ChatGPT

Small Language Models are the Future of Agentic AI

Title: *Small Language Models are the Future of Agentic AI*
Authors: Peter Belcak; Greg Heinrich; Shizhe Diao; Yonggan Fu; Xin Dong; Saurav Muralidharan; Yingyan Celine Lin; Pavlo Molchanov
Venue, Year: arXiv preprint (under review), 2025
DOI: 10.48550/arXiv.2506.02153
URL: https://arxiv.org/abs/2506.02153
Code/Data: *No official code or dataset released.* (Position paper with conceptual framework and case studies)
Tags: Agentic AI; Small vs Large Language Models; Efficiency; Modular AI; AI Agents

TL;DR

The authors posit that **Small Language Models (SLMs)** – language models that are small enough to run on consumer devices (usually <10B parameters) – are **sufficiently powerful, more operationally suitable, and far more economical** for many tasks in AI agent systems, compared to large language models. They argue that SLMs can handle the narrow, repetitive subtasks typical of *agentic AI* with lower cost and latency, making SLM-first architectures the future of agent systems [1]. In cases needing open-ended reasoning, they advocate using SLMs by default and only **calling large models sparingly** when necessary [2].

Key Contributions

- **Position Statement:** Formulates a clear stance that **small LMs should replace large LMs in many agent applications**, claiming that SLMs are *"the future of agentic AI"* [1]. The paper identifies three core tenets (V1–V3) asserting SLMs' sufficiency, suitability, and cost-efficiency in agent systems [3].
- **Evidence and Arguments:** Presents **five supporting arguments (A1–A5)** with empirical and conceptual backing. These highlight recent advances in SLM capabilities (e.g. 2–10B models achieving parity with 30–70B models on key tasks) [4], operational benefits of SLMs (latency, fine-tuning speed, etc.), and the nature of agentic tasks (narrow scope, strict formatting) favoring specialized smaller models [5] [6].
- **Acknowledgment of Barriers:** Discusses current industry practice and **barriers to SLM adoption**. The authors recognize that today's agents almost exclusively rely on single generalist LLM APIs [7] and that significant **business inertia and infrastructure investment** are tied to the LLM-centric model [8]. This context motivates their push for change despite the status quo.
- **Conversion Algorithm:** Proposes a practical **LLM-to-SLM conversion framework** for existing agentic applications [9]. This six-step process (S1–S6) details how to log agent queries, curate and cluster data, select appropriate SLMs, fine-tune specialists, and iteratively refine an agent's model ensemble [10] [11]. The goal is to **"painlessly" migrate** from a monolithic LLM to a set of SLM experts without losing performance.
- **Case Studies & Discussion:** Provides short **case studies (Appendix B)** estimating how much of the LLM functionality in popular open-source agents could be replaced by SLMs, to concretize the

potential impact. Finally, the paper **calls for community discussion and feedback**, pledging to publish correspondence on an open forum [12]. This reflects the authors' intent to spark dialogue and invite critique of their position.

Method

Approach: This work is a **position paper** that builds its case through logical argumentation and evidence from existing research. The authors first define key terms (**SLM** vs **LLM**) and state their position in terms of three claims (V1, V2, V3) [13]. They then enumerate five arguments (A1–A5 in Section 3) supporting those claims, each grounded in recent findings or characteristics of agentic systems. For example, they cite results from multiple papers to show SLMs' capabilities (A1) and cost advantages (A2), and examine the structure of agent tasks to argue why SLMs are a better fit (A4, A5) [5] [6]. In Section 4, they **address counterarguments** (e.g. the notion that only large models have the necessary breadth of knowledge) and provide rebuttals [14].

