Adaptive Minds: Empowering Agents with LoRAs as Tools

Anonymous ACL submission

Abstract

We introduce a new framework, **Adaptive Minds**, in which LoRA adapters are treated as callable tools that a base language model can dynamically select and invoke. We hypothesize that, when adapters are trained to provide strong domain-specific gains and are exposed with clear metadata, a base model can reliably route queries to the appropriate expert, effectively aggregating the benefits of many specialized adapters within a single framework. Within this framework we study both single-step semantic routing and multi-step agentic reasoning. In simple routing the base model selects a single specialist and generates the answer; in the multi-step regime the agent can iteratively invoke multiple adapters alongside other tools (external APIs, retrieval, or execution environments) and reason over their outputs across multiple steps. This reframes adapters as modular skills that can be invoked during reasoning rather than statically applied. In our evaluation, the routing layer reaches 98.3% accuracy on a 30-adapter library, well-trained specialists provide $+4.6$ to $+84.0$ pp of strict-scorer gain across nine task families under a single shared recipe, and the AM router aggregates these gains within ± 5 pp of the direct specialist. Our findings suggest that the effectiveness of this approach depends on the quality and specialization of individual adapters, and that enabling flexible orchestration of many such experts can significantly expand the practical capabilities of language model agents, moving toward more general, tool-augmented intelligence.

1 Introduction

Recent advances in language models have produced two powerful but largely disconnected capabilities: parameter-efficient specialization through adapters (Houlsby et al., 2019; Hu et al., 2022), and tool-augmented reasoning through agents (Yao et al., 2023b; Karpas et al., 2022). LoRA adapters are lightweight parameter updates that specialize a shared base model without requiring a full copy of the model, a property driving the rise of domain-specialist LLMs in medicine, finance, and law (Singhal et al., 2023; Cheng et al., 2024), while agentic systems improve performance by interleaving reasoning with actions such as retrieval, execution, and API use. In existing multi-adapter workflows, however, selecting which adapter to use is still often handled manually or through fixed heuristics. This raises a natural question: can adapter selection itself become part of the model's reasoning process?

We introduce **Adaptive Minds**, a unified framework that answers this question by treating LoRA adapters as tools that a base model can dynamically select and invoke. At one end, this yields semantic routing, where the model chooses the appropriate specialized adapter from metadata and query semantics, replacing manual dispatch and brittle keyword matching with model-driven selection. At the other, the same formulation extends to multi-step agentic reasoning, where adapters can be invoked alongside other tools to solve more complex tasks requiring decomposition, intermediate reasoning, or multiple forms of expertise. This reframes adapters from static fine-tuning artifacts into reusable capability units that can be called and combined during inference. Like any other tool an agent reaches for, a LoRA adapter is bounded by how well it is built. Across our experiments we observe all three outcomes from the same framework. Well-trained specialists deliver large per-domain gains that the router automatically aggregates, weakly differentiated adapters leave accuracy unchanged, and in some cases adapters can degrade accuracy below the base model. The user-facing implication is simple — train good adapters, and the framework takes care of when, where, and how to use them. More broadly, the framework points toward a different route to stronger and more

general model behavior, not only by scaling a single monolithic model, but by enabling a base model to orchestrate many specialized capabilities within a single interpretable framework.

2 Contributions

1. We present Adaptive Minds, a general framework in which LoRA adapters are exposed as tools that a base model can dynamically select and invoke.
2. We unify two levels of adaptation within this framework: semantic routing over a library of domain specialists, and agentic multi-step reasoning in which multiple adapters can be used alongside external tools to solve more complex tasks.
3. We demonstrate that the same framework supports both direct adapter routing and multi-step agentic reasoning, enabling models to invoke specialized adapters together with external tools across multiple reasoning steps.
4. We measure that, under one shared training recipe and a single backbone, well-targeted LoRA specialists improve strict-scorer accuracy by $+4.6$ to $+84.0$ pp across nine task families, and the AM router aggregates these gains within ± 5 pp of direct expert selection on every benchmark whose queries carry domain-discriminating signal (§7.3).

3 Related Work

3.1 Tool-Augmented Agents

Tool-augmented language models improve performance by interleaving reasoning with actions such as retrieval, execution, or API use. ReAct (Yao et al., 2023b) established the reasoning-action loop, while Toolformer (Schick et al., 2023) and Gorilla (Patil et al., 2024) broadened the view of tools as modular interfaces that language models can invoke. HuggingGPT (Shen et al., 2023) extended this view further by treating entire pre-trained models as callable tools that an LLM controller can plan over, Reflexion (Shinn et al., 2023) introduced verbal self-reflection, and Tree-of-Thoughts (Yao et al., 2023a) extended reasoning to deliberate multi-path exploration, both refining agent behavior over reasoning steps. In these systems, however, tools are typically external capabilities. Our setting differs in that we ask whether a model's own fine-tuned specializations can also be exposed as tools, allowing LoRA adapters to become resources that an agent can explicitly select and use during inference.

