International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Small Language Models vs Large Language Models: A Comparative Study on Efficiency, Cost, and Enterprise Adoption

Bhumika Shinde [1], Aarti Gawai[2]

Student[1],Department of Information Technology and Computer Science, D.G Ruparel College, Mumbai, Maharashtra, India

Assistant Professor[2] ,Department of Information Technology and Computer Science, D.G Ruparel College, Mumbai, Maharashtra, India

---

Abstract - Language models now support tasks ranging from customer service to code generation, and organizations must decide whether a compact model or a frontier-scale model suits each workload. This paper compares Small Language Models and Large Language Models (LLMs) on three dimensions: operational efficiency, cost, and enterprise adoption. The study is a secondary-data analysis based on published model papers, technical reports, and vendor documentation, supported by a transparent total-cost-of-ownership (TCO) model whose assumptions are stated openly. The analysis shows that SLMs need far less memory and hardware, can run on a single consumer-grade GPU after quantization, and become cheaper than API-based access once monthly request volume passes roughly one million under the assumed conditions. LLMs remain stronger in multi-step reasoning and open-ended problems. Adoption follows organizational maturity: firms with limited ML staff favor managed LLM APIs, while technically mature teams increasingly self-host SLMs. The paper proposes a routing-based hybrid strategy that sends each request to the smallest model able to handle it, and concludes that the choice is a matter of workload fit rather than a contest between two model classes.

Key Words: Small Language Models, Large Language Models, Model Efficiency, Total Cost of Ownership, Enterprise Adoption, Quantization, Inference

1. INTRODUCTION

1.1 Background

The Transformer architecture reshaped natural language processing by making it practical to train very large models on massive text corpora. GPT-3, with 175 billion parameters, showed that a single model could perform many tasks from a handful of examples. This success encouraged a widespread belief that better capability requires ever larger models.

That belief has since been challenged. Open models such as Llama 2, Mistral 7B , and Phi-3 report competitive results at a fraction of the parameter count, mainly through better training data and design. In this paper, **Small Language Models (SLMs)** are models of about 3 to 13 billion parameters that can be served on a single GPU, while **Large Language Models (LLMs)** are models with tens or hundreds of billions of parameters, usually accessed through a vendor API. The boundary is a working convention rather than a formal standard.

1.2 Research Problem

Organizations adopting language models face a practical question: *for a given workload, when is a small model sufficient, and when is a large model worth its price?* The answer affects budgets, response times, data-governance exposure, and the engineering skills a team must hold. Existing work tends to examine models, techniques, or prices separately; decision makers need them side by side.

1.3 Research Objectives

- To compare SLMs and LLMs on memory, hardware, latency, and capability.
- To estimate the total cost of ownership of API-based and self-hosted deployment at different request volumes.
- To describe how different kinds of organizations adopt each model class and what barriers they face.
- To propose a practical framework for combining both classes.

1.4 Research Questions

RQ1: How do SLMs and LLMs differ in memory, hardware, latency, and capability?

RQ2: At what request volumes does self-hosting an SLM become cheaper than paying for an LLM API?

RQ3: How do organizational maturity and workload type shape adoption?

RQ4: How can both model classes be combined in one system?

2. LITERATURE REVIEW

2.1 Scale and Capability

Brown et al. demonstrated that scaling a Transformer to 175 billion parameters produced strong few-shot behavior across many tasks, establishing the case for very large models. The cost of that scale, however, is heavy: such models cannot be held on a single commodity accelerator and are normally reached through hosted services.

2.2 Compact Models

Touvron et al. released the Llama 2 family from 7 to 70 billion parameters, giving researchers open weights across a range of sizes. Jiang et al. reported that Mistral 7B

outperformed Llama 2 13B on all benchmarks they evaluated. Abdin et al. reported that Phi-3-mini, at 3.8 billion parameters, reached about 69% on MMLU, a level they describe as comparable to much larger models, and attributed this mainly to carefully filtered training data. Together these reports suggest that parameter count alone is a weak predictor of usefulness.

