters based on your needs. The code examples are standalone and executable, so you can run them without reading the surrounding theory first, though understanding the theory will help you interpret the results.

A note on pace and specificity

The local language model ecosystem moves fast. New models, new runtimes, and new benchmarks appear regularly. This book is written with two goals:

to give you current, accurate information about the tools and methods available today, and to give you durable principles that will remain useful as the ecosystem evolves. Where details are rapidly changing, the book explains the underlying mechanics so you can reason about new tools when they appear. Where principles are stable, the book is explicit about why they hold.

Performance numbers, model rankings, and tool recommendations will age. The methodology for measuring performance, the statistical principles for comparing results, the understanding of system bottlenecks, and the frameworks for workload-specific evaluation will not. This book is designed so that the parts that matter most for your long-term work are the parts that stay correct.

Chapter 1: Defining the Landscape

The landscape of small and local language models has changed dramatically in recent years. Models that once required multi-GPU clusters now run on consumer laptops. Frameworks that once demanded deep expertise now offer simple APIs. This progress is both an opportunity and a hazard. The opportunity is that powerful models are now accessible. The hazard is that the same accessibility makes it easy to deploy something inadequate and discover the problem only after users notice.

This chapter establishes the terms and context for the rest of the book. We define what we mean by small and local, why local deployment matters, what hardware is available, and the trade-offs that shape every decision.

What is a Small Language Model

There is no universally agreed definition of what constitutes a small language model. The boundary between small and large has moved as models have grown and as optimization techniques have improved. One organization calls anything below 72 billion parameters small [1]. Another treats small as meaning millions to low billions of parameters [2]. For the purposes of this book, a small language model is one that can run on a single consumer-grade device or a modest server, typically with parameter counts in the range of 1 billion to 14 billion, though the effective upper bound depends on quantization and hardware.

The distinction between small and large is not just about parameter count. It is about the deployment context and the capability envelope. Small models are designed to run where large models cannot: on laptops, phones, edge devices, air-gapped servers, and constrained environments. They are specialized rather than generalist, optimized for specific tasks rather than maximum breadth. A well-chosen small model deployed locally can outperform a distant large model when latency, privacy, cost, or reliability matter.

The current landscape includes several notable families of small models:

- Meta Llama 3.2 1B and 3B: Highly efficient, designed specifically for edge deployment with strong instruction-following capability for their size.
- Google Gemma 2: Open-weight models in the 2B to 9B range, competitive with older Llama variants.
- Qwen 2.5 series: 0.5B to 14B parameter models with strong multilingual and coding capability.
- Mistral and Mixtral: 7B and 8B dense models, plus mixture-of-experts variants that trade peak quality for efficiency.
- Phi series: Microsoft models in the 1.5B to 3.8B range that demonstrate strong performance relative to size through careful training data selection.
- Various community and domain-specific models: Thousands of fine-tuned variants built for specific tasks, domains, or languages.

The key insight is that model quality is no longer a simple function of parameter count. A 3B parameter model trained on high-quality, domain-specific data and evaluated on that domain may outperform a 70B generalist model. This is why evaluation must be workload-specific rather than based on aggregate benchmark scores.

What is a Local Language Model

A local language model is one that runs on infrastructure you control, rather than through a third-party API. This includes models running on your laptop, on an on-premise server, in a private cloud, on edge devices, or in any environment where the model weights, the inference code, and the data all remain under your control.

Local deployment means three things: data sovereignty, operational control, and self-sufficiency. Data never leaves your infrastructure, which matters for privacy, compliance, and security. You control the operational details: which model to use, how to update it, how to monitor it, how to scale it. And you do not depend on external availability or pricing changes.

The alternative to local deployment is not just public cloud APIs. It includes managed services that still send your data to a provider, hybrid architectures that split workloads between local and remote, and even models hosted on third-party infrastructure you rent but do not fully control. The defining

characteristic of truly local deployment is that you own the full stack from weights to response.

Local deployment has become practical because three trends converged:

First, model quality at smaller sizes improved dramatically. A 7B parameter model from 2024 can handle many tasks that once required 70B or more parameters, thanks to better training data, better architectures, and better alignment techniques.