3.2 LoRA Routing and Serving

A growing body of work studies how to host, switch, and route among many LoRA adapters efficiently. Systems such as S-LoRA (Sheng et al., 2024) and dLoRA (Wu et al., 2024) focus primarily on efficient multi-adapter serving, while LoRA-Switch and related approaches perform finer-grained routing or blending, often at the token level. Recent methods such as LoRAMoE (Dou et al., 2024), MoLoRA (Shah and Wagle, 2026), and task-level adapter routing approaches further show that inference-time routing can scale to large adapter libraries. This line of work is closely related to ours, but its primary emphasis is on efficient serving, blending, or adapter selection. Our framework operates at a different layer and is composable with these systems — we focus on treating adapters as explicit tools inside a reasoning process, where the model can decide not only which adapter to use, but also when, how often, and in what sequence to invoke multiple adapters while solving a problem. Our own implementation, for example, uses vLLM (Kwon et al., 2023) as the underlying serving stack.

A closely related line of work learns a continuous gating router over LoRA experts at training time, often inside the transformer itself (Dou et al., 2024). Our framework is conceptually orthogonal. Selection is performed by the base model at the tool-call level using natural-language metadata, requiring no additional router training and exposing each adapter invocation as a discrete, named action. This trades the optimization capacity of learned gating for interpretability, modularity, and the ability to add or remove adapters at deployment time without re-training a router.

3.3 LoRA Merging

Another related direction studies merging at the parameter level, drawing on the task-arithmetic paradigm in which fine-tuning deltas are treated as composable vectors (Ilharco et al., 2023; Yadav et al., 2023). LoRAHub (Huang et al., 2024) formulates dynamic composition through adapter selection and weighting, while LoRA Soups (Prabhakar et al., 2025), building on weight-averaging foundations (Wortsman et al., 2022), explores merging LoRAs into a single combined parameter state. Our

framework is complementary but conceptually different. In our setting, the primary form of combination happens across reasoning steps and intermediate outputs rather than within merged parameters, making adapter usage more explicit, interpretable, and easier to analyze.

3.4 Modular Computation and Expert Routing

Our work is also connected to research on modular computation and expert routing. Neural Module Networks (Andreas et al., 2016) show that complex behavior can emerge by routing inputs through specialized sub-networks, and Routing Networks (Rosenbaum et al., 2018) extend this paradigm to multi-task learning by jointly training a router with a set of function blocks. Mixture-of-Experts (Jacobs et al., 1991; Shazeer et al., 2017) extends this intuition to learned expert routing, typically at the token or layer level. We draw on this literature but shift the unit of specialization to the adapter level. Unlike MoE architectures, which rely on a fixed set of learned experts inside a single model, our framework treats adapters as modular inference-time tools that can be added, removed, and combined more flexibly. This also makes the framework compatible with a wide range of base models, including architectures that may themselves already use MoE internally.

Across these lines of work, prior research has shown how to build agents with external tools, how to serve and route many LoRA adapters efficiently, and how to merge adapters at the parameter level. Our contribution sits at the intersection of these directions. We treat LoRA adapters as tools inside a general reasoning framework. This allows a base model to perform both semantic routing for simple requests and multi-step consultation for more complex ones, while preserving a clearer execution trace than purely implicit parameter-level merging.

4 LoRA-as-Tools: Formalism

We formalize each LoRA adapter as a tool that can be selected and invoked by a base model during inference. Given an adapter a_i with parameters $\Delta\theta_i$, a domain description d_i, and an associated prompt template s_i, we define the corresponding tool as

$$\tau_i = (\text{name}_i,\ d_i,\ \{\text{sub_query}:\text{str}\},\ \text{generate}_i), \tag{1}$$

where generate_i activates adapter a_i via `set_adapter`(a_i), constructs an expert-conditioned prompt from s_i, and returns the resulting domain-specific response.

This formulation turns adapters from static fine-tuning artifacts into explicit computational resources that can be planned over during reasoning. The tool registry is generated directly from adapter metadata, so adding a new domain expert requires no changes to the agent logic.

Under this view, routing appears as the special case of tool use in which only a single expert is consulted. Let Q denote the user query, $\mathcal{H}_{<k}$ the interaction history up to step k, and M_θ the base model. At each step, the model reasons over the query and prior context, selects an expert if needed, and obtains an observation:

$$t_k = \text{Think}(Q, \mathcal{H}_{<k};\ M_\theta), \tag{2}$$

$$o_k = \text{Generate}(\text{sq}_k;\ M_{\theta+\Delta\theta_{j_k}}), \tag{3}$$

$$R = \text{Synthesize}(Q, \{o_1, \ldots, o_K\};\ M_\theta). \tag{4}$$

When $K = 1$, this reduces to semantic routing. When $K > 1$, the model can iteratively consult multiple experts and combine their outputs across reasoning steps.

An important consequence of this formulation is auditability. Because each tool invocation is an explicit, named action, the resulting execution trace $\{(t_k, j_k, \text{sq}_k, o_k)\}_{k=1}^{K}$ is observable and attributable. This provides a clearer account of how expertise is used than parameter-level merging, where the contribution of individual adapters is entangled in the merged weights.

5 System Architecture

5.1 Adaptive Minds Framework

Adaptive Minds is a unified framework that supports both expert routing and multi-step agentic reasoning. The central idea is simple. Adapters are exposed as tools, and the base model decides whether a query can be handled by consulting a single expert or whether it requires a richer sequence of reasoning steps involving multiple experts and other tools. Rather than treating routing and agentic reasoning as separate paradigms, we view them as two operating regimes of the same architecture.