2.3 Efficiency Techniques

Three lines of work make small models cheaper to run. Knowledge distillation transfers behavior from a larger teacher to a smaller student. Quantization lowers numeric precision; QLoRA showed that a 4-bit quantized model can be fine-tuned on a single GPU while preserving the task performance of 16-bit fine-tuning. Serving systems such as vLLM use PagedAttention to manage memory and report substantially higher throughput than earlier serving systems at similar latency.

2.4 Cost-Aware and Agentic Use

Chen et al. proposed FrugalGPT, which cascades queries from cheaper to more expensive models and reported large cost reductions at comparable quality. Belcak et al. argue that small models are well suited to the repetitive, narrow sub-tasks that make up most calls inside agentic systems, and that mixed systems using both classes are a natural fit.

2.5 Research Gap

The literature is rich on models and optimization techniques but offers few compact treatments that connect efficiency, cost, and adoption for a decision maker. This paper addresses that gap.

3. RESEARCH METHODOLOGY

The study uses **secondary data** and a **comparative analysis** method. Sources are peer-reviewed and pre-print papers, official technical reports, and vendor documentation on models, serving software, and pricing. The work proceeds in four steps: (1) define the two model classes and the comparison criteria; (2) compare them on memory, latency, and capability using published results and simple calculation; (3) build a TCO model with explicit, adjustable assumptions; and (4) synthesize adoption patterns from the literature and platform documentation. No primary survey was conducted. Because prices and benchmarks change quickly, cost results are presented as illustrative scenarios rather than market quotations.

4. OPERATIONAL EFFICIENCY

4.1 Memory and Hardware

The memory needed to hold model weights is approximately the parameter count multiplied by the bytes stored per parameter. A 7-billion-parameter model at 16-bit precision needs about 14 GB, which fits on a 24 GB consumer GPU such as the NVIDIA RTX 4090, and about 3.5 GB at 4-bit precision. A 70-billion-parameter model needs about 140 GB at 16-bit precision, which requires several data-center accelerators. Fig -1 shows these values. They cover weights only; the attention cache and activations add further memory that grows with context length and batch size.

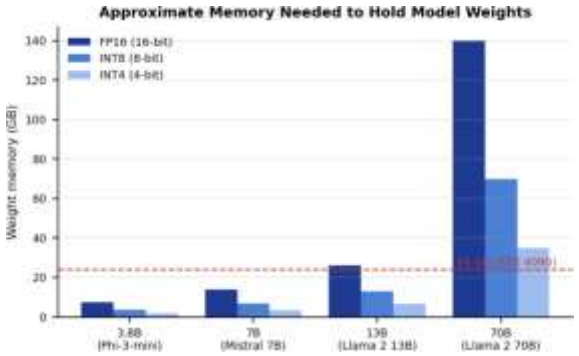


Fig -1: Approximate weight memory by model size and precision.
Source: Author's calculation (parameters × bytes per parameter).

4.2 Latency and Throughput

A smaller model performs fewer computations for each generated token, so on identical hardware it generally responds faster and serves more requests per second. Exact figures depend on hardware, batch size, prompt length, and serving software, so this paper does not quote a single latency number. Optimized servers such as vLLM [8] raise throughput for both classes. API-based LLM use also adds network delay and provider queueing that self-hosted deployment avoids.

4.3 Capability Trade-offs

SLMs perform well on bounded tasks: classification, information extraction, moderation, and question answering about a defined product or document set. Fine-tuning on domain data can narrow the gap further. LLMs keep an advantage on multi-step reasoning, broad knowledge synthesis, and unfamiliar problems, although published results such as these show the gap on standard benchmarks is narrowing. Table -1 summarizes the comparison.

Table -1: Comparison of SLMs and LLMs.

Criterion	SLM	LLM
Typical size	About 3–13B parameters	Tens to hundreds of billions
Hardware	Single GPU or high-end CPU (quantized)	Multi-accelerator cluster or vendor API
Latency	Low; local serving	Higher; network and queueing added
Cost pattern	Upfront hardware, low running cost	Pay per token; grows with volume
Customization	Easy to fine-tune in-house	Limited or costly
Data control	On-premises possible	Data sent to provider
Reasoning	Weaker on multi-step tasks	Stronger on complex tasks
Best suited to	High-volume, well-defined tasks	Open-ended, complex tasks

5. COST ANALYSIS AND TOTAL COST OF OWNERSHIP

5.1 Deployment Options

Three deployment routes are common: (a) consuming a commercial LLM through a pay-per-token API; (b) using managed hosting of open models from a platform provider; and (c) self-hosting an open SLM on owned or rented GPUs. API cost scales directly with usage, whereas self-hosting converts most cost into a fixed upfront investment.

