Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

A Modular 'Patch-and-Route' Framework for Continual Learning in LLMs

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)

eingereicht von: Leon Wagner
geboren am: 13.09.1996
geboren in: Mainz

Gutachter/innen: Prof. Dr. Alan Akbik
Prof. Dr. Verena Hafner

Betreuer: Ansar Aynetdinov

eingereicht am: 18.06.2026 verteidigt am: .....

Abstract. Globally, there is still no complete solution that allows enterprises, research centers, and similar organizations to continuously train their Large Language Models (LLMs) in a long-term, scalable, and automated way, ensuring they internalize newly acquired information, changing conditions, or updated rules, and have them readily available whenever users ask. Unlike LLMs, humans can iteratively learn new things without losing what is already anchored in their long-term memory. With current approaches to training LLMs on new content, this is unfortunately not yet possible, as LLMs suffer from the phenomenon of catastrophic forgetting, which is why the entire industry heavily favors Retrieval-Augmented Generation (RAG) as a workaround.

This thesis introduces the Patch-and-Route (PnR) framework, a single unified architecture that offers two alternative routing variants, both combining parametric and non-parametric (embedding-based) approaches. Within this framework, training occurs isolated in independent LoRA experts. Queries are routed dynamically to these experts, leaving the base model entirely untouched and thus fully preserving its original capabilities. The PnR framework is evaluated against standard training methods such as monolithic LoRA and LoRA+RAG, as well as two highly competitive continual learning approaches, RECIPE and X-LoRA.

Our results demonstrate that with correctly implemented routing, continual learning is fundamentally possible with near-zero catastrophic forgetting. Furthermore, we show that this can be implemented with almost zero inference overhead, and that the training cost per update does not scale with the size of the knowledge base.

However, we frame this result honestly by acknowledging that achieving this shifts the problem of catastrophic forgetting away from shared weights and onto the routing mechanism itself. A stress test of our experimental setup using out-of-domain queries revealed that the routing gate can leak queries into experts, showing that our implementation requires further refinement to offer a fully generalizable solution for enterprise deployment.

Contents

List of Acronyms

AI artificial intelligence.

CL continual learning.

CMR Continual Model Refinement.

EWC Elastic Weight Consolidation.

KE Knowledge Editing.

LLM large language model.

MLP multi-layer perceptron.

MoE Mixture-of-Experts.

PEFT parameter-efficient fine-tuning.

PnR Patch-and-Route.

RAG retrieval-augmented generation.

1 Introduction

It is striking how something that seems so simple for us humans is so incredibly difficult for artificial intelligence (AI). Humans can learn continuously with great ease: First they learn to speak, then they acquire general knowledge at school, and afterward they pursue higher studies, all without losing previously acquired skills or knowledge. If this sequential process were applied to an large language model (LLM) using current training methods, the results would likely be detrimental. An AI that has initially learned to generate natural language would tend to suffer severe linguistic degradation upon learning general knowledge, and by the end of its specialized studies it would likely have forgotten much of the general knowledge and lost much of its ability to articulate well. This is primarily because AI models are prone to catastrophic forgetting (Shi et al., 2024; Yang et al., 2024; Wang et al., 2024b).

The main reasons identified so far as to why today's AI models cannot learn continuously the way humans do are, on the one hand, that both new and old knowledge are trained on the same weights, that gradient descent is performed globally across all parameters, and that plasticity in AI models prevails globally and uniformly (French, 1999), whereas humans keep only those synapses plastic that are locally involved in a specific task (Hebbian learning rule) (Hassabis et al., 2017; Lazari et al., 2022). On the other hand, we humans possess a dual memory system, in which we use the hippocampus as a fast, temporary buffer for new knowledge and the neocortex for long-term storage (McClelland et al., 1995). By comparison, standard deployed AI models have only a single "long-term memory" (namely, their weights). New knowledge must be forced directly into this static system.

Moreover, standard deployed AI models have no built-in consolidation process analogous to sleep in humans. The human brain strengthens and integrates new knowledge during rest phases, in which memories are replayed in the background and integrated in a structured manner into the existing world model of the neocortex (Schapiro et al., 2018). This ongoing consolidation process is largely absent in such models. Furthermore, today's AI models are often polysemantic: individual neurons respond to entirely unrelated things (for example, "DNA sequences," "legal language," and "Python code" all at once), because the model encodes more concepts than it has neurons, and these concepts therefore share overlapping weights (Elhage et al., 2022). Together with the generally distributed storage of knowledge, this makes it difficult and error-prone to selectively edit individual learned facts out of the model without damaging neighboring knowledge (Meng et al., 2023). Finally, the biological brain constantly forms new synapses (synaptogenesis) and breaks down unused ones over the long term (pruning) (Piochon et al., 2016). By contrast, the most widely used AI models rely on a fixed, predetermined architecture.

Because true fine-tuning for continuous updates has proven simply too expensive, too slow, and too prone to catastrophic forgetting (Interrante-Grant et al., 2025), Retrieval-Augmented Generation (retrieval-augmented generation (RAG)) has emerged as the genuinely viable alternative. Following its introduction in 2020 (Lewis et al., 2020) and the subsequent mainstream breakthrough of LLMs in 2023 (Zhao et al., 2026), and aided by the release of popular frameworks, RAG has become the industry standard (Huang

and Huang, 2026). RAG turns to its advantage the fact that it does not really learn any information at all; instead, it exploits the in-context learning capabilities of LLMs so that, unlike other parametric approaches, it does not actually memorize things but simply looks them up as soon as they are needed by the user. Compared with the human perspective, RAG resembles a kind of limited hippocampus for the machine, allowing it to retain information in the first place, but without any connection to the neocortex.

The problems with this approach become increasingly apparent over the long term, as the vector databases used for RAG grow larger and larger, until finding the correct knowledge snippets becomes ever more difficult and imprecise, essentially turning into a "needle in the haystack" problem (Gao et al., 2026). In addition, one must reckon with rising costs and latency when dealing with large contexts, and such systems often suffer from the "lost in the middle" effect (Liu et al., 2024; Barnett et al., 2024; Soman and Roychowdhury, 2024). Furthermore, RAG does not enable any true cross-domain understanding; rather, it gives the impression of superficial, last-minute memorization rather than genuine integration (Song et al., 2025). It is at this point that RAG reaches its limits, and it becomes clear that parametric knowledge storage is ultimately indispensable in the long run (Su et al., 2025). Yet it is precisely here that no blueprint exists demonstrating how this could be done correctly, without catastrophic forgetting.

1.1 Thesis Objective

Many of the individual components that would be important for genuine continuous learning (inspired by humans) have already been studied intensively (more on this in Section 2). Importantly, this thesis treats the neuroscientific parallels above as a source of inspiration rather than a literal blueprint to be replicated. Concretely, it investigates whether currently available approaches can be combined into a practicable continual-learning architecture that integrates new parametric knowledge without catastrophic forgetting, and evaluates how well such an architecture preserves prior knowledge under sequential updates.

1.2 Thesis Structure

The following two sections provide an overview of the literature on continual learning (CL) and identify the precise research gap and objectives. Section 4 describes the implemented framework, whose complete source code is publicly available.[1] Section 5 details the experimental setup, including the datasets, the by-construction stability probe, metrics, and baselines. Section 6 then presents the results, which are interpreted in Section 7 alongside an analysis of the threats to validity. Finally, Section 8 concludes the thesis, and Section 9 outlines avenues for future research.

[1] `https://github.com/Leon-AW/PnR-framework`

2 Related Work

This thesis stands at the intersection of four research areas: CL, LLM architecture, parameter-efficient fine-tuning (PEFT) application, and the newer field of Model Editing and Refinement. The central problem draws on each area. The key question that threads them together is how to integrate domain-specific, evolving knowledge into an LLM without causing catastrophic forgetting or clashes between facts.

2.1 CL: A Foundational Overview

CL, often referred to as lifelong or incremental learning, requires an AI system to learn from a continuous flow of data in order, while retaining previously learned information. One central tension underlies the entire problem: stability, retaining old knowledge, versus plasticity, absorbing the new. Wang et al. (2024b) offers a detailed overview and categorizes the defenses against catastrophic forgetting into five groups.