Second, quantization and compression techniques advanced. Models can now be run at 4-bit or even lower precision with quality loss that is often acceptable for many tasks, reducing memory requirements by 75 percent or more.

Third, inference runtimes matured. Frameworks like llama.cpp, Ollama, and MLX make it possible to run models with minimal setup, and the performance on consumer hardware is now good enough for many production workloads.

Why Local Matters

The choice between local and cloud deployment is not purely technical. It is shaped by requirements around privacy, cost, latency, reliability, and control. Each factor becomes important in different contexts.

Privacy and data sovereignty are the primary drivers for many organizations. The average data breach costs around $4.44 million [3], and regulatory frameworks like GDPR impose fines up to 4 percent of global annual turnover. When your data includes personal information, medical records, financial details, intellectual property, or proprietary business data, sending it to a third-party API introduces risk. Even if the provider is trustworthy and their security is good, the data must traverse networks outside your control, and you depend on their internal processes. Local deployment eliminates this exposure by design.

Cost becomes important at scale. A cost-benefit analysis published in 2025 shows that on-premise deployment can reach breakeven with cloud services after three to six months for organizations processing more than 50,000 queries per month [4]. Local inference can cost approximately $0.11 per million tokens compared to around $2.00 for cloud APIs at production scale [5]. The exact numbers depend on hardware choices and utilization, but the pattern is consistent: the more you use the model, the more local deployment saves.

Latency matters for user experience. Every API call adds network round-trip time, queueing delay, and provider-side processing overhead. A local model eliminates network latency entirely and can respond in milliseconds when the model is already loaded in memory. For interactive applications, coding assistants, or agentic workflows that require dozens of sequential calls, local inference can be orders of magnitude faster.

Reliability is about uptime and predictability. Cloud services have outages. Rate limits change. APIs are deprecated. Pricing increases. When your business depends on a model, depending on an external service introduces a single point of failure. Local deployment puts availability in your hands.

Control means the ability to customize, audit, and evolve the model. You can fine-tune on your data, apply custom quantization, patch vulnerabilities, modify the runtime, and roll back changes without asking anyone. You can audit what the model is doing and why. For regulated industries this is not optional.

These factors compound. An organization that needs privacy also needs control. An organization that processes high volume also cares about cost. The organizations most interested in local models are usually motivated by multiple factors simultaneously.

The Hardware Landscape

Understanding the hardware landscape is essential because model performance is ultimately determined by the physical constraints of the device running it. The same model behaves very differently on different hardware, and choosing the right hardware is as important as choosing the right model.

Consumer-grade hardware has become powerful enough for meaningful local inference. Modern laptops with Apple Silicon (M1 through M5) offer unified memory architectures with bandwidth up to 153 GB/s and memory capacities up to 192 GB, making them capable of running 14B parameter models responsively and even 70B parameter models at lower throughput. Windows and Linux laptops with dedicated NVIDIA GPUs (RTX 4060 through RTX 5090) provide high-performance compute with 8 GB to 32 GB of VRAM. CPUs on modern laptops are also capable for small models through frameworks like llama.cpp, though at lower speeds.

Desktop and workstation GPUs are the workhorses for serious local deployment. NVIDIA RTX 4090 (24 GB VRAM), RTX 5090 (32 GB VRAM), and professional cards like the NVIDIA L4 (24 GB) provide sufficient memory for 7B to 14B models at high quality quantization with room for context. Multi-GPU setups allow running larger models by splitting layers across cards or running multiple instances in parallel.

Server-grade hardware becomes relevant for production workloads. NVIDIA H100 (80 GB), L40S (48 GB), and A100 (40-80 GB) provide the throughput and memory for running 70B+ models at production scale. AMD Instinct MI300X (192 GB) is a powerful alternative with massive memory capacity. The choice between these depends on budget, power constraints, and whether you need cutting-edge performance or simply adequate throughput.