LLM-to-SLM Migration Algorithm: In Section 6, the paper shifts to a more procedural contribution – an outline of how to **replace an agent's LLM with a network of SLMs**. The proposed method involves: **(S1)** instrumenting the agent to securely log all prompts and outputs over time [10]; **(S2)** filtering and anonymizing this collected data to ensure privacy and quality; **(S3)** clustering the queries and behaviors into distinct task categories; **(S4)** selecting suitable SLM models for each task (based on capabilities, context length, licensing, etc.); **(S5)** fine-tuning each chosen SLM on the task-specific data (using efficient fine-tuning techniques like LoRA to save compute) [11]; and **(S6)** deploying these specialist models with a routing mechanism, then iterating as new data arrives [15]. This algorithm provides a **practical roadmap** for organizations to gradually transition agentic systems from one-size-fits-all LLMs to a heterogeneous collection of small, specialized LMs.

No new model training or novel algorithm is introduced beyond this *workflow*. Instead, the methodology synthesizes best practices (data logging, clustering, fine-tuning) into a recipe for implementing the paper's vision in real-world settings.

Data

Data Sources: This paper does not introduce a new dataset. Instead, it leverages *existing benchmark results and usage data* to support its arguments. For instance, the authors recap performance metrics from prior works – e.g., a 2.7B parameter model (Phi-2) achieving code generation on par with a 30B model while running ~15× faster [16]. They compile such comparisons from multiple sources to demonstrate that modern SLMs approach LLM capabilities. These data points (on commonsense reasoning, tool use, code writing, etc.) come from published research and are cited in-text rather than re-evaluated here.

Proposed Data for Fine-Tuning: In the conversion algorithm, the authors suggest collecting an **agent's own interaction logs** as training data for SLM specialists [10]. This includes prompts, tool call inputs/outputs, and agent responses gathered during normal operation. By curating and filtering this **organic usage data**, one can obtain task-specific datasets for fine-tuning small models to each subtask. The paper emphasizes privacy and quality filtering in this process (removing personal or sensitive info, using automated tools to mask or paraphrase private data) [17] [18]. Thus, while no new dataset is released, the work outlines how organizations can **leverage their own data** to train SLMs tailored to their agents.

Results

The paper's **findings** are presented as a set of reasoned arguments supported by evidence. Key results and claims include:

- **SLM Capability vs LLM:** Recent small models (ranging ~2B–10B parameters) have demonstrated performance **comparable to much larger LLMs** on many relevant tasks. For example, Microsoft's Phi-2 (2.7B) matches 30B LLMs on commonsense reasoning and code generation tasks while being ~15× faster [16]. Similarly, a 7B model (Phi-3 Small) achieved language understanding on par with a 70B model [16]. These examples support the claim that **SLMs are now sufficiently powerful** to handle the kinds of subtasks agents need (argument A1).

- **Efficiency and Cost: Serving SLMs is dramatically cheaper and faster** than serving LLMs. The paper notes that hosting inference for a 7B model can be *10–30× less costly in latency, energy, and FLOPs* than a 70B–175B model [19]. This means agents can respond in real-time at scale using SLMs, with much lower infrastructure burden. Moreover, **fine-tuning and updating SLMs is far quicker and more feasible**: small models can be fine-tuned with only a few GPU-hours (often overnight) as opposed to the weeks-long fine-tuning cycles for giant models [20]. These efficiency gains (argument A2) translate into operational cost savings and the ability to iterate rapidly on agent improvements.

- **Flexibility & Democratization:** Because of their low cost and smaller size, **SLMs offer greater operational flexibility** (argument A3). It becomes practical to maintain a *portfolio of specialized models* for different functions within an agent, and to retrain or swap them out as needs evolve [21]. The authors highlight that this enables *rapid adaptation* – e.g. quickly supporting new user requests, output formats, or complying with changing regulations by fine-tuning a niche model [22]. An important side effect of this flexibility is **democratization**: when models can be small and cheap, **more individuals and organizations can afford to develop and deploy language models** for agents [23]. Lowering the resource barrier means the community of agent developers can be larger and more diverse, potentially reducing systemic biases and fostering innovation through competition [24].