5.2 From Routing to Agentic Reasoning

At one end of this framework is semantic routing. For queries that are primarily single-domain, the

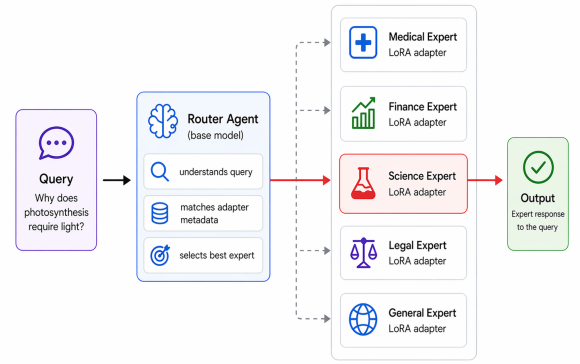

Figure 1: Routing architecture of Adaptive Minds. Given a user query, the base model acts as a router that matches the query to the best domain-specialized LoRA adapter and dispatches it for expert execution. This provides a simple single-step operating mode that replaces manual adapter selection with interpretable model-driven routing.

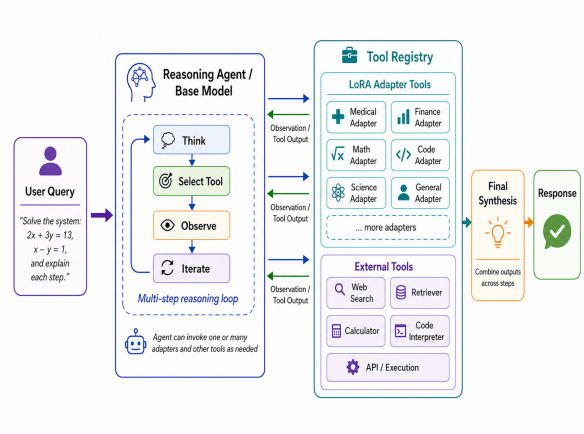

Figure 2: Multi-step agentic reasoning in our framework. LoRA adapters are exposed as callable tools alongside standard external tools, allowing the base model to perform iterative tool selection, observation, and final synthesis over multiple reasoning steps.

model selects the most appropriate adapter from its metadata and dispatches the query directly to that expert. This provides a practical way to aggregate the benefits of many specialized LoRAs within a single deployed framework, without relying on brittle keyword rules or hand-written dispatch logic.

For more complex queries, the framework naturally extends to an agentic setting. Here the model can iteratively invoke multiple adapters alongside external tools such as retrieval, search, execution, or other task-specific resources. This allows the framework to decompose a problem into subqueries, consult different forms of expertise when needed, and synthesize a final answer from intermediate observations. In this regime, adapters function not only as specialists, but also as modular skills or memory-like units that can be called and combined during reasoning.

5.3 Unified Adapter Runtime

We implemented the framework over two complementary runtimes for multi-adapter inference. The first is a direct PEFT-based runtime in which all specialists are attached to a shared base model and selected dynamically at inference time. This setting allowed us to study adapter switching and placement strategies directly, including GPU-resident, CPU-staged, and on-demand loading. The second runtime serves the full adapter library through a single vLLM engine with LoRA support (Kwon et al., 2023). In this setting, expert selection is handled at request time through the serving interface, while the runtime manages adapter paging and execution internally. Together, these implementations show that our approach is not tied to a single serving stack, but can be realized across different multi-adapter runtimes.

A key architectural principle is that control and reasoning remain with the base model, while specialized adapters are used only for expert execution. This separation preserves the general reasoning ability of the base model while allowing domain-specific capabilities to be injected only when needed. As a result, the framework maintains a clear distinction between deciding *what* expertise is required and generating *with* that expertise, making adapter use more modular, more interpretable, and easier to audit than implicit parameter-level merging. In practice, we found the vLLM-based runtime to be both faster and operationally cleaner for large-scale multi-adapter serving. It hosts the full library — up to 60 adapters — within a single engine, and on the 30-adapter library achieves substantially lower router-mode latency than the earlier PEFT-based runtime, while preserving the same high-level routing abstraction.

5.4 Agent Loop and Dynamic Control

The agent alternates between reasoning and tool use. Given Q and registry $\mathcal{T}$, the base model decides whether to act, optionally activates a specialist, executes a sub-query, appends the observation, and repeats before synthesizing a final answer. When a single expert is selected the process col-

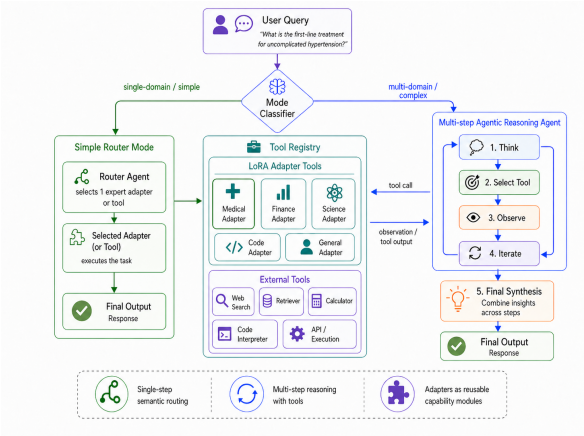

Figure 3: Unified operating modes of the framework. The system first classifies query complexity, then either dispatches the query through single-step semantic routing or invokes a multi-step agentic reasoning loop that combines specialized adapters with external tools.

lapses to routing; otherwise it supports iterative multi-step reasoning.