5.2 TCO Model and Assumptions

To make the comparison transparent, a simple model is used with the following illustrative assumptions:

- Each request uses 1,000 tokens (input plus output).
- Blended API price of US$0.002 per 1,000 tokens, representing a low-cost API tier.
- One GPU server costs US$10,000 and can serve 5 million requests per month.
- Running cost is US$600 per server per month (power, hosting, maintenance share).
- Staff time, fine-tuning, and evaluation effort are excluded.

Table -2 gives the results. Break-even equals upfront cost divided by monthly saving.

Table -2: Illustrative cost comparison (US$).

Req. / month	API cost / month	Servers (upfront)	Self-host / month	Break-even	Year-1 net saving
0.1 M	200	1 (10,000)	600	Not reached	−14,800
1 M	2,000	1 (10,000)	600	≈ 7.1 months	6,800
5 M	10,000	1 (10,000)	600	≈ 1.1 months	102,800
10 M	20,000	2 (20,000)	1,200	≈ 1.1 months	205,600

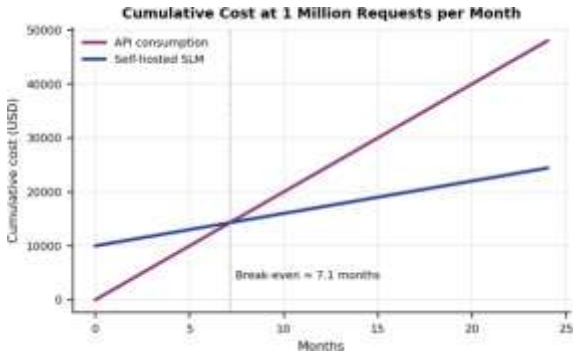


Fig -2: Cumulative cost, API vs self-hosted SLM, at 1 million requests per month.
Source: Author's TCO model (Table -2 assumptions).

Three results follow. First, at low volume (0.1 million requests), the API is cheaper because the fixed cost is never recovered. Second, near 1 million requests per month, break-even falls at about seven months, consistent with a 6 to 12 month range. Third, at 10 million requests the first-year net saving is about US$205,600 under these assumptions. The results are sensitive to price: with a premium-tier API at ten times the assumed rate (US$0.02 per 1,000 tokens), the 1-million-request API bill is US$20,000 per month and payback falls to about half a month. Conversely, if engineering staff time of a few thousand dollars per month is charged to the self-hosted option, the saving at 1 million requests may disappear. Organizations should therefore run this calculation with their own figures.

5.3 Other Cost and Risk Factors

Beyond direct spending, self-hosting keeps data inside the organization, which matters in regulated sectors such as finance, healthcare, and government. It also reduces dependence on a single vendor and makes monthly cost more predictable. Quantization from 16-bit to 4-bit precision cuts weight memory by 75% by arithmetic [7], though quality retained differs by task and must be tested before deployment. Hidden costs include monitoring, model updates, security patching, and evaluation.

6. ENTERPRISE ADOPTION

6.1 Adoption Patterns

The patterns below are synthesized from platform documentation and the literature; they are indicative tendencies, not survey measurements.

- **Large enterprises with established AI budgets** tend to prefer managed LLM APIs, accepting higher unit cost for faster access to new features and for shifting operational duties to the provider.
- **Mid-sized firms and teams with ML engineering skills** increasingly adopt open SLMs using tools such as Hugging Face, Ollama, and vLLM to gain cost control and vendor independence.
- **Cloud-native organizations** using platforms such as AWS SageMaker, Azure Machine Learning, or Google Vertex AI often follow a hybrid pattern: an LLM for reasoning-heavy work and SLMs for repetitive, high-volume tasks.

6.2 Barriers and Drivers

LLM APIs need little technical expertise but bring vendor lock-in and cost that rises with scale. SLM adoption demands in-house skills in deployment, optimization, and monitoring. These barriers are falling as managed platforms offer hosted open models with SLM-like economics. The main drivers toward SLMs are cost at scale, data privacy, low latency, and the ability to run on edge or offline systems.

