

2.2 Statement

V1: SLMs are sufficiently powerful for language modeling in agentic applications.

V2: SLMs are more operationally suitable than LLMs for agentic systems.

V3: SLMs are more economical than LLMs in most agentic applications due to smaller size.

Conclusion: SLMs are the future of agentic AI . this is not a recommendation, but a reflection of how differences between SLMs and LLMs align with community priorities.

2.3 Elaboration

LLM dominance in agentic design is excessive and misaligned with functional needs.

Most agentic subtasks are repetitive, scoped, non-conversational.

SLMs offer:

- Lower latency
- Lower memory and compute requirements
- Lower operational costs

Adequate performance for constrained tasks

Agentic systems decompose goals into modular subtasks → better fit for specialized SLMs.

Using LLMs for all tasks is a misallocation of resources.

Advocates heterogeneous agentic systems:

Default: SLMs Selective fallback: LLMs
Paradigm shift to SLM-first is both technically superior and ethically desirable due to sustainability concerns.

3. Position Arguments

3.1 SLMs are already sufficiently powerful (supports V1)

Capability trends: SLMs approach LLM performance due to steepening scaling curves.

Benchmarks: Phi-2 (2.7B): On par with 30B, 15× faster

Phi-3 small (7B): Comparable to 70B

Nemotron-H (2–9B): Matches 30B models

SmolLM2 (125M–1.7B): Matches 14B contemporaries, 70B prior-gen

Hymba-1.5B: Outperforms 13B in instruction following, 3.5× faster

DeepSeek-R1-Distill (1.5–8B): Outperforms Claude-3.5 and GPT-4o

RETRO-7.5B: Matches GPT-3 (175B) using retrieval

xLAM-2-8B: SOTA in tool calling

Agentic augmentation: Tool use, verifier feedback, self-consistency

Conclusion: Capability > parameter count for agentic applications.

3.2 SLMs are more economical (supports V3)

Inference cost:7B SLMs are 10–30× cheaper than 70–175B LLMs

Enabled by systems like NVIDIA Dynamo

Fine-tuning agility: LoRA/DoRA → fine-tuning in hours, not weeks

Edge deployment: Systems like ChatRTX enable real-time local inference

Parameter utilization: LLMs sparsely activate parameters
SLMs may utilize their parameters more efficiently

Modular design: "Lego-like" composition: small expert models over monolithic LLMs
Tool calling, routing, caching enhance scalability

3.3 SLMs are more flexible (supports V2, V3)

Cheaper training → easier iteration and adaptation

Supports: Rapid behavior updates
Local regulation compliance
New output formatting

Democratization: Broader access → greater diversity, reduced bias
Stimulates competition and innovation

3.4 Agents expose narrow LM functionality (supports V1, V2)

Most LM calls in agents are constrained and specific.
LLMs are overkill: generalist skills get pruned via prompt engineering.
Fine-tuned SLMs suffice for these narrow functionalities.

3.5 Agentic interactions require close alignment (supports V2)

Strict output format (e.g., JSON, XML) is essential.
SLMs can be fine-tuned to produce consistent, aligned output.
Hallucinations and format deviations from LLMs are undesirable.

3.6 Agentic systems are naturally heterogeneous (supports V2)

Modular architecture enables multiple LMs:
Use SLMs for subtasks
Use LLMs selectively (e.g., top-level orchestration)


3.7 Agentic interactions generate data (supports V2)

Natural fine-tuning dataset from agent usage logs
Enables future replacement of LLMs with expert SLMs
Data is high-quality and aligned with real agent behavior

4. Alternative Views

AV1: LLMs will always perform better on general language tasks
CA1: Scaling laws → larger models = better performance
CA2: LLMs may contain a "semantic hub" for abstraction and generalization

Rebuttal:

A8: Scaling law studies assume constant architecture
A9: SLMs can be fine-tuned effectively
A10: Test-time compute and reasoning scaling is cheaper for SLMs
A11: Agentic decomposition makes abstract generalization less relevant. 
single, abstract general-purpose model becomes less important
AV2: LLM inference is cheaper at scale due to centralization
CA3: LLMs allow better load balancing and endpoint utilization
CA4: Cost of infrastructure and ML ops for SLMs is high

Rebuttal:

A12: Inference modularization reduces CA3
A13: Infrastructure setup costs are falling
AV3: LLM-centric world has momentum and head start
Industry inertia favors LLMs. Acknowledge its plausibility, but claim SLM advantages (A1–A7) can overcome it.