Regularization-based Methods These add a penalty term to the loss function that restricts updates to parameters deemed essential for earlier tasks. Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) estimates this importance using the Fisher Information Matrix (FIM). Memory Aware Synapses (MAS) derive it from the total gradient gathered during training (Aljundi et al., 2018). In either case, the method identifies which parameters must remain stable for existing knowledge to persist.

Replay-based Methods These strategies aim to keep past experiences alive by either preserving a small amount of old data for rehearsal (experience replay) or training a generative model to create pseudo-samples from the old distribution (generative replay) (Delange et al., 2021). They tend to perform well in practice. Their principal limitations are storage constraints and data privacy issues.

Architecture-based Methods In this approach, the architecture itself evolves, assigning each task its own parameters. Progressive Neural Networks (PNN) expand the network directly (Rusu et al., 2016); other methods utilize masking to carve out parameters. Many PEFT techniques used sequentially also belong to this category.

Optimization-based Methods These alter gradient updates during training, keeping interference with prior tasks to a minimum. Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017) ensures that the loss on reviewed samples never increases by projecting each gradient. Later techniques advance this by projecting updates into subspaces orthogonal to the gradient spaces of earlier tasks, which allows for interference-free updates.

2.2 CL in the Context of LLMs

For LLMs, the stakes are higher. Their pre-training is static, so they can fall behind or overlook specialized knowledge, and retraining is expensive to perform frequently. A survey by Shi et al. (2024) outlines the LLM lifecycle, dividing continuity along two axes:

Vertical Continuity This tracks an LLM's evolution from general to specialized (general pre-training → domain-adaptive pre-training → task-specific fine-tuning), where the changes in data and objectives occur hierarchically.

Horizontal Continuity This captures adaptation over time, or across different domains at a similar level of specificity.

To handle the challenges of horizontal continuity, such as long task sequences and abrupt distributional shifts, researchers employ a variety of CL techniques. While some models rely on experience replay or parameter regularization, the massive scale of LLMs has made architecture expansion particularly popular. This sets the stage for approaches that step away from typical incremental fine-tuning by keeping the base model's parameters completely intact, opting instead to channel evolving domain knowledge into discrete modules.

2.3 Architectural Approaches: PEFT and Modular Systems

Full fine-tuning is costly and often leads to catastrophic forgetting. Architectural methods, especially PEFT, therefore become the preferred choice for budget-limited LLM adaptation. These methods freeze the base model and only train a small set of new parameters, effectively isolating fresh knowledge.

LoRA (Hu et al., 2021) incorporates trainable low-rank matrices into the Transformer layers and is effective at preserving general knowledge. However, if applied naively in a continual context, adapters trained on varying datasets can begin to interfere with each other. Recent studies are addressing this collision issue. InfLoRA (Liang and Li, 2024) ensures its update subspace remains orthogonal to the gradient spaces of past tasks, promising an interference-free adaptation that absorbs new data without disrupting existing knowledge.

A more ambitious architectural approach looks towards modular systems such as Mixture-of-Experts (MoE) models. MoE frameworks attach a new expert module for every new domain, effectively compartmentalizing knowledge. Lifelong-MoE (Chen et al., 2023, 2025a) and DEMix (Gururangan et al., 2021) illustrate how to train and integrate new domain experts while keeping previous ones frozen. This architecture combats catastrophic forgetting and organizes new knowledge by theme.

2.4 Continual Model Refinement and Knowledge Editing (KE)

The issue of outdated information has pushed Model Editing into a continual framework, often referred to as Continual Model Refinement (CMR) or Lifelong Model Editing. Most

work in this space resolves conflicts by identifying the relevant parameters in the model and overwriting them. The editing literature reveals how contradictions emerge within a Transformer and which methods can effectively isolate them.

Identifying and Localizing Knowledge Conflicts Any editing process must consider how an LLM holds knowledge and what occurs when that knowledge gets contradicted. Feeding the model conflicting claims creates noticeable tension within its internal structures, and this disruption is most evident in the residual stream (Zhao et al., 2024). Resolving these contradictions is what maintains factual accuracy (Wang et al., 2023). Early editing techniques traced factual associations to broad Transformer layers, but current methods operate at a much finer granularity. They now reach fine-grained neuron-level localization to pinpoint the precise parameters associated with a specific fact (Pan et al., 2025). This level of precision allows for targeted adjustments while keeping nearby network structures intact.

Performing and Scaling Targeted Updates Identifying the outdated fact, however, is only the first step; the model's parameters must still be updated accordingly. The main approach tweaks the weights of the identified neurons directly. Some techniques scale this to mass-edit a Transformer's memory, allowing the rewriting of thousands of facts simultaneously (Meng et al., 2022). The central limitation is that overwriting parameters repeatedly degrades the model. Frequent edits lead to catastrophic forgetting of earlier updates or diminish the system's foundational general knowledge.

Non-Parametric and RAG-Based KE In addition to precise parameter changes, RAG-based KE has developed into a reliable non-parametric option for CL. Instead of risking catastrophic forgetting from weight rewrites, these methods view editing as managing a dynamic external memory and the strategies for retrieving from it. Purely non-parametric techniques depend on online index updates, graph-structured memories (HippoRAG 2 (Gutierrez et al., 2025)), or agent-driven assimilation and accommodation (ActiveRAG (Xu et al., 2026)) to continuously reshape their stored knowledge. When sources do not agree, a system such as Astute RAG (Wang et al., 2025) intervenes during inference to identify and suppress outdated or contradictory evidence before generating output. Hybrid architectures connect both approaches, combining retrieval with parametric modularity. **RECIPE** (Chen et al., 2025b) is particularly noteworthy: it is a recent retrieval-augmented method for lifelong KE that condenses knowledge statements into brief continuous prompts through a trained encoder (RoBERTa + multi-layer perceptron (MLP)). A Knowledge Sentinel determines in real-time whether the retrieval repository contains anything relevant to a query, swapping a set similarity threshold for a learned decision boundary. With the LLM parameters remaining fixed and the edits existing non-parametrically, RECIPE avoids the cumulative parameter degradation that affects surgical editing methods (like the approaches from ROME and MEMIT) and other parameter-based methods (T-Patcher). At the time of its publication it achieves leading lifelong editing performance across ZSRE and CounterFact while keeping the LLM's

general capabilities and inference speed unharmed (Chen et al., 2025b). However, the system's effectiveness hinges on the reliability of the Knowledge Sentinel: if the sentinel misjudges relevance (admitting irrelevant retrievals or suppressing valid ones), the corrupted continuous prompt propagates directly into the LLM output, a failure mode structurally analogous to the gate leakage inherent to routing- and gating-based architectures in general. As the strongest publicly reproducible non-parametric baseline for lifelong KE available at the time of this work, RECIPE is adopted as an experimental baseline in Section 5. Other systems take a slower route, gradually distilling dynamic memory updates back into the base model over time (Fan et al., 2025).

2.5 Recent Approaches in Modular Architectures and Nested Learning

While editing-based methods attempt to localize and overwrite specific parametric knowledge within existing weights, a parallel research line pursues a fundamentally different strategy: keeping all parameters frozen and routing queries dynamically to specialized, isolated modules. The three approaches discussed in this section are described in greater depth than the earlier overviews, as they serve as the primary points of comparison in this thesis's evaluation.

Recent advancements have introduced novel methods for routing and composing adapters to address the limitations of static PEFT in CL. This shift towards modularity is supported by the recently proposed "Nested Learning" (NL) paradigm Behrouz et al. (2025). Behrouz et al. argue that traditional deep learning architectures struggle with continual adaptation because they attempt to compress all "context flows" into a single optimization loop. Instead, they postulate that effective learning requires a hierarchy of nested optimization problems, separating high-level context selection from low-level feature learning. This theoretical insight aligns with emerging practical frameworks that route and compose specialized modules.