Edge devices present unique constraints. Smartphones with NPU hardware (Apple Neural Engine, Qualcomm Hexagon) can run models in the 1B to 3B range on device. Embedded systems and IoT devices with limited memory and power require specialized optimization, often with extreme quantization and model distillation.

The key principle is to match hardware to workload. A laptop running llama.cpp is excellent for development, demos, and light workloads. A desktop with a good GPU is appropriate for a small team. A server with multiple GPUs is needed for production serving at scale. The wrong hardware choice wastes money and frustrates users.

Cost and Privacy Implications

The economics of local deployment are straightforward in theory but nuanced in practice. The upfront cost is hardware: purchasing GPUs, servers, and storage. A capable entry-level server for running a 7B model starts at $8,000 to $12,000 [4]. The ongoing cost is electricity, cooling, maintenance, and engineering time. The savings come from not paying per-token fees, not paying for API calls, and not paying for data transfer.

The breakeven analysis depends on three variables: usage volume, model quality requirements, and hardware efficiency. If you process millions of tokens per month, local deployment almost always wins on cost. If you process

thousands, cloud is cheaper. The transition point shifts as cloud prices change and hardware becomes more efficient.

Privacy implications are more absolute than cost implications. There is no breakeven calculation for data that must not leave your network. When regulations require data to stay in a specific jurisdiction, when intellectual property cannot be exposed to third parties, when patient data is involved, or when legal counsel requires it, local deployment is not a cost optimization. It is a requirement.

The combination of privacy and cost is what makes local deployment compelling for enterprises. A bank that needs local deployment for compliance reasons will also benefit from the cost savings at scale. A healthcare provider that must keep medical data private will also save on API costs by deploying locally. The privacy requirement justifies the initial investment, and the cost savings justify continued operation.

Organizations should also consider the hidden costs of cloud deployment: vendor lock-in, API compatibility changes, unpredictable pricing, and the operational complexity of managing multiple provider relationships. Local deployment centralizes control and makes long-term cost planning more predictable.

How This Book is Structured

This book is organized to build your knowledge progressively from foundations to advanced practice. Each chapter assumes knowledge from previous chapters, so sequential reading is recommended for newcomers.

Chapters 2 and 3 establish the technical foundations. You will learn about transformer architectures, parameter scaling, attention mechanisms, and the different dimensions of model quality. This knowledge is necessary for understanding why models behave the way they do.

Chapters 4 and 5 cover model formats and quantization. You will understand how models are stored, transferred, and loaded, and how quantization affects quality and performance. These chapters are essential for practical deployment work.

Chapters 6 through 8 introduce the inference stack and specific runtimes. You will learn about llama.cpp, Ollama, MLX, and vLLM, including their architectures, trade-offs, and configuration options.

Chapters 9 and 10 explain tokenization and performance metrics. You will learn to measure latency, throughput, and system utilization correctly, and to identify performance bottlenecks.

Chapters 11 through 14 cover evaluation methodology, statistical rigor, and public benchmarks. This is the core of the book: how to design evaluations that produce trustworthy results, how to apply statistics correctly, and how to interpret benchmark scores without being misled.

Chapters 15 and 16 show you how to build custom evaluation harnesses with complete code. You will learn to measure quality systematically, detect hallucinations, validate structured outputs, and generate reproducible reports.

Chapters 17 and 18 address workload-specific evaluation and safety testing. You will learn how different use cases require different evaluation approaches, and how to test for reliability, robustness, and safety.

Chapter 19 covers advanced optimization techniques including speculative decoding, continuous batching, model distillation, and hardware acceleration.

Chapter 20 brings everything together with decision frameworks, case studies, and production deployment guidance.

Throughout the book, code examples are provided as complete, runnable implementations with installation instructions, execution steps, and expected outputs. You can run them as you read to build hands-on experience with evaluation techniques.

The goal is not to produce benchmark tables or model rankings. The goal is to give you the skills, tools, and understanding to build your own evaluation capability so that you can answer questions about model performance for your specific needs with evidence rather than opinion.