- **Narrow Task Focus: Agentic applications typically require only a narrow slice of a language model's full capabilities**, which supports using smaller specialized models (argument A4). In an agent, the LM is often constrained to very specific duties via prompting and a fixed tool interface. The paper points out that a general-purpose LLM, which contains a "large palette of skills," is in practice *restricted to a small section of those skills* when deployed in an agent [5]. Instead of using an oversized model and then steering it to act narrow, one can use an appropriately fine-tuned SLM that **directly excels at the needed narrow task**. The authors argue this would suffice for the agent's purposes while also reaping the efficiency benefits.

- **Alignment and Reliability:** Agentic AI workflows demand **strict adherence to format and behavior**, which SLM specialists can satisfy more easily than a giant model (argument A5). Because agents often interact with external tools or APIs, the LM must output in very specific formats (e.g. a JSON with certain fields, or code with exact syntax) [25]. Large models, with their vast and varied training, might occasionally deviate or "hallucinate" outputs that violate the expected format. A smaller model fine-tuned *only* on the required format and task will be **more predictable and tightly**

aligned with these requirements [6]. In other words, using an SLM for each structured subtask reduces the risk of errant or off-format responses, increasing reliability in the overall agent.

Overall, the paper's results reinforce that **SLM-first agent architectures can maintain strong performance while drastically improving efficiency**. The authors conclude that a "Lego-like" assembly of many small expert LMs is *"cheaper, faster to debug, easier to deploy, and better aligned"* with real-world needs, providing *"the best path forward for cost-effective, modular, and sustainable agentic AI"* [26].

Strengths & Limitations

Strengths: This paper tackles a timely and practical question—how to build AI agents more efficiently—and provides a well-argued perspective backed by concrete examples. Its strengths include: - *Comprehensive Argumentation:* The authors examine the issue from multiple angles (capability, cost, flexibility, alignment), making the case robust. Each claim (A1–A5) is supported by references to recent models or theoretical reasoning, lending credibility. For instance, they cite specific SLM achievements (Phi, Nemotron, etc.) to substantiate power and efficiency claims [16] [19]. - *Clarity of Position:* The paper clearly states its thesis that SLMs should be favored in agents, and repeatedly ties back evidence to this central position. The use of defined viewpoints (V1–V3) and numbered arguments (A1–A5) helps structure the narrative and makes it easy to follow the logic. - *Practical Impact:* Unlike purely theoretical position pieces, this work offers actionable guidance (the migration algorithm). This shows the authors have considered *"how do we implement this in practice?"* — a valuable strength for influencing real-world adoption. The step-by-step conversion plan (logging data, clustering tasks, fine-tuning models) serves as a recipe that practitioners could try to follow [10] [11]. - *Acknowledging Reality:* The paper doesn't ignore the current dominance of LLMs or pretend switching to SLMs is trivial. By discussing industry investment in LLM infrastructure and organizational inertia, the authors demonstrate a nuanced understanding of the landscape. They position their idea as a necessary future direction, even if today's practices lag, which strengthens their credibility (showing they are not naïve to practical challenges).

Limitations: As a position paper, this work also has some limitations: - *Lack of Original Experiments:* The paper does not present new experimental results or benchmarks of an SLM-based agent. The evidence provided, while compelling, is all second-hand (drawn from other papers or general knowledge about agent systems). Readers might question how an actual fully SLM-powered agent compares to an LLM agent in practice, since that head-to-head comparison was not directly executed here. The arguments would be bolstered by a concrete case study or prototype demonstration. - *Scope of Claims:* The recommendation to prefer SLMs is persuasive for many *narrow and structured tasks* (e.g. code generation, data extraction, etc.), but the authors concede that **some situations still require LLMs**. For open-ended dialogue or very complex reasoning, a large model's broader knowledge may be necessary [14]. The paper advocates a hybrid approach in those cases (i.e. having an LLM as a backstop), which implicitly limits its claim – SLMs are not a panacea for *all* tasks. This nuance is appropriate, but it means the "SLM-first" approach still relies on careful identification of when a big model is needed. - *Engineering Complexity:* Re-architecting agents to use a collection of specialized SLMs (with a routing mechanism) could introduce complexity in development and maintenance. The paper portrays the modular approach as beneficial, but managing "N small models" might be harder than managing a single API endpoint in terms of engineering effort. Issues like model orchestration, consistency, and cumulative error handling in a multi-component system are not deeply explored. This is a potential challenge not fully addressed in the paper's discussion of drawbacks. - *Business and Ecosystem Factors:* While the authors acknowledge business barriers, there are other ecosystem limitations worth noting. For example, the largest, most capable models (GPT-4, etc.) are often proprietary