To choose between regimes, we score query difficulty by the base model's mean predictive entropy over its own first $T{=}16$ output tokens. Let $p_t(v) \triangleq p_t(v \mid Q, h_{<t})$; then

$$H(Q) = \frac{1}{T}\sum_{t=1}^{T}\Big(-\sum_{v\in\mathcal{V}} p_t(v)\log p_t(v)\Big). \quad (5)$$

Queries with $H(Q){<}0.8$ go to Router; $H(Q){>}1.5$ to Agent; intermediate scores fall back to Router with escalation. Thresholds are not tuned per benchmark. A lightweight verification stage gates intermediate observations before final synthesis. The same tool-call interface supports a range of agent patterns. We have implemented and validated the framework with ReAct (Yao et al., 2023b), LangGraph (LangChain Inc., 2024), Tree-of-Thoughts (Yao et al., 2023a), Reflexion (Shinn et al., 2023), and LATS (Zhou et al., 2024) variants, all of which successfully invoke adapters during reasoning, indicating that the contribution is at the framework layer rather than tied to a specific agent design.

6 Domain Experts

We train domain-specialized LoRA adapters on Qwen2.5-7B-Instruct and Qwen3.5-9B (Yang et al., 2025) under one shared recipe per track (reasoning-trace SFT, structured-generation SFT, or GRPO with execution-based reward), so that downstream behavioral differences reflect domain specialization rather than hyperparameter variation. Each adapter is lightweight relative to the base model, making it practical to host many experts simultaneously and switch among them at inference time. Full per-track recipes are reported in App. F.

Brain	$\lvert\mathcal{L}\rvert$	n	Kwd	Rtr	Δ
Llama 3.1 8B	5	29	48.3	100.0	+51.7
Qwen3.5-9B	30	60	31.7	98.3	+66.6
Qwen2.5-7B-Instruct	60	139	71.9	89.9†	+18.0

Table 1: Routing accuracy across 5 to 60 self-trained adapters (App. F). $\lvert\mathcal{L}\rvert$: library size; n: held-out queries; Kwd / Rtr: keyword and LLM router. Each row uses the brain it was evaluated with; within-row Δ is the apples-to-apples comparison. †Equivalence-set scoring; alternative scorings 82.0% (exact-name) and 83.5% (stem-match).

7 Experiments

7.1 Experimental Setup

We evaluate our framework along two complementary axes. First, we study *semantic routing*, asking whether a base model can reliably select the appropriate domain-specialized adapter from metadata and query semantics. Second, we study *specialist-driven gains and their aggregation*, asking whether well-targeted LoRA adapters trained under one shared recipe provide measurable gains on weak-base structured-generation and niche tasks, and whether the AM router recovers those gains automatically within a single deployed framework.

Experiments run on a single NVIDIA L40S (48 GB). The consolidated specialist study uses Qwen2.5-7B-Instruct; routing experiments use Qwen3.5-9B and Llama 3.1 8B as noted. Benchmarks are summarized in §7.3; a complementary MCQ evaluation of four reasoning-trace adapters is in App. B. All Qwen evaluations use `enable_thinking=False`, greedy decoding, and Wilson 95% confidence intervals. The strict scorer is exact-match against the gold answer with no normalization beyond whitespace and case, except where noted in Table 2 (e.g., structural validation for SQL/Cypher/SPARQL, normalized SMILES canonicalization for ChEMBL).

7.2 Semantic Routing Accuracy

Table 1 shows that model-driven routing remains highly reliable as the adapter library grows substantially in both size and ambiguity. In the 5-adapter setting, the router achieves 100.0% accuracy, out-

Benchmark	n	Base	Specialist	AM	Router-acc	Δ Specialist
Strong-base reasoning (no capability gap)						
MATH-500	500	69.2%	67.2% (reasoning v1)	–	–	−2.0 pp
GSM8K-250	250	86.0%	88.0% (reasoning v1)	–	–	+2.0 pp
Weak-base structured generation (five grammars under one recipe)						
Spider-dev (SQL)	300	12.3%	41.7% (sql v1)	39.7%	100%	+29.4 pp
Text2Cypher (graph queries)	300	0.0%	39.3% (cypher v1)	38.0%	92%	+39.3 pp
nl2bash (shell pipelines)	606	8.4%	17.3% (bash v1)	12.0%†	98%	+8.9 pp
LC-QuAD (SPARQL, structural)	100	0.0%	59.0% (sparql v1)	58.0%	95%	+59.0 pp
MermaidSeqBench (visual DSL)	100	4.5%	21.2% (mermaid v1)	24.0%	100%	+16.7 pp
Tool-style and niche adapters						
PII (NER+redaction)	100	2.7%	79.0% (pii v1)	74.0%	99%	+76.3 pp
Qiskit-HumanEval-hard (quantum, GRPO)	151	10.6%	15.2% (quantum v1)	15.0%	93%	+4.6 pp
LEDGAR (legal, 3-stage curriculum)	100	0.0%	84.0% (legal v1)	84.0%	100%	+84.0 pp
ChEMBL canonical SMILES (chem)	300	6.3%	36.7% (chem v1)	35.0%	100%	+30.4 pp