Chapter 2: Model Architectures and Scaling

To evaluate models effectively you need to understand how they work internally. Architectural differences explain why two models with similar parameter counts can have vastly different inference speeds, memory requirements, and quality characteristics. This chapter covers the technical foundations of transformer models, parameter scaling, attention mechanisms, and architectural choices that directly affect your evaluation work.

Transformer Fundamentals

The transformer architecture introduced in the Attention Is All You Need paper [6] became the foundation for virtually all modern language models. Understanding the basic mechanics is essential because inference performance and resource usage follow directly from the architecture.

A transformer processes text as a sequence of tokens. Each token is mapped to an embedding vector, a numerical representation in high-dimensional space. The core computation happens in transformer layers, each containing two main components: a self-attention mechanism and a feed-forward network.

The self-attention mechanism allows every token in the sequence to attend to every other token. For each token, the model computes three vectors: a query vector, a key vector, and a value vector. Attention scores are computed by taking the dot product of query and key vectors, scaled by the square root of the dimension. These scores determine how much each token attends to each other token. The output is a weighted sum of value vectors, where the weights come from the attention scores.

The feed-forward network is applied independently to each token position. It consists of two linear transformations with a non-linearity in between. Its purpose is to process the information from the attention mechanism and produce the final output for each position.

During inference, tokens are generated autoregressively. The model generates one token at a time, each conditioned on all previous tokens. At each step, the entire sequence is processed through all layers. This is where the key performance constraint emerges.

The KV cache is the memory structure that stores the key and value vectors for each token in each layer. Once computed for a token, these vectors are cached so they do not need to be recomputed at each generation step. Without caching, inference would require reprocessing all previous tokens at every step. With caching, each new token only requires computing its own key and value vectors while attending to all cached vectors.

The KV cache grows linearly with sequence length and number of layers. This makes it the dominant memory consumer for long contexts and the primary bottleneck for batched inference. Understanding this is critical for performance evaluation.

Parameter Count and Scaling Laws

Parameter count is the most commonly cited model characteristic, but it is neither the whole story nor as simple as it appears. A parameter is a weight in the neural network that is learned during training. The total parameter count is the sum of all weights across all layers.

For transformer models, parameters are distributed across several components:

- Embedding layer: maps token IDs to vectors, roughly vocabulary size times embedding dimension
- Attention layers: query, key, and value projection matrices, plus output projection
- Feed-forward layers: two weight matrices per layer
- Layer normalization: scale and shift parameters per layer

A 7 billion parameter model contains 7 billion trainable weights. Each weight in full precision (FP16) occupies 2 bytes. The raw model size is therefore 14 gigabytes, not counting activation memory, the KV cache, or other overhead.

Scaling laws describe the empirical relationship between model size, data, compute, and performance. The Chinchilla paper [7] showed that for a

given amount of training compute, there is an optimal ratio between model parameters and training tokens. Larger models require proportionally more training data to reach their potential. This finding has practical implications: a model that is large but trained on insufficient data may underperform a smaller, better-trained model.

More relevant for evaluation is the observation that model quality scales predictably but with diminishing returns. Doubling the parameter count does not double the quality. The improvement is real but sublinear, especially for task-specific performance. A 14B model is better than a 7B model, but not twice as good. A 70B model is better than a 14B model, but the relative improvement continues to decrease.

This means that for local deployment there is a practical ceiling. Beyond a certain size, the marginal quality improvement does not justify the marginal resource cost. For many tasks, a 7B to 14B model provides sufficient quality at a fraction of the resource cost of a 70B model. Your evaluation work should determine where that ceiling lies for your specific workload.

Attention Mechanisms and Variants

The attention mechanism is where most architectural innovation happens, and it is also where inference performance is most affected. The standard multi-head attention (MHA) divides the embedding dimension into multiple heads, each computing attention independently. If the model has h heads and each key-value head has dimension d_k, then MHA uses h key-value heads.

The problem with MHA during inference is that it requires storing separate key-value caches for each head. For a model with 32 heads and 128-dimensional heads, at 128K context tokens, the KV cache alone consumes significant memory. This is the fundamental bottleneck that attention variants attempt to address.