and require API use; smaller open models might lag on some cutting-edge capabilities or require extensive fine-tuning data that only big players have. The paper's vision assumes organizations can obtain or train SLMs that are sufficiently good. In reality, access to high-quality small models or the expertise to fine-tune them may be a bottleneck for some. The work could be strengthened by more discussion on how smaller players can obtain these SLMs (though it does cite community-driven models like those in references).

In summary, *"Small Language Models are the Future of Agentic AI"* provides a strong conceptual case and practical starting points, but it remains to be seen how well its ideas pan out in real-world agent deployments. The arguments are thought-provoking, but convincing the industry will likely require more empirical validation and tooling support for SLM-centric agent architectures.

Connections

This paper connects with **several ongoing trends and discussions** in the AI community:

- **Modular AI & Composite Systems:** The idea of composing multiple specialized models instead of relying on one general model aligns with a broader push toward modular AI system design. The authors reference prior work that advocates for *"composite agentic systems"* [27] and note that mainstream frameworks are slowly incorporating heterogeneous model usage. This echoes the concept of *mixture-of-experts* and other modular architectures where different models handle different tasks. It's a shift from the end-to-end monolithic philosophy to a more engineering-driven assembly of expert components – akin to microservices in software design, but for AI models.
- **Efficient and Accessible AI:** The SLM-first view ties into the movement for more **resource-efficient AI**. In recent years, there's growing interest in techniques for compressing models, distillation, and training small models that perform well. For example, the authors cite *SmolLM2 (2025)*, a project on data-centric training of a small LM [28], and *Tiny Transformers* for specific tasks [29]. These efforts indicate a community recognition that not everyone can run 100B+ parameter models, and that making smaller models *work better* is important for democratizing AI. NVIDIA's own work on inference systems (like **NVIDIA Dynamo for SLM inference** [30] and **ChatRTX for local deployment** [31]) show industry alignment with optimizing for smaller models in production.
- **Industry Perspectives:** There have been informal discussions and articles about *"Small vs Large LMs"* (the authors cite a few 2024 blog posts in their references [32]). This paper can be seen as giving a rigorous, research-backed voice to those debates. It provides a framework to think about *when a smaller model is "good enough"* and encourages a cost-benefit analysis rather than blindly using the biggest model available. This conversation is particularly relevant for startups and enterprises concerned with AI deployment costs and wanting to run models on-premise or on-edge devices.
- **Related Academic Work:** The paper's authors build on some of their **own prior research** – for instance, Belcak et al. (2024–2025) have worked on *MiniFineTuning* (low-data adaptation for LMs) [33], which is directly relevant to fine-tuning SLMs for specific tasks. They also cite **Flextron (ICML 2024)**, which is about a flexible multi-LLM system [34], and other community efforts like HuggingFace's initiatives on small models [35]. This situates the paper within a network of research aiming to make AI systems more *flexible, adaptive, and efficient*. In particular, it contributes a high-level vision that complements technical work on model compression and efficient training.
- **AI Alignment and Policy:** Interestingly, by touching on democratization and bias, the paper connects to discussions in AI ethics and policy. If more groups can train their own models (because models are smaller), we might see a diversification in AI behavior and potentially less concentration of power among a few AI providers. This could reduce systemic biases (as the authors suggest) [36],

but also raises questions of **standards and safety** when many custom models are deployed. While not the focus of the paper, this connection to alignment (ensuring models behave as intended) and societal impact is an important backdrop.

In essence, “SLMs are the Future of Agentic AI” is a logical next step in conversations about sustainable AI. It complements technical research on small models and engages with practical concerns of AI deployment, potentially influencing both the **research direction (toward improving small models)** and **industry best practices (toward more modular, efficient agents)**.