Araujo et al. introduced **L2R** ("Learning to Route"), a method designed to overcome interference and suboptimal routing in CL settings Araujo et al. (2024). L2R isolates the training of new PEFT modules to ensure task specialization and subsequently trains a router network that leverages a small memory of past examples to effectively compose these modules. The framework proposes multiple routing strategies, among which the soft weighted average configuration (**L2R-wavg**) achieves the highest accuracy by blending the outputs of all available adapters. However, this comes at a steep computational cost, as inference complexity scales linearly with the number of adapters. Furthermore, L2R exhibits significant routing fragility under domain uncertainty: on the Web of Science (WOS) benchmark in a class-incremental setup, routing errors cause a performance drop of more than 10 percentage points compared to an oracle upper bound (79.90% vs. 90.06%), even though the underlying adapter weights still contain the necessary knowledge. This reveals a structural decoupling between parametric retention and effective knowledge access. Furthermore, the empirical verification of these claims is currently hindered by the lack of publicly available source code or reproducible implementation details for L2R,

severely limiting its transparency and immediate viability for enterprise deployment. For these reasons, L2R is not adopted as an experimental baseline in this thesis, despite its conceptual relevance.

In parallel, Buehler and Buehler proposed **X-LoRA** (Mixture of Low-Rank Adapter Experts), a framework designed to achieve deep interdisciplinary synthesis by dynamically mixing a set of specialized, frozen LoRA adapters Buehler and Buehler (2024). Unlike conventional routing mechanisms that operate at a static task or sentence level, X-LoRA introduces an unprecedented level of granularity: it computes scaling coefficients (λ) dynamically for each individual input token and independently across each transformer layer. This token- and layer-level scaling is performed by a lightweight, trainable *scaling head* that consults model hidden states during inference. In order to obtain this structured self-adaptation, X-LoRA employs a two-passforward mechanism: it first performs a "scaling pass" in which the base model runs without adapters, extracting relevant hidden states and feeding them to the scaling head which computes the weight λ. The resulting token-specific weighted averages are then passed to a "forward pass" in which outputs are produced by combining the outputs of different domain experts. This very fine-grained ("soft") gating scheme enables a wide range of knowledge integration flexibility, however, it presents several peculiar challenges when it comes to continuous adaptation: the continued blending of adapters at the token level can have a detrimental "washing-out" effect. When there is a contradiction in parametric information between adapters, that information is instead mixed at the token level and so the architecture faces a difficult task in definitively updating a knowledge fact and resolving it via discrete updates to override a fact. In Section 5, X-LoRA is employed as an experimental baseline, serving as a baseline against which hard and soft routing are compared.

Most recently, to address the specific challenge of parameter interference when merging models, techniques such as **DARE** (Drop And Rescale) and **TIES-Merging** have emerged. DARE Yu et al. (2024) employs a "prune-then-merge" strategy, setting a large fraction of delta parameters to zero to reduce redundancy before rescaling the remaining ones. Building on this, **TIES-Merging** (Trim, Elect, Sign & Merge) Yadav et al. (2023) introduces a mechanism to resolve sign conflicts between models. By trimming noise, electing a dominant sign direction, and disjointly merging updates, TIES prevents conflicting parameters from cancelling each other out. This combination offers a promising theoretical basis for "overriding" old knowledge with new patches, rather than merely averaging them. Although neither DARE nor TIES is directly employed within the PnR framework, this conceptual grounding informs the design rationale of treating each knowledge update as a discrete, additive patch rather than a blended weight change.

However, although such architectures based on modularity and retrieval have their own specific merits, the problem to be described still awaits an answer, which is presented in the next chapter.

3 Research Gap and Objectives

3.1 The Technical and Cognitive Gap

To date, none of the proposed approaches have resolved the problem of catastrophic forgetting in an effective way that scale to real-world. Modular architectures such as L2R (Araujo et al., 2024) showed that task-specific knowledge can be factored out into parametric adapters, but once these architectures scale to realistic settings, two severe issues appear:

1. **Inference Inefficiency:** Optimal routing in current modular architectures usually depends on calculating a soft weighted average for all available adapters with each incoming query. Consequently, the computational load increases linearly as the number of tasks grows, making these methods unworkable when handling hundreds of updates tailored to specific domains.

2. **Routing Fragility:** In situations of domain uncertainty, such as in Class-Incremental Learning setups, the internal routing mechanisms often struggle to correctly select or evaluate the appropriate modules. This routing error leads to a substantial drop in performance when compared to the theoretical maximum. Consequently, even though the underlying parameters retain the knowledge, the system fails to access it.

On the other hand, retrieval-based methods such as RECIPE completely avoid parameter degradation, yet they depend solely on non-parametric knowledge injection. This setup means they cannot access task-specialized parametric experts.

At the same time, research in KE and parameter rewriting mainly focuses on locating outdated facts in global parameters and carefully modifying them (Meng et al., 2022; Pan et al., 2025). However, this approach introduces problematic side effects and runs counter to how biological cognition works. In human thinking, disproven or old links are not usually deleted or replaced; rather, they are inhibited or reduced in priority as a new, stronger connection is created and takes over as the dominant one (Anderson and Green, 2001; Levy and Anderson, 2002).

This aligns with the idea that intelligent systems need to maintain several separate context streams instead of continually modifying one global state (Behrouz et al., 2025). The central research gap, however, is not about fine-tuning parameters or retrieval; it concerns a dynamic re-routing mechanism. This mechanism should isolate conflicting knowledge in distinct parametric modules and replace outdated execution paths while preserving them, shifting the cl paradigm from erasing parameters to inhibiting structure.

3.2 Research Questions, Hypotheses, and Objectives

Based on these technical and cognitive gaps, this thesis investigates how LLMs can continually integrate changing enterprise knowledge without facing the architectural weaknesses of monolithic or soft-routing methods. This study is directed by the following main Research Question:

How can a modular architecture enable Large Language Models to integrate conflicting, domain-specific updates without catastrophic forgetting while maintaining real-time computational efficiency?

To address this question, the proposed framework must meet two main system **Requirements (R)**:

- **R1 (Conflict Resolution):** The framework should actively separate different Knowledge Patches and suppress any conflicting base or historical adapters when contradictions arise, ensuring that the latest information ($\mathcal{D}_{\text{conflict}}$) consistently takes precedence over older knowledge states.
- **R2 (Stability & Efficiency):** The design should separate new updates from established baseline knowledge to prevent catastrophic forgetting of historical domain knowledge ($\mathcal{D}_{\text{base}}$). This way, the cost of ongoing updates stays lower than doing a full retraining.

3.3 Architectural Inspirations

To meet the needs of **R1** and **R2**, the proposed Patch-and-Route framework moves away from the usual global parameter changes. Instead, it combines five key architectural principles mentioned in section 2:

- **Modular Isolation:** Drawing from ongoing moe systems, this framework ensures tight compartmentalization by fully freezing the base llm parameters, which helps avoid the stability-plasticity issue.
- **LoRA Adapters as Containers:** PEFT layers (e.g., LoRA) are utilized as discrete physical storage mechanisms to encapsulate separate temporal knowledge states. Adjusting internal capacity metrics (e.g., scaling rank parameters between standard and highly contradictory updates) provides the necessary parameter isolation to insulate conflicting gradients.
- **Cognitive Inhibition over Deletion:** This critiques the issues with parameter degradation that arise from direct model editing. It is based on the observation that even adapter-based updates cannot consistently erase established parametric knowledge because they lack sufficient spectral update strength. Therefore, the architecture uses a routing layer to inhibit old execution paths rather than rewriting weights. As a result, historical domain states remain intact and can be audited.
- **Non-Parametric Source-Replay:** To reduce the information loss that arises from selecting only one expert, every adapter is explicitly linked to its training data indices ($\text{Adapter}_i = \{\text{Weights}_i, \text{DataIndices}_i\}$). This setup makes it possible to retrieve precise token-level context along with the active weights.

- **Branch-Solve-Merge Execution:** To tackle routing fragility and timing uncertainty during zero-shot domain shifts, the framework splits incoming queries across several potentially suitable experts at the same time, using the base model to combine and generate the final output by considering recent timing.

Anchoring the architecture around these existing fundamentals allows the framework to ingest conflicting data continuously without causing catastrophic forgetting of old data. The actual engineering of the individual components (from dual stage routing to the Expert Pool) is detailed in the following Methodology chapter.

4 Methodology

This chapter explains the **Patch-and-Route** framework and how it was developed. The aim was to develop a practical system for CL, one that stays affordable, grows with more domains, and manages conflicts in knowledge. The layout follows the flow of a query. We begin with the architecture and its inference workflow, then discuss the expert pool and its training, and finish with the two routing strategies (two interchangeable implementations of the same Stage-2 dispatcher, not two separate systems) that Section 6 will assess for resolving conflicts.