Table 2: Consolidated specialist results on Qwen2.5-7B-Instruct (one shared training recipe, all adapters served by a single vLLM engine). Strict-scorer accuracy per benchmark; AM denotes the deployed router (heuristic + LLM fallback); Router-acc is the fraction routed to the domain-matched specialist; "–" = AM not run. †AM on nl2bash measured on n=100 subset (specialist + base direct on full n=606).

performing keyword matching by 51.7 percentage points. In the 30-adapter setting, it still achieves 98.3% accuracy on a 60-query manually curated evaluation set, despite a much larger choice space and genuine overlap between related adapter families. For example, prompts in mathematics may reasonably map to `math`, `math_reasoning`, or `math_grpo`, while prompts in medicine may overlap across `medical`, `biomedical`, and `medical_reasoning`. In this larger setting, some queries admit more than one valid adapter because several specializations cover closely related capabilities; predictions are therefore counted as correct when the selected adapter falls within the valid set for that query. By contrast, the keyword baseline drops from 48.3% to 31.7%, increasing the router's absolute advantage from +51.7 to +66.6 percentage points.

This is a more realistic test than an isolated-specialist setting: with overlap the question is whether the router can choose sensibly among nearby capabilities, not just match an unambiguous specialization. Accuracy holds as the pool grows 6×, supporting model-driven selection as an alternative to manual dispatch or keyword rules. All adapters in the 5/30/60-adapter libraries were trained by us under disclosed recipes (App. F).

Scaling to 60 real adapters. To probe routing on a larger pool with genuine semantic overlap, we extended the library to 60 adapters on Qwen2.5-7B-Instruct (139 held-out queries). The library introduces overlap absent from the 30-adapter setting, including multiple training stages of the same generator (SMILES GRPO V1/V3/V4), family aliases (`math_reasoning` vs. `math`), and same-domain different-lineage adapters (case-law vs. legal-SFT). Because several adapters can legitimately answer the same query, we credit a pick if it falls within a query-specific equivalence set — the same scoring policy used for the smaller pools. Under this policy the LLM router reaches 89.9%, holding above 80% as the pool doubles in size. The 7B brain routes reliably across ∼60 overlapping adapters once equivalent picks are credited.

Naturalistic queries. As a spot check on a benchmark whose queries do not carry surface-form domain keywords — PII redaction over benign-looking personal letters (e.g., "Dear Haylee, I see you're reaching your 90th birthday March 15, 1941…", n=300) — the deployed router still routes correctly on 93% of queries.

Deployment cost. On the 30-adapter library, vLLM serves the full set from a single engine at mean router-mode latency 3.48 s on a single L40S; adapter placement (GPU-resident / CPU-staged / on-demand) changes end-to-end latency by <5%, so generation time dominates (App. A).

7.3 Specialists Drive the Gains; AM Aggregates Them

The nine LoRA specialists in Table 2 share one training recipe (r=32, α=64, all-linear, 400–600 SFT steps; the chem/legal/quantum trio adds an SFT-broad → SFT-narrow → GRPO curricu-

Mode	All	Chem	Fin	AI	Med	Gen
Vanilla	56.0	55	65	25	80	55
Router	62.0	60	60	40	75	75
Agent	58.0	70	40	45	75	60

Table 3: Three-way comparison on 100 domain-matched MMLU questions (Llama 3.1 8B).

lum), and are served simultaneously from a single vLLM engine. Each benchmark received an automatic strict scorer plus a fixed-seed manual audit (n=20−30).

Headline finding. Three patterns in Table 2. (i) *Specialists help where the base has a real capability gap.* On strong-base reasoning the recipe moves accuracy $< \pm 2$ pp; on the nine weak-base tasks it gains $+4.6$ to $+84.0$ pp. (ii) *AM aggregates these gains.* AM-route tracks the directly-pinned specialist within ± 5 pp on all 9 measured benchmarks — the router recovers each per-domain gain without a human picking the right adapter. (iii) *Routing is reliable at scale.* With §7.2 (98.3% on 30 adapters, $+66.6$ pp over keyword), the pipeline of trained adapters + reliable router + per-specialist accuracy is coherent end-to-end.

Directional transfer. Off-domain transfer is essentially zero on 7 of 9 specialist-vs-benchmark pairs; two pairs show partial transfer with structural overlap (Bash on regex, SQL on Mermaid sequence diagrams), so the domain-matched-expertise claim holds modulo structurally adjacent grammars.

7.4 Routing vs. Multi-Step Reasoning

To better understand the regimes in which routing and iterative reasoning are useful, we compare vanilla inference, semantic routing, and multi-step agentic reasoning on 100 domain-matched MMLU questions using Llama 3.1 8B. Table 3 shows that routing already provides a meaningful gain over vanilla inference on matched single-domain queries. Multi-step reasoning is not uniformly better; it helps most when the query benefits from decomposition, but can underperform routing on tasks that are better handled by direct expert selection. This supports the core design of our framework as a unified system spanning both regimes rather than privileging one mode universally.