7. PROPOSED HYBRID FRAMEWORK

Building on cascade ideas [9] and the argument for mixed agentic systems [10], this paper proposes a two-tier routing framework. **Tier 1:** an SLM handles every incoming request that falls into a well-defined task category. **Escalation:** if the SLM signals low confidence, the task needs multi-step reasoning, or the output fails a validation check, the request is forwarded to **Tier 2**, an LLM. Logged escalations can later be used to fine-tune the SLM so that fewer requests need escalation. Table -3 gives workload guidance.

Table -3: Workload guide for model selection.

Workload	Suggested tier
Password-reset or FAQ chatbot	SLM
Document classification, data extraction	SLM (fine-tuned)
Content moderation in fixed scope	SLM
Regulated data that must stay on-premises	Self-hosted SLM
Multi-step financial or legal analysis	LLM
Novel, cross-domain problem solving	LLM
Mixed traffic with unknown difficulty	SLM first, escalate to LLM

Table -4: Key findings by research question.

Research Question	Evidence	Main Finding
RQ1 - Efficiency differences	Memory calculation; literature [3]–[5], [8]	SLMs fit on one consumer GPU; LLMs need clusters or APIs.
RQ2 - Cost break-even	TCO model (Table -2)	Break-even near 1 M requests/month; depends on API price and staff cost.
RQ3 - Adoption factors	Platform documentation; literature	Maturity, privacy needs, and volume drive the choice.
RQ4 - Combining models	Cascade and agentic studies [9], [10]	Route to SLM first; escalate hard cases to LLM.

8. LIMITATIONS AND FUTURE SCOPE

This study relies on secondary data. Hardware and API prices change rapidly, so the TCO figures are illustrative and exclude staff, fine-tuning, and evaluation costs. Benchmark results reported by model authors may not reflect performance on a specific business task. The adoption patterns are qualitative. Future work could add primary data from enterprise surveys or interviews, test routing thresholds on real workloads, and measure energy use and carbon footprint for both classes.

9. CONCLUSION

This paper compared small and large language models on efficiency, cost, and adoption. SLMs require much less memory and hardware, can be served locally, and, under the stated assumptions, repay their upfront cost within roughly a year at about one million requests per month, sooner at higher volume or when compared with premium APIs. LLMs retain their advantage for complex reasoning and novel problems. Adoption follows technical maturity, data-governance needs, and workload volume. The results indicate that the question is not which model class is better but which is right for each task. A routing-based hybrid, using SLMs for frequent well-defined work and LLMs for hard cases, offers a practical way to balance quality, cost, and control.

ACKNOWLEDGEMENT

I would like to thank D. G. Ruparel College, Mumbai, for providing the opportunity and resources to carry out this research. I sincerely thank my research guide, Prof. Aarti Gawai, for her guidance, support, and encouragement throughout this work. I am also grateful to the Department of Computer Science and all my faculty members, and finally to my family and friends for their constant motivation.

REFERENCES

1. A. Vaswani et al., “Attention Is All You Need,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017. arXiv:1706.03762.
2. T. Brown et al., “Language Models are Few-Shot Learners,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020. arXiv:2005.14165.
3. H. Touvron et al., “Llama 2: Open Foundation and Fine-Tuned Chat Models,” arXiv:2307.09288, 2023.
4. A. Q. Jiang et al., “Mistral 7B,” arXiv:2310.06825, 2023.
5. M. Abdin et al., “Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone,” arXiv:2404.14219, 2024.
6. G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” arXiv:1503.02531, 2015.
7. T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “QLoRA: Efficient Finetuning of Quantized LLMs,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023. arXiv:2305.14314.
8. W. Kwon et al., “Efficient Memory Management for Large Language Model Serving with PagedAttention,” in Proc. ACM Symposium on Operating Systems Principles (SOSP), 2023. arXiv:2309.06180.
9. L. Chen, M. Zaharia, and J. Zou, “FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance,” arXiv:2305.05176, 2023.
10. P. Belcak et al., “Small Language Models are the Future of Agentic AI,” arXiv:2506.02153, 2025.