The architecture consists of three main parts.

1. **The Frozen Foundation (Base LLM):** a pre-trained, instruction-tuned LLM (Mistral-7B-Instruct-v0.3, 4-bit quantised) that delivers all the linguistic and reasoning abilities. Its parameters are **never changed**.

2. **The Expert Pool (a Mixture-of-Adapters):** a dynamic set of lightweight LoRA modules that hold domain knowledge. It features:
   - **Base Adapters** ($\mathcal{D}_{\text{base}}$)**:** trained on broad initial domain texts, capturing the established knowledge of the domain before any updates from Knowledge Patches.
   - **Knowledge Patches** ($\mathcal{D}_{\text{conflict}}$)**:** an expanding collection of specialized LoRAs, ranging from single-fact updates (e.g. `CEO_Patch_v2` where The CEO is Müller) to larger enhancements to the Base Adapter. Each Knowledge Patch encapsulates a conflicting update $\mathcal{D}_{\text{conflict}}$ that would otherwise overwrite the corresponding $\mathcal{D}_{\text{base}}$ knowledge in a monolithic setup.

3. **The Two-Stage Routing System:** a mixed system that balances automation with enterprise control.
   - **Stage 1: Automatic Domain Routing.** A lightweight MiniLM-L6-v2 + MLP classifier identifies between registered expert domains and an out-of-domain class, either passing it to Stage 2 or directly to the frozen base when the input is outside known domains.

- **Stage 2: Intelligent Dispatcher (Micro-Router).** This automatically routes within the domain selected at Stage 1, directing the query to the fitting Knowledge Patch or Base Adapter. This dispatcher is the one component realised in two interchangeable ways (Section 4.3); everything else in the framework is shared between them.

4.1 The Expert Pool (Domain-Specific Adapters)

Each expert consists of a set of **LoRA** weights, managed via the Hugging Face `peft` library. We maintain two distinct types: large **Base Adapters** (trained on the initial domain corpora) and small **Knowledge Patches** (trained on specific updates).

4.1.1 Training Process

Base Adapters and Knowledge Patches use the same training setup: one optimizer, a single chat-format data pipeline, and the same frozen 4-bit base underneath. The only component that changes is the LoRA rank for each domain. Knowledge that aligns with the base model's priors trains at rank $r = 16$, $\alpha = 32$. In contrast, knowledge that directly contradicts the model's parametric memory trains at rank $r = 32$, $\alpha = 64$. Both configurations follow the $\alpha = 2r$ convention (Rathore et al., 2025); the higher operating point serves two purposes. First, raising α increases the effective scaling factor $s = \alpha/r$: Luong and Chen (2026) demonstrate that standard-scale LoRA updates frequently fail to override entrenched parametric knowledge because their singular values fall below those of the pretrained weights, and that sufficient scaling of α is necessary to push updates past the critical threshold at which they can locally override dominant pretrained directions. At the global model level, this same spectral limitation motivates the inhibition-over-deletion design of the routing layer: rather than attempting to overwrite entrenched base-model priors through parameter edits, the PnR framework routes around them entirely. It is precisely this elevated spectral strength that a $\mathcal{D}_{\text{conflict}}$ Knowledge Patch requires at the local level. Second, intermediate ranks $r \in \{32, 64\}$ represent the empirically robust operating range for capacity-intensive adaptation tasks (Rathore et al., 2025). These specific values are informed by this evidence but chosen heuristically; no rank-specific ablation was conducted within this continual-patching setting. A potential issue arises when a domain corpus includes various distinct relation types. Automatic clustering on embeddings may create poorly separated partitions, leading to experts that mix unrelated facts and obscure the router's signal. We avoid this by using a curated partition, guided by the semantic structure of the corpus, which groups relations into thematically tight families that route more cleanly.

4.2 Inference with the Patch-and-Route Workflow

Inference works as a two-stage pipeline, as illustrated in Figures 1 and 2.

Stage 1: Domain Classification The Stage 1 classifier checks the query to determine if it belongs to a registered expert domain. If it does not fit (such as asking Where is the Eiffel Tower located? or if it is outside any known domains), the request is **sent directly to the frozen LLM**, as shown in Figure 1. No adapters load, and the Expert Pool stays untouched. This approach preserves the model's general abilities and prevents domain adapters from degrading the model's common-sense reasoning. While this gate is designed to strictly bypass the Expert Pool, its practical reliability, and in particular its susceptibility to leaking out-of-domain queries into registered experts, is empirically examined in Section 6.6.

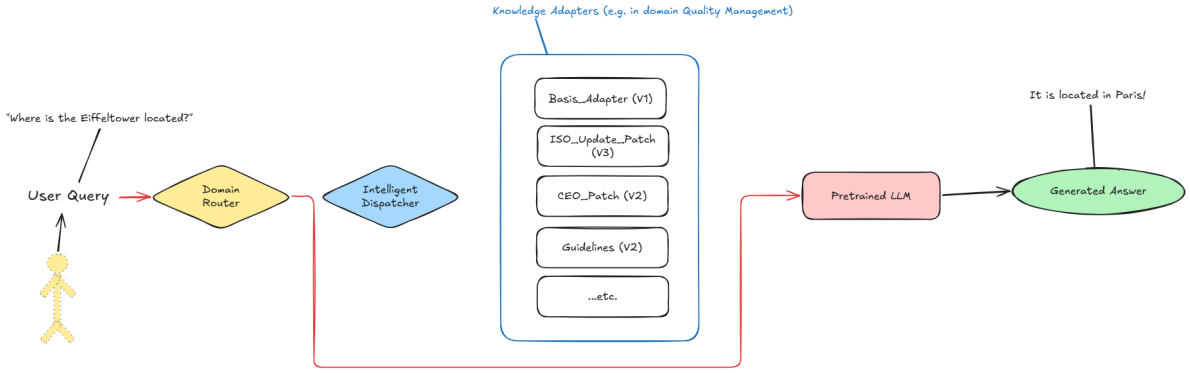

Figure 1: Inference path for an out-of-domain or general-knowledge query. The Stage 1 classifier redirects the request directly to the frozen LLM; the Expert Pool is bypassed entirely.

Stage 2: Intra-Domain Dispatching (Micro-Routing) Only queries that the Stage 1 classifier forwards to a known domain reach this stage. The Intelligent Dispatcher identifies the relevant knowledge components from the available Expert Pool. Figure 2 shows this: a domain-specific question (such as Who became the new CEO?) prompts the Dispatcher to select a Base Adapter along with a recent Knowledge Patch that addresses the update.

Rather than merging the experts, the Dispatcher keeps each set of weights distinct and uses one of two composition strategies:

1. **Single-Expert Selection:** the best-matching expert loads on its own (replacing the frozen base) and, where needed, is complemented with retrieved context instead of another set of weights. This strategy is what the proposed Time-Aware Centroid Router uses in Section 4.3.

2. **Contextual Synthesis (Horizontal Composition):** the prompt is processed by each selected adapter independently to generate separate candidate outputs, which a final synthesis pass (the proposed Parallel-Orchestrator from Section 4.3) then resolves and combines.

Section 4.3 details the two specific routing implementations that complement this design space.

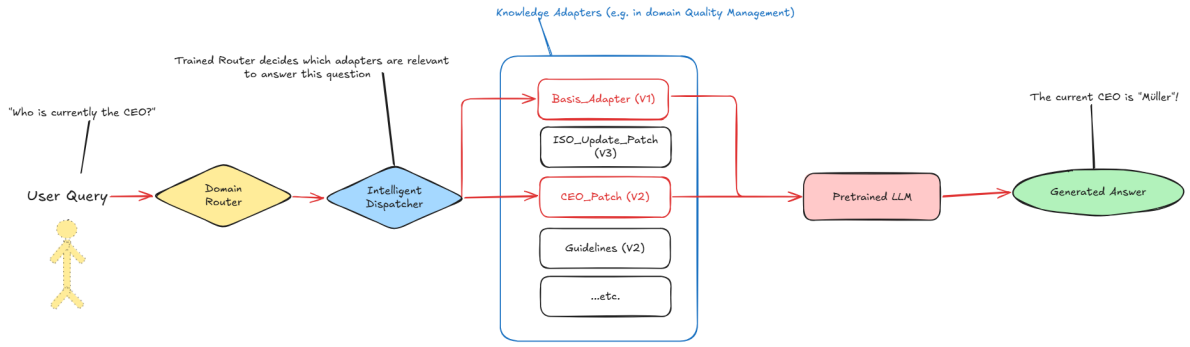

Figure 2: Inference path for a domain-specific query. The Intelligent Dispatcher selects (and, in the synthesis version, composes) the relevant adapters from the pool (specifically a domain Base Adapter and a targeted Knowledge Patch) to generate a contextually appropriate answer within the active domain.