8 Discussion

Routing is already a strong capability. A central finding of this work is that semantic routing is not merely a pre-processing heuristic, but a substantive capability of the base model. When adapters are exposed with clear metadata, the model can reliably select the appropriate expert and thereby aggregate the benefits of many specialized LoRAs within a single deployed framework. This alone makes the framework useful, even before considering more complex agentic behavior.

Routing and agentic reasoning serve different regimes. Our three-way comparison suggests that routing and multi-step reasoning are best understood as complementary operating modes. Routing is effective for simpler or clearly matched queries, while iterative reasoning becomes more relevant when the task requires decomposition, external tools, or multiple experts. This motivates the design of a unified framework that can move between these regimes rather than commit to a single inference strategy.

Interpretability is a practical advantage of explicit tool use. Because expert selection and invocation remain explicit, the framework yields an inspectable, attributable trace, a meaningful advantage over parameter-level merging where the contribution of individual experts is harder to recover.

Train good adapters; the framework handles the rest. LoRA adapters, once exposed as tools, behave like any other tool an agent has, and their utility is bounded by how well they are built. The nine specialists in Table 2 deliver $+4.6$ to $+84.0$ pp strict-scorer gain, and AM-route recovers each of those gains within ± 5 pp of the direct specialist. The user-facing proposition is therefore not that more adapters always help, but that adapter quality is what carries the framework. The ceiling is set by how well each tool is trained, the floor is the base model, and the framework's job is to make the difference available on the right query at the right time.

Scope: out-of-distribution MCQ. A balanced view requires noting where the framework does not help. On GPQA Diamond, a graduate-level science MCQ benchmark on which none of our adapters were trained, single-step routing underperforms the base model on Qwen2.5-7B-Instruct (-20.7 pp) due to format mismatch between specialist outputs and the strict `\boxed{X}` scorer. A multi-step ReAct loop, which keeps synthesis in the base model's voice, recovers most of the gap (within 1.3 pp of base; App. E). This reinforces the conditionality

story. The framework aggregates specialist gains where they exist, and on tasks with no matching specialist the routing layer should be bypassed in favor of the multi-step controller.

9 Conclusion

We presented Adaptive Minds, a unified framework in which LoRA adapters are treated as tools that a base model can dynamically select and use. Our results show that semantic routing is highly reliable, substantially outperforming keyword-based matching while remaining lightweight enough for practical deployment. At the same time, the downstream evaluation of individual experts reveals that adapter benefits are selective rather than universal, supporting a view of adapters as modular sources of expertise rather than generic accuracy boosts. Taken together, these findings motivate a broader agentic setting in which adapters can be combined with other tools and invoked iteratively during reasoning. More broadly, our results suggest that stronger and more general model behavior may emerge not only from scaling a single model, but from enabling a base model to orchestrate many specialized competencies within one coherent and interpretable framework.

Limitations

With routing reliable on libraries of up to 60 adapters, several directions remain open.

Scale. Our largest tested library is 60 adapters; whether routing holds at hundreds or thousands with heavier overlap is open. The 89.9% figure at $|\mathcal{L}| = 60$ also depends on a hand-curated equivalence set; automated larger-scale evaluation is future work.

Training-method coverage. Every adapter here is SFT or GRPO with execution-based reward; CPT, DPO, and preference- or distillation-based recipes are unexplored. A natural next step is *training-as-a-tool*, where adapter creation itself becomes an agent action (App. F sketches the pipeline); we leave its quality, routability, and safety properties to future work.

Agent depth. The multi-step agent underperforms direct routing on strict-format benchmarks (GPQA Diamond, -20.7 pp; App. E); deeper agentic strategies, longer horizons, and head-to-head comparison against existing tool-augmented frameworks are natural next steps.

Evaluation breadth. Our specialist study uses n=20–30 manual audits per benchmark, single seed, and a single primary base model; multi-seed evaluation with full execution match would tighten §7.3.

No human evaluation. All scoring is automatic. Domains where automatic verification is weak (legal IRAC reasoning, mechanism chemistry) are bounded by the validator's strictness.

Ethical considerations

Data. All training data is from publicly available, license-permissive sources. The PII adapter is trained on synthetic PII generated by a templated tool; no real personal information is included in the training set.

Dual-use. The same routing mechanism that selects between SQL and Cypher experts could route between detection and generation for sensitive categories. We release detection-oriented adapters only and recommend deployment behind appropriate access controls.

Compute. Training is modest. Each specialist LoRA is ≤ 1.5 GPU-hours on a single L40S, so the full library is $\sim$30 GPU-hours of training compute; end-to-end evaluation is on the order of a few additional GPU-hours.

Reproducibility. Code, adapters, training configs, and evaluation scripts are released. Numbers reported in this paper are computed by the release scripts; we do not maintain a separate scoring pipeline.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *CVPR*.

Daixuan Cheng, Shaohan Huang, and Furu Wei. 2024. Adapting large language models to domains via reading comprehension. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *ICLR*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. LoRAHub: Efficient cross-task generalization via dynamic LoRA composition. In *Conference on Language Modeling (COLM)*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. 2022. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv:2205.00445*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*.