Replication Notes

Because this is a position paper without new experimental results, traditional replication isn’t applicable in the usual sense (there’s no novel model to train or dataset to evaluate exactly as in the paper). However, one could **validate the paper’s claims through case studies or prototypes**: - **Reproducing an Agent with SLMs:** A potential replication exercise would be to take an existing agent (for example, an open-source agent like Auto-GPT or LangChain-based agent that currently uses an LLM) and attempt the **LLM-to-SLM conversion** as described. This would involve collecting the agent’s interaction logs, clustering tasks, choosing small models (perhaps open-source models like LLaMA-7B or smaller), fine-tuning them for each task, and then measuring the agent’s performance and costs. By doing this, one could directly observe how the SLM-powered agent compares to the original LLM-powered agent in success rate, speed, and cost. The paper gives a roadmap for this, but to truly replicate/confirm their position, implementing it and sharing results would be valuable. - **Benchmarking Specific Tasks:** Another approach is to replicate the kind of comparisons cited in the paper. For example, the authors mention Phi-2, Hymba-1.5B, xLAM-8B, etc., and their performance vs larger models [37] [38]. One could run a mini-benchmark: pick a set of agent-relevant tasks (say, a knowledge query, a code generation task, a tool-using task) and compare a state-of-the-art 7B model against a 70B model (or GPT-4 via API) on those tasks. This would empirically test the “SLM sufficiency” claim. The expectation, per the paper, is that the 7B model will perform similarly on the narrow tasks, albeit maybe not on open-ended ones. - **Reproduce Case Study Analysis:** In Appendix B, the authors estimated what fraction of components in some open agents could be handled by SLMs. While we don’t have those exact numbers in the summary, one could replicate that analysis by inspecting popular agent frameworks. For instance, break down an agent’s workflow (planning, web browsing, code writing, etc.) and assess for each if there is a small model that could do it. This doesn’t produce a numeric result easily, but it’s a qualitative replication of their reasoning.

One challenge in any replication is **measuring “agent performance”** comprehensively. Agents often involve complex, multi-step tasks and tool use, so one might need to craft specific evaluation scenarios. Additionally, sourcing or training the best SLMs for each niche task could be non-trivial (the paper assumes these SLMs are available or can be fine-tuned, which in practice requires some expertise).

Finally, because the paper’s claims span efficiency, one should also attempt to measure not just accuracy/performance but actual *latency and cost*. For a fair replication, monitoring GPU usage, response times, and dollar costs when using a large model vs an ensemble of small ones would provide concrete evidence to either support or challenge the paper’s economic arguments.

In summary, “replicating” this work means **applying its recommendations and seeing if the expected benefits materialize**. The authors have essentially issued a call to action, and replication would be taking up that call on a small scale to report empirical outcomes.