4.3 The Intelligent Dispatcher and Conflict Resolution

We implemented and evaluated two routing setups. Both are implementations of the same Stage-2 dispatcher within the single Patch-and-Route framework: they share an identical Stage-1 domain gate, expert pool, and frozen base, and differ only in how Stage 2 resolves the domain selected at Stage 1. These are precisely the two routing variants referred to in the abstract; neither is a separate system. One is a lightweight embedding-based selector (the **Time-Aware Centroid Router**, Section 4.3); the other is an ensemble-based synthesizer (the **Parallel-Orchestrator**, Section 4.3). Full-page diagrams of both architectures are provided in Appendix 9.6 (Figures 10 and 11).

4.3.1 Time-Aware Centroid Routing with Source-Replay (Embedding-based)

The main implementation, the **Time-Aware Centroid Router with Source-Replay**, combines centroid-based routing with retrieval-augmented replay. Embedding-based routing is not entirely new in Multi-LoRA systems: LoraHub (Huang et al., 2024) assembles LoRA modules on the fly, while systems like S-LoRA (Sheng et al., 2024) and LoRAX (Predibase, 2023) adjust adapter selection for efficient inference. Unlike L2R (Araujo et al., 2024), which trains a router and averages outputs from adapters, our Stage-2 dispatcher takes a straightforward approach: deterministic, embedding-based centroid selection, chosen for efficiency. The inherent limitation of this winner-takes-all selection is potential information loss, which is addressed through a Source-Replay mechanism drawn from retrieval-augmented CL (Gao et al., 2024).

- **Mechanism:** Each adapter has its own centroid vector calculated from training data, stored as a tuple $Adapter_i = \{Weights_i, DataIndices_i\}$, linking the weights

to their training chunks in a vector store. During inference, we embed the query, score it using cosine similarity against the adapter centroids, and hot-swap the highest-scoring expert into the frozen base. A score is considered a match only if it exceeds a similarity threshold τ. We set τ slightly below where in-domain queries typically land, rather than depending on a held-out grid search. This decision increases the recall of genuine domain queries but, in turn, increases the chance that out-of-domain inputs are misdirected to a registered expert. This trade-off between recall and precision is what Section 6.6 examines when the query stream extends beyond the registered domains.

- **Winner selection and time-aware tie-breaking:** The adapter with the highest similarity score wins and is hot-swapped into the frozen base. If two adapters are very close in similarity (e.g., a CEO record from 2020 versus one from 2025), the timestamp decides the tie in favor of the newer authority T_{new}, ensuring that the more recent of the two matches is the one loaded.

- **Always-on Source-Replay:** Regardless of which adapter wins, its own *DataIndices* are queried by default, and the retrieved training chunks feed into the context window. This brings in the complete $Adapter_i = \{Weights_i, DataIndices_i\}$ tuple: the LoRA shifts the distribution towards the correct answer space, while the retrieved text delivers the exact token sequence. If a second adapter also qualifies, the unselected T_{old} can provide a **scoped retrieval** from its *DataIndices*, maintaining access to its historical context even if only one set of weights is loaded.

- **Minimising information loss:** The parametric memory of every non-winning adapter is offloaded to save on VRAM, yet its non-parametric knowledge remains accessible through this targeted replay. The result strikes a balance: it operates at approximately the cost of a single loaded adapter while reclaiming much of what full retrieval augmentation would have provided.

4.3.2 Parallel-Orchestrator Architecture (Ensemble & Synthesis)

The second Stage-2 implementation keeps the same Stage-1 gate, expert pool, and frozen base, and changes only how the dispatcher resolves the selected domain. The Centroid Router commits to a single adapter per query. When multiple adapters plausibly match (for example when a fact has been updated more than once and two temporal patches both exceed the similarity threshold), that hard choice may discard relevant information. The **Parallel-Orchestrator** defers that choice: instead of picking one adapter upfront, it lets all qualifying adapters generate an answer independently, then resolves any conflict in a final synthesis pass. This synthesis step adapts the **Branch-Solve-Merge** pattern (Saha et al., 2024) to parameter-efficient adapters on a shared base, in the spirit of Mixture-of-Agents (Wang et al., 2024a); Branch-Solve-Merge names the resolution mechanism, not a third routing system.

The workflow has three steps:

1. **Select candidate adapters.** After the Stage-1 domain gate identifies the relevant adapter family, the router scores every in-domain adapter by cosine similarity. Adapters that exceed their per-adapter threshold τ_i become candidates. If only one adapter qualifies, the orchestrator short-circuits directly to that adapter and skips synthesis entirely, making it equivalent to the Centroid Router for unambiguous queries.

2. **Generate independently.** Each candidate adapter is hot-swapped into the frozen base in turn and produces its own answer. As with the Centroid Router, each generation is conditioned on that adapter's Source-Replay context (the training chunks retrieved from its own *DataIndices*), so each branch carries both its parametric knowledge and its supporting evidence.

3. **Synthesise and resolve.** The frozen base (with the most recent adapter still loaded) receives all candidate answers together with their deduplicated evidence and produces a single final response. Conflicts are resolved by recency: if two adapters disagree, the one with the later training date takes precedence, and its answer is emitted directly rather than hedged with transitional phrases like "previously X, but now Y."

Section 6.4 compares the two implementations, quantifying the trade-off between the centroid router's efficiency and the orchestrator's robustness under routing ambiguity.

5 Experimental Setup

This section explains how we set up the experiments to test requirements R1 and R2. A central idea underlies the design: to shape the evaluation so that the **joint** edit-success/forgetting metric becomes the main distinguishing factor. Everything mentioned here supports that goal. We use three different update domains, a stability probe that the frozen base meets by design, and three distinct viewpoints on edit success.

5.1 Implementation

Every experiment uses **Mistral-7B-Instruct-v0.3** as the fixed foundation, loaded in 4-bit NF4 quantization (with double quantization enabled, BF16 compute type) through `bitsandbytes`. Adapters are trained following the QLoRA approach (Dettmers et al., 2023): the base weights stay in 4-bit for both training and inference, while the LoRA update matrices are kept and improved in BF16 using a paged AdamW 8-bit optimizer. Each expert acts as a LoRA module (rank $r = 16$, $\alpha = 32$, dropout 0.05, unless noted) applied to all seven linear projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`), trained with Hugging Face `peft` (Mangrulkar et al., 2022) and TRL `SFTTrainer` on top of `transformers` (Wolf et al., 2020) and `datasets` (Lhoest et al., 2021). The parameters of the base model do not change. The Stage-1 domain classifier is a MiniLM-L6-v2 sentence encoder with a small MLP head on top (about 467 KB

checkpoint); the Stage-2 micro-router measures cosine similarity against per-adapter centroids.

5.2 Datasets

The evaluation involves three datasets: two public reproducible benchmarks and a proprietary enterprise case study. We chose them deliberately because they have different structures, so a good outcome is not simply due to one data format.

- **SituatedQA** (SQA) (Zhang and Choi, 2021), focuses on temporal updates. This public dataset has clear temporal annotations: everything before 2019 acts as the stable base ($\mathcal{D}_{\text{base}}$), while data from post-2019 becomes the stream of updates for Knowledge Patches.

- **CounterFact** (CF) (Nanda, 2022; Meng et al., 2023), atomic factoid edits. A controlled benchmark for precise KE. We split the 19,728 train-split records into six Knowledge Patches through a curated Wikidata P-id mapping, grouping the 34 relations into six thematically tight families (physical geography, biographical, linguistic, role/occupation, production/corporate, administrative/structural), each landing in a [2,500, 4,500]-record band (mean $\approx$ 3,288). The grouping criterion is semantic coherence: because every record carries exactly one Wikidata relation, mapping relations to families yields buckets that are internally consistent and cleanly separable at the relation level, rather than partitions defined by surface embedding similarity. An alternative agglomerative clustering on relation embeddings was discarded for failing this criterion, as it merged semantically unrelated relations into a single bucket (for instance, 6,692 records spanning citizenship, occupation, sport-position, and place-of-death). A secondary benefit of the resulting partition is that it distributes conflicts across all six experts and thus also exercises the framework's multi-expert routing.