LangChain Inc. 2024. LangGraph: Low-level orchestration framework for stateful, multi-actor LLM applications. https://github.com/langchain-ai/langgraph. Software framework.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Akshara Prabhakar, Yuanzhi Li, Karthik Narasimhan, Sham Kakade, Eran Malach, and Samy Jelassi. 2025. LoRA soups: Merging LoRAs for practical skill composition tasks. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track (COLING)*.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. 2018. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *International Conference on Learning Representations (ICLR)*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shrey Shah and Justin Wagle. 2026. MoLoRA: Composable specialization via per-token adapter routing. *arXiv preprint arXiv:2603.15965*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. 2024. S-LoRA: Serving thousands of concurrent LoRA adapters. In *Proceedings of the 5th MLSys Conference*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, and 1 others. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. 2024. dLoRA: Dynamically orchestrating requests and adapters for LoRA LLM serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

An Yang and 1 others. 2025. Qwen3 technical report. *arXiv:2505.09388*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *ICLR*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

A Latency and Serving

Router mode is $3.1\times$ faster than baseline (mean 3.49 s vs. 10.81 s), and adapter switching takes less than 1 ms. A deployment ablation comparing GPU-resident, CPU-staged, and on-demand placement shows less than 5% end-to-end latency difference, indicating that generation dominates runtime.

Mode	Init (s)	Router (s)	Agent (s)	GPU (GB)
GPU-resident	7.68	23.02	58.72	17.86
CPU-staged	7.17	23.85	62.63	17.06
On-demand	3.74	24.09	64.36	17.06

Table 4: Adapter-placement serving ablation (10 prompts, Qwen3.5-9B).

B Domain Expert Evaluation: Full Matrix

C Reasoning-Trace Adapter Training

All adapters are trained with LoRA $r = 32$, $\alpha = 64$, dropout 0.05, and all-linear target modules using AdamW-fused, cosine learning rate, 50 warmup steps, 500 total steps, batch size 2×8 gradient accumulation, sequence length 2,048, and bf16. Training outcomes are: Medical (loss 1.187, 183 min), Finance (0.968, 354 min), Science (0.410, 171 min), and General (0.424, 345 min).

Config	MedQA (n=500)	FinBench (n=150)	ARC (n=500)
Base	76.6 [72.7,80.1]	66.0 [58.1,73.1]	95.0 [92.7,96.6]
+ Medical	75.0 [71.0,78.6]	66.7 [58.8,73.7]	95.2 [93.0,96.8]
+ Finance	75.4 [71.4,79.0]	**70.0** [62.2,76.8]	94.4 [92.0,96.1]
+ Science	72.4 [68.3,76.1]	64.7 [56.7,71.9]	93.4 [90.9,95.3]
+ General	75.4 [71.4,79.0]	65.3 [57.4,72.5]	95.0 [92.7,96.6]

Table 5: Reasoning-trace adapters on Qwen3.5-9B with Wilson 95% confidence intervals. This complementary smaller-scale MCQ study sits alongside the main consolidated specialist results in §7.3; only Finance $\times$ FinanceBench produces a clean in-domain gain ($+4.0$ pp), consistent with the conditionality pattern.

Implementation notes. TRL auto-detects Qwen3.5 as a VLM, requiring `packing=False` and `processing_class=tokenizer`. Setting `per_device_eval_batch_size=1` avoids evaluation OOM. We also found that `enable_thinking=False` together with a `/no_think` suffix is required for consistent logit extraction.

Adapter	Dataset	Used
Medical	medical-o1-reasoning-SFT	10,000
Finance	Fino1_Reasoning_Path_FinQA	5,499
Science	CAMEL STEM mix	10,000
General	OpenThoughts-114k	10,000

Table 6: Reasoning-trace adapter datasets used for training on Qwen3.5-9B. Dataset names are abbreviated for readability.

D Acceptance Policy for the 60-Adapter Routing Evaluation

The 60-adapter row of Table 1 reports the equivalence-set acceptance score, matching the policy used for the smaller pools where queries that admit more than one valid adapter are credited if the pick falls inside a curated equivalence set. We report all three policies for transparency:

- **Exact-name (82.0%):** exact name match between predicted adapter id and the gold target.
- **Stem-match (83.5%):** a pick is accepted if its name shares a same-lineage stem with the gold target (e.g., `sql_v1` and `sql_v2` share the stem `sql`). No hand-declared equivalences beyond stem matching.
- **Equivalence-set (89.9%, headline):** a pick is accepted if it falls into a hand-curated equivalence set for the query, declared before scoring. Three categories of equivalences are

admitted: (i) multiple training stages of the same generator (e.g., the SMILES optimizer's `GrpoQedOptimV1/V3/V4` checkpoints are treated as equivalent to a query that asks for goal-directed SMILES generation); (ii) family aliases where the LLM emits the family token rather than a specific stage (e.g., emitting `math` for a query whose target was `math_reasoning`); (iii) same-domain different-lineage adapters when both legitimately match the query (e.g., `case_law` vs. `legal_sft` for a generic legal question).

What equivalence-set scoring lets in. Across the 139 held-out queries, equivalence-set scoring newly accepts 12 LLM-router picks beyond exact-name scoring, comprising 9 family-alias / lineage picks (6 on the SMILES GRPO stages, 1 math alias, 2 stages of the SMILES generator) and 3 same-domain different-lineage picks (2 case-law vs. legal, 1 flower-tuned medical vs. medical SFT). The keyword baseline accepts 0 additional rows under equivalence-set scoring — its misses are domain-wide errors (e.g., picking an astrophysics adapter when nothing else matches) that no acceptance policy could rescue. The equivalence-set gap between router and keyword therefore reflects true routing wins, not a scoring artifact.