Questions for the Authors

1. **Threshold for LLM usage:** In a heterogeneous agent that is mostly SLM-driven, how do you decide when to invoke a large model? Did you envision an automatic gating mechanism based on task complexity/confidence, or would it be manually defined thresholds? More broadly, *what tasks do you believe currently still* require *an LLM?* (E.g., open-ended conversation, complex novel problem-solving?) Clarifying this would help practitioners know the boundaries of SLM-first approaches.
2. **Maintaining Multiple Models:** Managing an ensemble of specialized SLMs could introduce overhead. What are your thoughts on the **engineering complexity** of maintaining many small models (each needing updates, evaluation, possible drift over time) versus a single large model? Do the benefits clearly outweigh this added complexity, and have you observed any best practices for maintaining such modular systems (e.g., validation pipelines, model routing logic) in a production environment?
3. **Quality of Fine-Tuning Data:** The conversion algorithm leans on an organization's ability to log and fine-tune on its own data. In cases where an agent is new or doesn't have a lot of usage data, how would you recommend proceeding? Is there a role for *synthetic data generation* or using foundation model knowledge to bootstrap SLM specialists? Essentially, what's the **minimum data regime** in which you think SLM specialization is viable?
4. **Generality vs. Specialization:** SLM specialists excel at narrow tasks, but how do we ensure the agent as a whole doesn't lose the ability to integrate knowledge across domains? For example, an agent might need to carry context from a document summarization (one SLM) into code generation (another SLM). Did you encounter any issues or thoughts around **coordination between SLMs** or shared memory so that the agent's overall performance remains coherent?
5. **Future of Large Models:** If your vision is realized and many tasks shift to SLMs, what do you foresee as the role of giant LLMs in the long term? Do they become mostly *research tools* or *training teachers* (for distillation), or do you think they'll always be needed as occasional "big guns" for certain users? It would be interesting to hear how you see the LLM landscape evolving if the community embraces SLM-first for deployments.
6. **Evaluation Metrics:** You advocate for cost-effective deployment, which implies new metrics beyond accuracy – e.g., *cost-performance ratio, energy usage, latency*. Do you think the research community needs to start reporting these metrics more systematically? Have you considered any specific **benchmark or metric** that could help formalize the comparison between an LLM-centric agent and an SLM-first agent (for instance, something like "task success per dollar" as a metric)?
7. **Security and Privacy:** Small models running on-device (SLMs) have the advantage of data privacy (avoiding sending data to central APIs). However, they might also be more vulnerable to being tampered with on edge devices or could leak information if not carefully handled. Did you consider any **security implications** of moving to SLMs? For example, verifying that each specialized model doesn't inadvertently retain sensitive info (since you fine-tune on potentially private data) or that the ensemble can't be more easily attacked than a centralized service?
8. **Reception from Industry:** Since publishing this position (or as you have circulated it under review), what has been the **response from industry practitioners**? Are companies actually attempting to implement SLM-first strategies, or do they express skepticism? Any feedback or case studies since then would be enlightening to see how the idea is catching on (or what objections are raised).
9. **Comparison to MoE (Mixture of Experts):** Your approach of many small models each handling a subset of tasks is reminiscent of Mixture-of-Experts in some ways (where different experts handle different inputs). However, classical MoE is usually within one large model architecture. How would

you compare your vision of an *external ensemble of SLMs* to the *internal MoE architectures* that some large models have? Do you think one has advantages over the other in the agent context?

10. **Environmental Impact Quantification:** You mention sustainability; have you tried to quantify how much energy/carbon reduction might be achieved by shifting a certain percentage of workloads to SLMs? It would be interesting if there are any estimates (even rough) of the environmental impact if, say, half of current LLM API calls were instead handled by local SLMs.

These questions aim to probe clarifications and future directions, building on the thoughtful discussion the paper has started.

Key Quotes (from the paper)

> *"While LLMs offer impressive generality and conversational fluency, the majority of agentic subtasks in deployed agentic systems are repetitive, scoped, and non-conversational — calling for models that are efficient, predictable, and inexpensive."* 39

> *"The above-mentioned 'Lego-like' composition of agentic intelligence — scaling out by adding small, specialized experts instead of scaling up monolithic models — yields systems that are cheaper, faster to debug, easier to deploy, and better aligned with the operational diversity of real-world agents. When combined with tool calling, caching, and fine-grained routing, SLM-first architectures appear to offer the best path forward for cost-effective, modular, and sustainable agentic AI."* 26

Personal Rating

Rating: 8/10 – *This paper provides a compelling and forward-thinking perspective that challenges the status quo. The arguments are well-supported with current data and the inclusion of a migration framework adds practical relevance. It loses some points only because it lacks an empirical demonstration of a fully SLM-based agent, and thus leaves certain implementation questions open. Nevertheless, as a position paper, it succeeds in sparking an important conversation about efficiency and modularity in AI agents.* The clarity of writing and logical organization (V1–V3, A1–A5) make it accessible and convincing. I expect this work to influence how researchers and companies consider deploying language models, especially as the costs of large models continue to be a concern.

Presentation Outline (Beamer Slides)

(Design note: The presentation uses Bowling Green State University colors – a theme with Orange, Brown, and White for a professional academic look.)