- **AIT QM Corpus**: A long-form document QA dataset for enterprises. To build it, we used an LLM pipeline to extract core facts from a proprietary Quality Management (QM) corpus and format them into structured question-answer pairs. Because the raw source documents are access-bound, we used these real QM facts to create a semi-synthetic conflict set inspired by CounterFact's design. The real extracted facts provide the current answers, while a teacher model Qwen 3.5 MoE 35B total / 3B active, quantised to 4-bit) generates contradicting, outdated versions using the generation prompt in Figure 8. Each candidate is then passed through a second verification call (Figure 9) that checks whether the two answers genuinely contradict; both prompts are collected in Appendix 9.6. This results in 500 verified contradicting conflict-pairs (285 in German and 215 in English). The base adapter (`base_qm`) trains on the outdated facts, and the patch (`patch_qm_current`) focuses on the current ones, ensuring the conflict the router needs to resolve is both real and directional. The QM evaluation set is categorized into three groups: 500 conflict items (changed facts, testing edit success), 500 stable items (facts that remained

over time, testing retention of the base knowledge), and the shared 1,000 TriviaQA control items (examining forgetting). Figure 3 shows a representative conflict pair as seen from both adapter perspectives.

5.3 Stability Probe ($\mathcal{D}_{\text{control}}$)

To tell routing-induced forgetting apart from plain baseline ignorance, we build a filtered set of 1,000 **TriviaQA (Joshi et al., 2017) QA pairs** on which the unadapted base LLM answers every single one correctly. The filter runs at batch size = 8; final evaluation decodes sequentially (batch size = 1). Because GPU non-determinism produces slightly different greedy paths at the two settings, a small number of borderline samples can flip, leaving a structural floor of FR $\approx 0.6\%$ on the frozen base itself (examined in Section 6.2). Any post-adaptation drop beyond this floor on $\mathcal{D}_{\text{control}}$ then traces to routing interference rather than to a knowledge gap. The entire stability evaluation rests on this. It eliminates a separate baseline run, because the frozen base is the reference. The set is sized to detect forgetting-rate differences of ≥ 1 percentage point at $p < 0.01$. For CounterFact it joins the conflict probe as $\mathcal{D}_{\text{eval}} = \mathcal{D}_{\text{conflict}} \cup \mathcal{D}_{\text{control}}$, where $\mathcal{D}_{\text{conflict}}$ is a uniform-random sample of 1,000 records drawn across all six patches from the training split (fixed seed = 42, shared across all systems). The six experts are trained on the full 19,728 records; edit success is then scored on this shared 1,000-record conflict sample drawn from them.

5.4 Metrics

We show edit success using three separate measurement perspectives. This way, if a parsing or harness artifact appears in one, it registers as a disagreement with the others rather than quietly skewing a key number.

1. **Generation ESR (exact match):** This is the share of conflict records where the system, using deterministic decoding, produces the new (counterfactual or updated) target. It is the edit-success metric that users see. 2. **Teacher-forcing ESR (TF-ESR):** $P(\text{target}_{new} \mid \text{prompt}) > P(\text{target}_{true} \mid \text{prompt})$ during teacher forcing; this is the standard log-probability measure applied in ROME, MEMIT, GRACE, and RECIPE (Meng et al., 2023; Chen et al., 2025b). It captures a successful internal preference even if greedy decoding misses it verbatim. 3. **Strict (containment) ESR:** For lengthy QM answers, target_{new} has to be included, while target_{true} must be absent. This is a locality-aware standard that a simple exact match cannot convey.

We use the single term ESR (edit-success rate) for the generation-based edit-success quantity throughout, and let the matching rule follow the answer format instead of introducing a separate metric name per dataset: exact match for the short, atomic answers of CounterFact and SituatedQA, and strict containment for the long-form answers of AIT QM. (The short-answer rule is what other QA work calls exact match; we report it as ESR to keep one consistent edit-success column across all three domains.) Table 1 summarises the full metric set.

Metric	Test per item	Used for
ESR (exact match)	normalised output = target_{new}	CF, SQA
ESR_{str} (containment)	target_{new} present $\wedge$ target_{true} absent	QM (long-form)
TF-ESR (log-prob)	$P(\text{target}_{new} \mid q) > P(\text{target}_{true} \mid q)$ (teacher forcing)	all
FR (forgetting rate)	$1 - \text{acc}(\mathcal{D}_{\text{control}})$	all (stability)
Judge	Gemma-4 binary correctness verdict	all (semantic)

Table 1: Metric definitions. First three rows are the edit-success (R1) family; last two measure stability (R2) and semantic correctness.

Stability is measured by the system-level forgetting rate $\text{FR} = 1 - \text{acc}(\mathcal{D}_{\text{control}})$, where lower is better and the aim is $\text{FR} \approx 0$. On CounterFact, the usual conflict-free metrics do not hold, given that the frozen base scores are ≈ 0 on the conflict set by design, leaving a normalized conflict-free rate largely undefined. Instead, we provide the complementary, well-defined forgetting rate on $\mathcal{D}_{\text{control}}$. Free-form text can be difficult to parse, so an **LLM-as-judge** (Gemma-4-26B-A4B-it) evaluates every system at once, giving one binary correctness verdict for each answer; the judge column appears next to exact match throughout.

5.5 Baselines

We compare against four declared baselines, chosen to span the design space of continual knowledge injection along two axes: parametric vs. non-parametric updates, and hard (discrete) vs. soft (continuous) routing. These are not convenient strawmen. Each baseline pins one axis to the opposite choice, so any advantage we report ties back to a specific design decision rather than to an under-tuned competitor. Two of the four (X-LoRA and RECIPE) are recent, competitive representatives of their paradigms.

Monolithic LoRA. A single LoRA adapter retrained on the full accumulated corpus (old facts plus new). It removes modularity entirely, providing the cleanest test of whether expert isolation matters, and serves as the primary efficiency and forgetting benchmark. For QM, we also include a sequential monolithic variant (trained first on outdated facts, then on current ones) to control for the multi-expert factor independently of training order.

LoRA + RAG. Monolithic fine-tuning combined with retrieval augmentation over the new documents. This widely-used practical baseline isolates the contribution of external retrieval over a single fine-tuned model, occupying the parametric-update, no-modularity cell of the design space.

X-LoRA (Buehler and Buehler, 2024). Discussed in Section 2, X-LoRA shares the premise that modular LoRA experts are the right unit of knowledge but resolves conflicts through continuous, token-level blending rather than discrete selection. It directly tests

the hard-vs.-soft routing axis: if a weighted average over experts could produce clean factual overrides, the discrete commit central to our design would be unnecessary. We hypothesize that continuous blending will dilute sharp conflicting facts into undecided averages, and that full-stack per-token recomputation will carry a non-trivial inference overhead; both claims are examined in Section 6.

RECIPE (Chen et al., 2025b). Discussed in Section 2, RECIPE is a highly competitive and well-established non-parametric approach to lifelong KE, and represents the strongest publicly reproducible non-parametric position in the design space. We hypothesize that keeping knowledge in the prompt, rather than in parameters, struggles with atomic, deeply internalized facts (as in CounterFact). In such cases, the model's parametric memory often overrides contradicting context injected via the prompt, a scenario where an isolated LoRA patch can succeed.

Retrieval-cache oracle. Beyond the four declared baselines, we include a retrieval-cache oracle as an additional reference ceiling rather than a competing system. It seeds a k-NN store with the evaluation facts themselves and returns the stored answer verbatim on a high-similarity hit, bypassing the LLM decoder entirely. Because the store is seeded from the evaluation data, it provides a near-perfect retrieval-recall upper bound; its purpose is to isolate how much of any remaining gap between PnR and perfect edit success is attributable to routing or generation quality rather than retrieval quality. It is described in full in Section 6.5.