Multi-query attention (MQA) [8] uses a single shared key-value head across all query heads. All query heads attend to the same key and value vectors. This dramatically reduces KV cache size because instead of storing h separate caches, the model stores only one. The KV cache shrinks by a factor of h.

The trade-off is quality. Sharing all key-value heads reduces representational capacity. The model cannot maintain head-specific information in the

cache. MQA typically causes a measurable quality degradation compared to MHA, especially on complex reasoning tasks.

Grouped-query attention (GQA) [9] is a middle ground. Query heads are grouped into g groups, with each group sharing a single key-value head. GQA uses g key-value heads instead of h. If g equals h, GQA becomes MHA. If g equals 1, GQA becomes MQA. For intermediate values, GQA balances KV cache efficiency and quality.

Modern models like Llama 3 use GQA extensively. Llama 3 8B uses 32 query heads grouped into 8 key-value heads. This gives 75 percent KV cache reduction compared to MHA while maintaining most of the quality. For inference evaluation, GQA is important because it directly affects memory requirements and batch throughput.

The practical implication is that attention choice matters for hardware planning. A model with GQA will fit larger contexts and support higher concurrency than an equivalent MHA model. Two models with similar parameter counts but different attention configurations will have different practical limits on the same hardware.

Grouped-Query and Multi-Query Attention

To understand the KV cache implications concretely, let us calculate the memory required for different attention configurations. The KV cache memory formula is:

2 * layers * kv_heads * head_dim * seq_len * bytes_per_element

The factor of 2 accounts for both key and value caches. Consider three models with identical parameter counts and 32 layers, evaluated at 32K context with FP16 precision:

- MHA with 32 heads, 128-dimensional: 2 * 32 * 32 * 128 * 32768 * 2 bytes = 13.4 GB
- GQA with 32 query heads / 8 kv heads: 2 * 32 * 8 * 128 * 32768 * 2 bytes = 3.4 GB
- MQA with 1 kv head: 2 * 32 * 1 * 128 * 32768 * 2 bytes = 0.4 GB

The difference is dramatic. An MHA model at 32K context needs 13.4 GB for KV cache alone. A GQA model needs one quarter of that. An MQA model needs one thirty-second.

For evaluation, this means attention configuration directly determines what workloads are feasible. On a 24 GB GPU, an 8B parameter model with MHA can run about 16K context before KV cache exhausts available memory. The same model with GQA can run 64K context. The difference between architectures is not just theoretical: it determines whether a model can actually serve your workload.

This also affects concurrency. Each concurrent request needs its own KV cache. Serving 8 simultaneous users at 32K context with MHA requires 107 GB of cache memory. With GQA it requires 27 GB. With MQA it requires 3 GB. If your workload requires concurrency, attention mechanism is one of the most important architectural factors.

Mixture of Experts Architectures

Mixture of Experts (MoE) models use a different approach to scaling. Instead of increasing the total parameter count of a single dense model, MoE models partition parameters across multiple expert networks and use a gating mechanism to route each token to a subset of experts.

A MoE model might have 128 billion total parameters but only activate 16 billion per token. The total parameter count is large but the active parameter count per inference step is small. This means inference is faster and uses less memory than a dense model with the same total parameters.

Mixtral 8x7B is a notable MoE model with 47B total parameters and 13B active parameters per token. The model has 8 expert networks, each with 7B parameters, but only 2 experts are activated for each token.

For local deployment, MoE models present a complex trade-off. The advantage is that inference is faster than a dense model of equivalent total size because fewer parameters are activated. The disadvantage is that all experts must be loaded into memory even though only a subset is used at each step. For Mixtral 8x7B, all 47B parameters must reside in memory even though only 13B are active.

This means that while MoE models are more efficient computationally, they are not more memory-efficient for total parameters. A 7B dense model still fits in less memory than Mixtral 8x7B. For resource-constrained local deployment, a dense model is often more practical.

MoE models also complicate evaluation. The routing mechanism introduces variability: different tokens activate different experts, which means performance can vary across different inputs. When comparing a MoE model to a dense model, you should evaluate on diverse workloads to account for routing variance.