Slide 1: Title

- **Small Language Models are the Future of Agentic AI**
  (Belcak et al., NVIDIA Research, 2025)
- *Presentation for [Course/Conference/Group]*

- **Presenter:** Your Name, Bowling Green State University
  (BGSU-themed title slide with orange background, white title text, and brown accents.)

Slide 2: Motivation

- **Agentic AI is on the rise:** Over half of large IT companies use AI agents (automated decision-making systems) as of 2024, with the sector valued at \$5.2B and projected to reach \$200B by 2034 [40]. There's huge interest in agents that can perform tasks autonomously.
- **Current standard – Big Models:** Today, most agents call a *cloud-hosted Large Language Model (LLM)* for their intelligence [7]. One giant model handles all tool use, planning, etc., via an API (e.g., calling GPT-4 for everything).
- **Pain Points:** This LLM-centric approach is **expensive and inefficient** – it incurs high latency, monetary cost, and depends on massive infrastructure (tens of billions of dollars in data centers) [41]. Also, agents often only need simple, repetitive operations, not the full power of a giant model.

Slide 3: Background

- **SLM vs LLM:** *Small Language Model (SLM)* – a model that *fits on consumer hardware* and runs with low latency (generally < ~10B parameters as of 2025) [42] [43]. An *LLM* is any larger model that doesn't meet those criteria (requires server-scale compute) [44].
- **Agentic AI systems:** AI *agents* break complex goals into sub-tasks, using an LM plus tools (APIs, code) to accomplish these tasks [45]. Example: an agent might parse an email, then call a calendar API – each step is a sub-task.
- **Status Quo Architecture:** Typically, one monolithic LLM handles all sub-tasks and decision-making for the agent [7]. The agent sends every request to the same large model, which is overkill for many simple tasks. *Illustration – Two agent architectures [46]: Left: Traditional single-LLM agent (the LLM handles both the user interface and tool orchestration). Right: A modular agent with a code-based controller coordinating multiple specialized SLMs. The latter design shows how smaller models, each expert at a narrow task, can replace the one big model in an agent pipeline. The controller (brown box) routes queries to the appropriate SLM or tool, enabling efficiency and specialization.*

Slide 4: Method (Authors' Approach)

- **Position & Claims:** The authors formally posit that SLMs are *capable enough* (V1), *more suitable operationally* (V2), and *more economical* (V3) for agent tasks [3]. They structure the paper around these three claims, supporting each with arguments and evidence.
- **Arguments (A1–A5):** Five key arguments underpin the position:
- *A1:* SLMs today have sufficient power for agents' needs (evidence: recent 2–10B models rival performance of much larger ones on relevant tasks).
- *A2:* SLMs are much cheaper and faster to run, which is critical for practical deployment.
- *A3:* SLMs offer flexibility – easy to fine-tune, adapt, and deploy in various environments (edge devices, etc.).
- *A4:* Agents only use a narrow slice of LM functionality, so a specialized small model can cover it.
- *A5:* Agents demand strict output formats/behavior alignment, easier to enforce with SLMs.
- **Defending & Discussing:** The paper also addresses counterpoints (e.g. "Don't we need LLMs for understanding?") and acknowledges business realities (existing investment in LLM infrastructure).

Finally, it calls on the community to weigh in, indicating the authors opened a forum for feedback [12].