Reproducibility. The full set of per-query records, equivalence-set definitions, and the picks audit fingerprint are released alongside the code (`routing_60real_relaxed.json`), so any reader can recompute exact-name, stem-match, and equivalence-set numbers from the released data without re-running inference.

E Out-of-Distribution MCQ: GPQA Diamond

To stress-test the framework outside the regime of its specialists, we evaluated on GPQA Diamond, a 198-question graduate-level science MCQ benchmark covering biology, chemistry, and physics. None of our adapters were trained on GPQA-style data, and the benchmark uses strict `\boxed{[A-D]}` answer extraction.

Three observations follow from Table 7. First, on Qwen2.5-7B AM-route regresses by -20.7 pp due to format mismatch: the router sends 168 of 198 queries to a reasoning-trace specialist whose training distribution emphasizes verbose multi-paragraph rationales rather than a clean `\boxed{X}` terminal answer. The router is making sensible adapter choices, but the chosen specialist's output style does not survive the strict scorer. Second, when the agent loop keeps the base model in the controller role and treats specialist outputs as intermediate observations, the format degradation disappears and accuracy returns within 1.3 pp of base. Third, on Qwen3.5-9B AM-route exactly matches base accuracy, and only a domain-matched biology adapter exceeds base (by $+1.0$ pp), consistent with the conditionality story in §7.3.

Base	Mode	n	Acc.
Qwen2.5-7B	base	198	33.8%
Qwen2.5-7B	AM-route	198	13.1% (-20.7)
Qwen2.5-7B	AM-react	80	32.5% (-1.3)
Qwen3.5-9B	base	198	28.8%
Qwen3.5-9B	AM-route	198	28.8% (± 0.0)
Qwen3.5-9B	biology-adapter	198	29.8% ($+1.0$)

Table 7: GPQA Diamond accuracy. AM-route dispatches each query to a single specialist via the router; AM-react runs the multi-step ReAct loop with the base model as controller.

F Adapter Inventory and Training Recipes

Three distinct training recipes were used across the project, each appropriate to its scale and intended role.

Reasoning-trace track (4 adapters, App. C). The four reasoning-trace specialists analyzed in App. B (Medical, Finance, Science, General) use r=32, α=64, all-linear targets, 500 SFT steps, batch 2×8, sequence length 2,048, learning rate 2e$-$4 cosine.

Specialist track (§7.3). The Qwen2.5-7B-Instruct structured-generation specialists (SQL, Cypher, Bash, SPARQL, Mermaid, regex) share the same r=32/α=64 LoRA hyper-config but with method-specific step budgets: 400–600 SFT steps for declarative grammars; 200 GRPO steps with execution-based reward for regex.

Domain-knowledge track (60 routed adapters total). A lower-cost recipe was used to construct the broader domain pool that backs the 30- and 60-adapter routing experiments: r=16, α=32, target modules restricted to the standard $\{q, k, v, o, \text{gate}, \text{up}, \text{down}\}$ projections, 100 SFT steps, batch 2×4, sequence length 1,536, learning rate 1e$-$4 cosine, AdamW-fused, bf16, gradient

checkpointing on. This recipe trades per-adapter optimization budget for breadth of domain coverage, which is the property the routing experiment requires. The 30-adapter routing pool (§7.2) is a curated subset of this library on Qwen3.5-9B; the 60-adapter pool extends it with additional knowledge domains and with multiple training stages of select generators (e.g., SMILES GRPO checkpoints V1/V3/V4), specifically to introduce the kinds of semantic overlap discussed in §7.2. The 5-adapter pool used in the earliest routing measurement on Llama 3.1 8B is a separate, smaller-scale library trained under the specialist track recipe above.

Training-as-a-tool pipeline. A separate pipeline builds adapters on demand. A dataset source (HuggingFace, web search via Tavily, or direct URL ingest) is converted to an SFT JSONL by a build worker; an orchestrator manages the vLLM and training-process lifecycle; and a freshly-trained adapter is symlinked into the LoRA pool, ready to be discovered by the router. As a worked example, an `electrical_drawings` adapter was built end-to-end from 66 web URLs $\rightarrow$ 100 chunks $\rightarrow$ 302 Q&A pairs synthesized by a local vLLM judge $\rightarrow$ 200 SFT steps, in $\sim$21 minutes of wall time on a single L40S. Adapter training is fast enough on a single GPU that "train a new tool on the fly" is a realistic deployment mode for the LoRA-as-tools framework.

G Architecture Details

Entropy gating. The entropy-based mode-selection rule is defined in Eq. 5 (§5.4). The same thresholds ($H{<}0.8 \rightarrow$ Router, $H{>}1.5 \rightarrow$ Agent) and the same window $T{=}16$ are used across every experiment in this paper; none of these values were tuned per benchmark.

Critic format.

```
Sub-query: {sub_query}
Observation: {observation}
FAITHFULNESS: yes/no
CONFIDENCE: 0.X
REASON: ...
```

Faithfulness=no or confidence < 0.4 injects a reroute hint.