MoE is also relevant for understanding future model families. As more models adopt MoE architectures, the evaluation methodology needs to account for the difference between total parameters and active parameters.

Context Window Mechanics

The context window is the maximum number of tokens a model can process in a single forward pass. Modern models advertise context windows ranging from 8K tokens to millions of tokens. The advertised number is not the same as the effective window.

During training, a model sees sequences up to a maximum length. If trained primarily on 4K token sequences and never tested on longer sequences during training, the model may not generalize well beyond 4K tokens at inference time. This is why context window claims require verification.

Three factors limit effective context:

First, architectural constraints. The attention mechanism scales quadratically with sequence length during training. For a 32K token sequence with multi-head attention, the attention computation is 64 times more expensive than for a 4K token sequence. Many models use optimized attention kernels (Flash Attention, etc.) to reduce this cost, but the scaling is still a factor.

Second, memory constraints. The KV cache grows linearly with context length. A model that can handle 8K context on a 24 GB GPU may not handle 64K context on the same hardware because the cache exceeds available memory, even though the model architecture theoretically supports it.

Third, performance degradation. Models tend to perform worse on information in the middle of long contexts, a phenomenon called lost in the middle [10]. Retrieval accuracy often follows a U-shaped curve, being higher for information at the beginning and end of the context than for information in the middle. This means that a 128K context window is not uniformly useful.

For evaluation, context window testing requires specific methodology. You should measure performance at different context lengths, not just at the maximum. You should test information retrieval from different positions in the context. You should verify that the model does not simply refuse longer contexts due to memory limits.

Context window is also workload-dependent. If your workload only needs 4K tokens of context, a model with a 128K window provides no advantage. But if your workload involves analyzing long documents or maintaining extended conversations, context window becomes critical.

Architectural Choices that Affect Inference

Several architectural choices beyond attention mechanism and context window directly affect inference performance and should be considered during evaluation:

Feed-forward dimension determines the capacity of the MLP layers. A larger feed-forward dimension relative to embedding dimension typically improves quality but increases inference cost because more computation is required per token. This is especially relevant for decoding throughput.

Layer normalization placement matters. Pre-norm architectures apply normalization before attention and feed-forward, which improves training stability. Post-norm architectures apply it after. Most modern models use pre-norm or variants like RMSNorm, which has slightly lower computational overhead.

Activation function choice affects both quality and speed. Models like Llama 3 use SwiGLU instead of the traditional GELU activation in feed-forward layers. SwiGLU splits the MLP path into two branches with a gating mechanism, which empirically improves quality but adds a small computational overhead.

Positional encoding is critical for context window behavior. Models using Rotary Positional Embeddings (RoPE) [11] handle longer contexts more gracefully than models using absolute positional encodings. RoPE encodes positions as rotations in the embedding space, which allows better generalization to sequences longer than those seen during training.

For evaluation, these architectural details matter when comparing models from different families. A model with SwiGLU may have better quality than

one with GELU but slightly slower inference. A model with RoPE may handle longer contexts better. When two models perform similarly on benchmarks, architectural efficiency can determine which is better suited for a specific deployment scenario.

Chapter 3: Model Quality and Capability Dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Instruction Following and Alignment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Reasoning and Chain-of-Thought

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Coding and Technical Tasks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Structured Output and Format Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Multilingual Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Long-Context Understanding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Tool Use and Function Calling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Hallucination and Factuality

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Refusal Behavior and Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 4: Model Weights, Formats, and Storage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Safetensors and PyTorch Weights

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

GGUF Format and llama.cpp

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

ONNX and Cross-Platform Export

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Model Card Information

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Quantization-Aware Format Choices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Storage Requirements and Disk I/O

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Weight Loading Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 5: Quantization Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Why Quantize

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Full-Precision vs. Quantized Representations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

INT8 and INT4 Quantization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Per-Tensor vs. Per-Channel vs. Per-Token

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Quantization-Aware Training and Post-Training

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

GGUF Quantization Schemes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Measuring Quantization Impact

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

When Quantization Fails

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 6: The Local Inference Stack

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Hardware Abstraction Layers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