Slide 5: LLM-to-SLM Conversion Plan

- **Migrating Agents to SLMs:** To help practitioners move to an SLM-first architecture, the authors outline a **6-step conversion algorithm**:
- *Log usage data:* Instrument the existing agent to record all prompts, outputs, and tool calls (securely and with privacy) [10]. This yields real examples of what the agent needs to do.
- *Curate & filter data:* Clean the collected data (remove sensitive info, anonymize, and ensure quality) [47] [18] for safe model training.
- *Cluster tasks:* Analyze the data to identify clusters of similar requests or actions [48]. These clusters represent distinct sub-tasks the agent performs (e.g. "summarize email", "generate code snippet").
- *Select SLMs:* For each task cluster, choose a candidate small model that could handle it [49]. Consider model capabilities (does it do coding? reasoning?), context length, and resource footprint. (Many open-source SLMs exist as starting points.)
- *Fine-tune specialists:* Fine-tune each chosen SLM on examples of its specific task [11]. Use efficient fine-tuning methods (like LoRA) to keep this fast and low-cost. If data is sparse, knowledge distillation from the original LLM can help teach the SLM [50].
- *Deploy & iterate:* Integrate these specialized SLMs into the agent with a routing mechanism (a controller decides which model handles which query). Continuously collect new data and retrain or adjust models as needed [15] to improve over time.
- **Outcome:** The agent transitions from relying on one large model to using a *team of small models*, each expert at a role. This promises lower running cost and easier debugging (you can pinpoint which component needs improvement).

Slide 6: Key Results / Arguments

- **SLMs can match LLMs (on specific tasks):** New small models (~7B params) achieve performance on par with 30B–70B models in tasks like code generation and commonsense Q&A [16]. *Example:* An 8B model from Salesforce outperforms GPT-4 on a tool-using benchmark [38], and a 1.5B model (Hymba) beats 13B models on instruction following [37]. ⇒ **Agent tasks don't always need a giant model.**
- **Major efficiency gains:** Using SLMs yields **10×–30× lower inference cost** (latency, energy) compared to a 100B+ model [19]. Smaller models also fine-tune in hours instead of days [20], enabling quick updates. ⇒ **Agents run faster and cheaper**, and can be updated overnight with new data or fixes.
- **Better operational fit:** SLMs enable a *modular, flexible system*: you can deploy models locally (on-device) for data privacy and offline use [31]; add or swap out models as new skills are needed [26]. More people/teams can train their own SLMs (lower barrier), **democratizing** the technology [23]. By contrast, one huge LLM is a black-box monolith – hard to adapt or debug.
- **Tailored to agent needs:** Agents typically perform *repetitive, structured tasks*. SLMs can be *fine-tuned to those exact formats and requirements*, making them reliably output what the agent's tools expect (e.g., well-formed JSON) [6]. A big LLM has a whole "palette" of capabilities it won't use, and might even deviate from the script. ⇒ **Specialized SLMs hit the sweet spot**: just the needed functionality, nothing extra, and thus **easier to control**.

Slide 7: Implications

- **Cost & Accessibility:** *Adopt SLMs for cost-effective deployment.* Organizations running AI agents can **dramatically cut costs and latency** by using small models wherever possible [19]. This is especially important for real-time agents or those running on-device – it makes AI assistants more accessible (no need for expensive cloud calls) and scalable.
- **Modular Agent Design:** *Design agents with a heterogeneous model approach.* Rather than one model to rule them all, **use SLMs for routine, narrow tasks and reserve LLM calls for the rare complex query** [2]. This modular strategy improves maintainability (each model is simpler) and lets you optimize each part. It's like having a toolbox of models – the right tool for each job.
- **Rapid Specialization:** *Leverage SLM agility for quick iteration.* Need your agent to support a new feature? Fine-tune a small model overnight on relevant data and plug it in. The ability to **train/update SLMs quickly** means agents can evolve faster to meet new requirements or regulatory changes [21]. Smaller models also mean more teams (even those with limited resources) can build their own tailored agents, spurring innovation.
- **Sustainability & Ethics:** Shifting to smaller models where possible can reduce the environmental footprint of AI (less energy per query). It also spreads out AI development – when more players can develop models, we get a diversity of perspectives (potentially mitigating biases from having only a few giant models in use) [23]. In sum, **SLM-first is not just an engineering choice, but a step toward more sustainable and inclusive AI**.

Slide 8: Q&A

- **Questions?**
  (Thank the audience and invite discussion.)
- *(Potential discussion points: challenges in implementing SLM-first, particular use-cases that might or might not work well, how to identify tasks for SLMs, etc.)*
- **Contact:** *[Your Email]* for follow-up.

---

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 Small Language Models are the Future of Agentic AI.pdf
file://file_00000000569461f7b68253048032b1ab