The frozen base model serves as the lower reference on edit success and the 100% reference on $\mathcal{D}_{\text{control}}$. Together, the four baselines bracket Patch-and-Route on every axis it commits to: no modularity (Monolithic), external retrieval (LoRA+RAG), modular but soft routing (X-LoRA), and non-parametric editing (RECIPE). The results in Section 6 therefore read as a controlled isolation of each design choice rather than one aggregate win.

5.6 Protocol and Reproducibility

Every evaluation run uses sequential greedy decoding (temperature = 0, batch size = 1), so generation matches the $\mathcal{D}_{\text{control}}$ pre-filter conditions exactly; any FR > 0 is then a genuine forgetting signal rather than a decoding artefact. All systems see the same 1,000 conflict records and the same 1,000 $\mathcal{D}_{\text{control}}$ probes per domain (fixed seed = 42). All evaluations are zero-shot: the query itself is the entire user turn, with no system prompt, task instruction, or few-shot examples added. Prompt formatting holds identical across Patch-and-Route (PnR), the Parallel-Orchestrator, monolithic, LoRA+RAG, X-LoRA, and the retrieval-cache oracle: each query arrives as a bare Mistral `[INST]...[/INST]` user turn, the same format used during adapter training and the $\mathcal{D}_{\text{control}}$ pre-filter. LoRA+RAG is a partial exception: its retrieved facts are prepended as a structured bullet list (`"Here are some relevant facts..."`) inside the same `[INST]` block, so the chat-template boundary is preserved. RECIPE is the full

exception. It receives the raw query without any chat-template wrapper, matching the completion-style format it was trained on; wrapping it would interleave special tokens between the injected continuous prompts and the generation point and degrade retrieval quality, so the bare-query setting is the most favourable one for RECIPE rather than a penalised one. Each system runs as a query-stateless router (no winner $\Rightarrow$ detach to the frozen base), so the forgetting rate is a property of the configuration and need not be re-measured per query. Experiments ran on NVIDIA A100 (80 GB) GPUs. The complete implementation (the Patch-and-Route framework, all baselines, and the training and evaluation scripts) is publicly available.[2]

6 Results

This section presents the empirical findings, organized around the two requirements from Section 3.2. The main finding comes first: discrete two-stage routing into isolated parametric experts is the only evaluated type that meets both conflict resolution (R1) and stability (R2) at the same time (Sections 6.1–6.2). We then quantify the efficiency aspect of R2 (Section 6.3) and compare the two routing realizations side by side (Section 6.4). The retrieval-cache oracle is presented last, serving as a retrieval-recall ceiling rather than a rival architecture (Section 6.5).

Every number here is drawn from the nine standardized evaluation directories (one per evaluated system configuration) and scored using the Gemma-4-26B-A4B-it external judge. Each system encounters the same 1,000 conflict records and the same 1,000 $\mathcal{D}_{\text{control}}$ probes (fixed seed = 42), through sequential greedy decoding (batch size = 1). The comparison is therefore byte-for-byte identical across methods.

6.1 R1: Conflict Resolution

Table 2 gives the primary conflict-resolution metric for each of the three update domains, with the system-level forgetting rate beside it. In its full multi-expert, two-stage configuration, Patch-and-Route leads every declared baseline on edit-success rate across all three datasets: CounterFact ESR = 30.4% (next best LoRA+RAG 7.7%), SituatedQA ESR = 86.4% (next best LoRA+RAG 29.2%), and the locality-aware AIT QM strict ESR = 62.4% (next best baseline RECIPE 50.0%). Figure 4 plots that joint frontier across the three domains.

[2] `https://github.com/Leon-AW/PnR-framework`

	CounterFact			SituatedQA			AIT QM			Sys.
Method	ESR	TF	J	ESR	TF	J	ESR_{str}	TF	J	FR ↓
Frozen Base	0.0	13.0	0.8	0.0	—	30.3	1.2	37.2	4.4	**0.6**
X-LoRA	0.0	29.3	3.6	0.4	—	21.6	0.0	32.8	0.4	74.8
Monolithic LoRA	0.0	41.7	1.6	20.1	—	45.9	23.4	69.2	39.8	100.0
LoRA + RAG	7.7	**87.3**	**35.1**	29.2	—	42.3	21.4	**91.6**	33.6	99.5
RECIPE	0.3	14.1	1.5	19.8	—	60.3	50.0	68.8	56.2	47.8
Parallel Orchestrator	**33.5**	76.2	33.6	**86.6**	—	**88.2**	57.0	**91.6**	90.2	**0.6**
PnR	30.4	75.0	30.8	86.4	—	**88.2**	**62.4**	90.2	**92.2**	**0.6**

Table 2: Conflict-resolution (R1) and stability (R2) results. ESR = generation edit-success rate (exact match for CF/SQA; strict containment ESR_{str} for QM); TF = teacher-forcing log-prob ESR; J = Gemma-4 judge; FR = $1-\text{acc}(\mathcal{D}_{\text{control}})$, lower is better. Bold marks FR at the frozen-base floor. All values in %. Full matrix in Appendix Table 11.

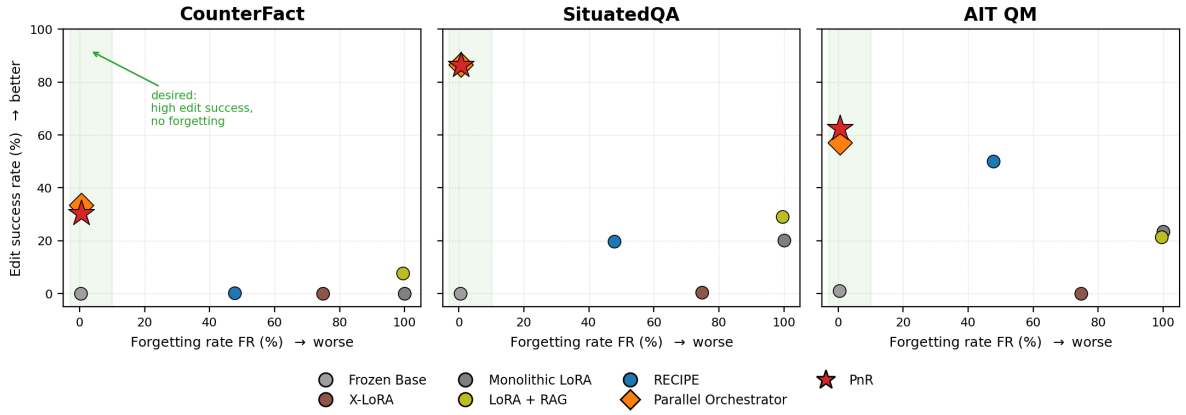

Figure 4: The joint edit-success / forgetting frontier across the three update domains. Each panel plots a system's domain edit-success rate (CF and SQA exact-match ESR, QM strict containment ESR) against its $\mathcal{D}_{\text{control}}$ forgetting rate (FR); the shaded band marks the no-forgetting region (FR $< 10\%$). The desired corner is top-left. Only the two routing systems (PnR, Parallel Orchestrator) reach it in every domain, with high edit success at the frozen-base forgetting floor, whereas every editing baseline trades one axis for the other and the frozen base edits nothing.

The baselines fail in domain-characteristic ways, each confirming a prediction from Section 4. **Monolithic LoRA** and **LoRA+RAG** edit the datasets they are tuned for moderately well, but they pay for it with total control forgetting (FR $\geq 99.5\%$): fine-tuning the single adapter on conflicting facts overwrites the very parametric knowledge $\mathcal{D}_{\text{control}}$ probes. **X-LoRA**'s token-level soft gating never overrides decisively. Its CF and QM exact-match ESR sit at 0.0%, the "washing-out" effect predicted for continuous blending (Buehler and Buehler, 2024). **RECIPE** is the strongest non-parametric baseline

(QM strict ESR 50.0%, SQA Judge 60.3%), yet its continuous-prompt injection guarantees no locality, so it still forgets (47.8% FR on SQA $\mathcal{D}_{\text{control}}$) and collapses to 0.3% on the atomic CounterFact edits, where overriding a strongly-held parametric belief is hardest of all. A sequentially trained monolithic variant (old facts, then new) is the closest single-adapter comparison. It edits QM better than the parallel-trained row above (conflict ESR 54.4%, strict 50.6%; Appendix Table 11), yet it never leaves the forgetting regime of shared-weight fine-tuning. It edits and forgets simultaneously, which is precisely the trade-off the routing design escapes.