GPU Runtimes and Drivers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Memory Management and Allocation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Context Switching and Scheduling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Process Isolation and Resource Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Container Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

The Inference Stack Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 7: llama.cpp and GGUF Ecosystem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

llama.cpp Architecture and Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

GGUF Model Loading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Quantization Selection in llama.cpp

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Threading and CPU Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Metal and GPU Offloading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

API and Embedding Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Performance Tuning and Flags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Limitations and Gotchas

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 8: Ollama, MLX, and Alternative Runtimes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Ollama Architecture and Modelfile

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Running and Customizing Ollama

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Apple MLX Framework

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

vLLM for Local Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Text Generation Inference

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Choosing Your Runtime

Chapter 9: Tokenization and Prompt Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

How Tokenization Works

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Vocabulary Size and Efficiency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Tokenizer Choice Matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Measuring Tokenization Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Special Tokens and Formatting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Prompt Templates and System Messages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Token Counting and Estimation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Multilingual Tokenization Issues

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 10: Latency and Throughput Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Time-to-First-Token

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Inter-Token Latency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Tokens Per Second

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Prompt Processing Speed

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Batch Processing and Parallelism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

What Drives Each Metric

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Hardware Bottleneck Identification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Measuring Without Contamination

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 11: Memory Behavior and KV Cache

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

What is the KV Cache

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

KV Cache Growth and Context Windows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Memory Planning and Budgeting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

KV Cache Quantization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Sliding Windows and Attention Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Prompt Caching Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Eviction and Reuse

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Practical Memory Monitoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 12: Designing Sound Evaluation Methodology

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Evaluation vs. Benchmarking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Task Selection and Workload Definition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Representative Test Construction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Prompt Design and Variance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Deterministic vs. Stochastic Evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Temperature and Sampling Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Sample Size and Saturation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Evaluation Framework Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 13: Statistical Rigor and Experimental Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Reproducibility and Seeding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Confidence Intervals for Model Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Statistical Significance Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Variance Sources and Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Measurement Error and Noise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Warm-up Effects and Caching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Normalizing Across Hardware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Reporting Standards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 14: Public Benchmarks and Their Limitations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Multiple-Choice Knowledge Benchmarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Math and Reasoning Benchmarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Code Generation Benchmarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Instruction and Chat Benchmarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Benchmark Contamination

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Overfitting to Benchmarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Aggregate Scores and Their Fallacies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

When to Trust Public Benchmarks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 15: Building Custom Evaluation Harnesses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Evaluation Harness Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Implementing a Basic Harness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Streaming and Telemetry Collection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Automated Experiment Orchestration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Result Storage and Retrieval

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Structured Output Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Comparative Analysis and Reporting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Visualization and Dashboards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 16: Measuring Model Quality in Depth

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Automated Quality Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Factuality Verification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Hallucination Measurement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Instruction Following Assessment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Reasoning Quality Evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Code Execution Verification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Long-Context Fidelity Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Model-as-Judge Evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 17: Workload-Specific Evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Conversational Assistants

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Coding and Development Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Document Analysis and Summarization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Retrieval-Augmented Generation Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Agentic Workflows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Edge and Offline Deployments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Enterprise and Private Assistants

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Embedding and Classification Tasks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 18: Safety, Robustness, and Reliability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Safety Evaluation Methodology

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Adversarial Prompt Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Jailbreak Resistance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Refusal Calibration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Consistency and Reliability Under Load

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Degradation Modes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Monitoring for Production Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Compliance and Policy Checks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 19: Advanced Optimization Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Speculative Decoding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Continuous Batching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Prompt and KV Cache Compression

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Compiler Optimizations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Distillation and Adapters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Model Merging Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Hardware-Specific Optimizations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Distributed and Multi-Node Inference

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Techniques for Improving Performance Without Compromising Quality

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Chapter 20: Decision Frameworks and Production Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Multi-Criteria Decision Frameworks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Model Selection by Workload

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Hardware Procurement Decisions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Cost Modeling and TCO

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Operational Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Case Study: Local Coding Assistant

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Case Study: Private RAG System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Case Study: Edge Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/evaluatinglocalandsmalllanguagemodels.