5. Barriers to Adoption

B1: Investment bias toward LLM infrastructure
B2: SLMs evaluated on generalist benchmarks, not agentic tasks
B3: Lack of marketing/awareness about SLMs
Not technological limitations — rather practical inertia.
Tools like Dynamo and research help break these barriers.

6. LLM-to-SLM Agent Conversion Algorithm

- S1: Log agent calls (prompts, outputs, tool invocations)
Use secure, anonymized logging pipelines
- S2: Curate and filter data
Remove sensitive data (PII, PHI)
Use masking/paraphrasing tools
- S3: Cluster tasks Use unsupervised learning to identify repeating request types
- S4: Select SLMs per task
Consider performance, context size, licensing, deployment footprint
- S5: Fine-tune SLMs per task
Use PEFT (LoRA, QLoRA) or full finetuning
Optionally distill from LLM outputs
- S6: Iterate and refine
Periodically update models with new data
Loop back to S2 or S4 as needed

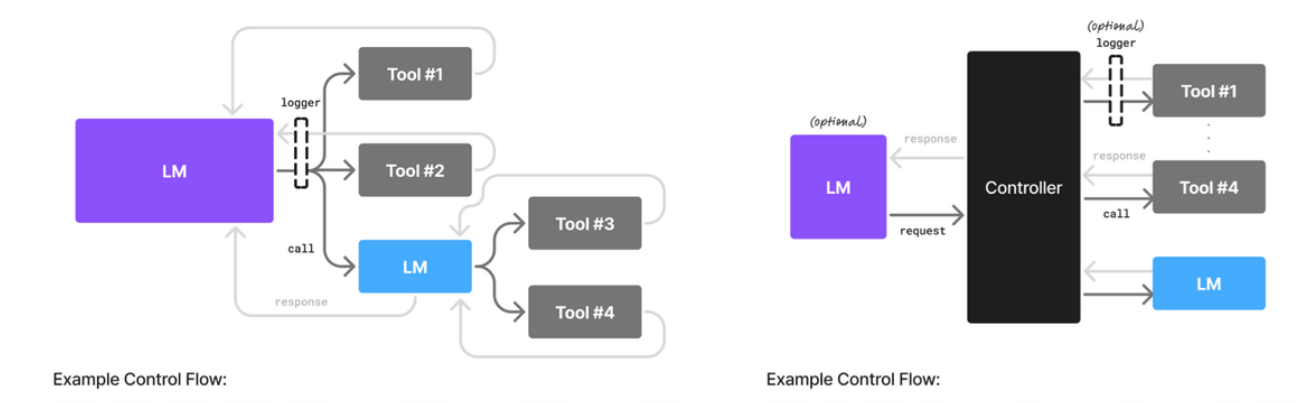


Figure 1: An illustration of agentic systems with different modes of agency. Left: Language model agency. The language model acts both as the HCI and the orchestrator of tool calls to carry out a task. Right: Code agency. The language model fills the role of the HCI (optionally) while a dedicated controller code orchestrates all interactions.

Left (Language Model Agency): The language model (LM) directly handles both user interaction and tool orchestration. It decides which tools or other LMs to call. Simpler but less modular.

Right (Code Agency): A central controller manages all logic and tool calls. The LM is used optionally for interface or subtasks. This setup is more modular, scalable, and production-friendly.

Key Difference: LM is in control on the left; controller code is in control on the right.