The teacher-forcing view (TF, the efficacy metric of the knowledge-editing literature) is most informative on CounterFact, where the generation ESR looks modest. PnR's CF generation ESR of 30.4% climbs to a TF-ESR of 75.0%: in three cases out of four the routed specialist assigns higher likelihood to the counterfactual target than to the original fact, even where greedy decoding does not surface it word for word. The edit, in other words, largely succeeds internally, and the gap to generation ESR is a decoding-margin effect rather than a failed edit. The modest absolute generation figure is itself informative: CounterFact is the hardest regime for the user-facing metric, because each edit overwrites a single atomic token against strong pretraining priors, and exact-match generation requires that token to surface verbatim while the original prior is still competitive at the decoding margin. This penalty is specific to atomic factoids. In the long-form regime that the enterprise use case targets, where the answer is a span rather than a single contested token, user-facing generation success is markedly higher (62.4% QM strict ESR, Table 2). The CounterFact generation number is therefore best read as a lower-bound stress test on atomic overwriting, not as the operating point for the intended deployment setting. This is the metric that aligns directly with ROME, MEMIT and RECIPE. It also makes the central point clear: on TF alone two baselines beat PnR (LoRA+RAG reaches 87.3% CF / 91.6% QM TF-ESR), but only by destroying control knowledge, their FR is 99.5%. The contribution is therefore no single dominant cell. It is the joint pattern of Table 2, high ESR/TF and a forgetting rate at the frozen-base floor, which only the two routing systems reach.

6.2 R2: Stability

The right-most column of Table 2 is the stability result. In its full configuration Patch-and-Route hits the editing levels above while pinning the forgetting rate at 0.6%, exactly the frozen base measured on the same $\mathcal{D}_{\text{control}}$ build, and it does this on all three benchmarks. No baseline with comparable editing comes close. Every method that touches the shared parameters pays for its edits with catastrophic control forgetting.

Why is the FR not exactly 0%, and what accounts for the residual 0.6%? It comes from exactly 6 out of the 1,000 $\mathcal{D}_{\text{control}}$ records, and the routing does not cause the drop. All six are sent straight to the frozen base. The base holds no patches, so it should not forget, yet it still misses those same six queries that have byte-identical answers (see Appendix Table 11, footnote ‡). A breakdown is given in Table 3.

In the top four rows the model answers correctly, and only the rigid string check disagrees (any human grader would pass them). The bottom two are genuine slips,

Question	Model answer	Accepted key	Verdict
Where would a troglodyte live?	"Cave"	"in cave"	correct
Autobiography of the first post-Apartheid SA president?	"Long Walk to Freedom" (with title prefix)	"long walk to freedom"	correct
Where was the 1903 World Fair held?	"Saint Louis"	"st louis"	correct
Which quantity has direction and magnitude?	"Vector"	"vector quantity"	correct
Whom did Pope John Paul II succeed?	"John XXIII"	"John Paul I"	wrong
Who said "middle age is when your age starts to show around your middle"?	"Eric Overmeyer"	"Bob Hope"	wrong

Table 3: The six $\mathcal{D}_{\text{control}}$ records missed identically by PnR, the Parallel Orchestrator, and the frozen base (the entire 0.6% FR). Four are essentially correct answers the exact-match key does not recognise; only the bottom two are genuine base-model errors.

though highly unstable ones, and they come entirely from the $\mathcal{D}_{\text{control}}$ construction pipeline. The dataset keeps only questions the frozen base answers correctly, but that filter runs in batches (size 8) while final evaluation decodes sequentially (size 1), and due to GPU non-determinism the two regimes are not bit-identical. A handful of borderline greedy choices flip. The base answered these two correctly at construction time, then diverged when decoded one at a time later, which is why both still pass teacher forcing: the correct answer keeps the highest log-probability, the knowledge stays intact, only the sampled path moved. Removing them would mean rebuilding $\mathcal{D}_{\text{control}}$ from scratch and re-running the full nine-system sweep, all for a 0.2 pp shift that changes no conclusion. We forgo that compute and report the 0.6% as-is. Routing-induced forgetting is therefore effectively 0; the 0.6% is a structural floor, compounding exact-match strictness with a GPU non-determinism artefact, so every miss was inherited rather than created.

Stability holds by construction. The reason FR is close to zero merits clarification, since the mechanism, not just the number, is what matters. PnR never changes the base weights, and the Stage-1 domain classifier sends out-of-domain and control queries to the frozen base. FR near zero is therefore guaranteed unless the classifier misroutes a control query into an adapter. The framework does not address catastrophic forgetting by reducing interference within shared parameters; it avoids the locus of forgetting by using continual addition-plus-gating instead of continual parametric learning. The key R2 question thus reduces to one: does the gating allow control queries to leak into the adapters? The evidence indicates that it does not, as seen in the Stage-1 classifier's performance on $\mathcal{D}_{\text{control}}$ (validation macro-F1 0.978, out-of-domain (`ood_trivia`) recall 96.6%), which is what keeps the measured FR at the frozen-base level. The main caveat

is that this evidence comes from the evaluated class partition; Section 6.6 directly tests how the guarantee behaves when queries fall outside of it. This is a design choice with a clear trade-off (Section 7), rather than a forgetting-mitigation algorithm.

6.3 R2: Efficiency

R2 also asks the framework to cut "the computational cost of updates compared to monolithic retraining." That arm splits in two: the per-query inference cost (Section 6.3.1) and the per-update training cost as the knowledge base grows (Section 6.3.2).

6.3.1 Inference cost

Table 4 reports per-query latency and peak VRAM, pulled per-record from each system's saved CounterFact `D_eval` run, so every method is timed on the same short-answer workload. We report median latency, since it is robust to the first-query cold-start and the adapter-load spike.

Method	Median ms/q	p95 ms/q	Peak VRAM (MB)
Frozen Base	429	1,889	4,995
X-LoRA	27,208	28,376	7,114
RECIPE	2,083	2,110	5,617
Monolithic LoRA	1,828	3,055	5,061
LoRA + RAG	1,297	2,591	5,136
Parallel Orchestrator	849	1,949	5,330
PnR	**457**	2,034	5,374

Table 4: Per-query inference cost on the CounterFact $\mathcal{D}_{\text{eval}}$ workload. Latency numbers are relative; runs shared GPU type but not necessarily load.

Two findings follow. First, **routing overhead is negligible**: PnR two-stage (457 ms/q) lands within ∼7% of the frozen base (429 ms/q), so a MiniLM domain classifier, a centroid lookup, and one hot-swapped adapter together cost almost nothing over plain base generation. Peak VRAM (∼5.4 GB) holds at the single-adapter level, because experts load one at a time rather than sit resident, so the pool can grow without an N-adapter memory blow-up. Second, **discrete switching beats continuous blending by ∼50–60×**: X-LoRA (27,208 ms/q) recomputes its soft mixture over all adapters at every decoding step, putting the prediction that soft gating is both costly and indecisive into hard numbers. PnR delivers higher ESR and roughly 60× lower latency.

6.3.2 Update cost

This is the cost R2 identifies directly. Both paths have the same LoRA setup and step budget, so comparing them at a fixed number of steps would largely cancel out and mask the effect. The relevant measure is cost per update as the knowledge base expands.

For adding increment k, PnR trains a patch using only increment k's data and leaves the base and prior patches alone, keeping its per-update cost constant at $O(\text{increment})$. A monolithic adapter, by contrast, must retrain on the total corpus to incorporate increment k without losing previous information, so its per-update cost grows linearly with the number of updates.

Table 5 shows one such update operation directly: it compares training a single PnR patch for one increment with retraining a full monolithic adapter on the entire corpus. Both were run back-to-back on a single A100 to ensure the numbers are comparable in terms of hardware.

Update operation	Records	Steps	Wall-clock	s / 1k rec	Peak VRAM (MB)
PnR patch (1 increment)	3,340	53	419 s	125.5	5,209
Monolithic (full corpus)	19,728	309	2,463 s	124.8	9,671

Table 5: Measured cost of one update operation on a single A100 (matched per-example exposure, effective batch 16, identical LoRA rank). PnR trains on one increment only; monolithic retrains on the full accumulated corpus.

One monolithic retrain already costs 5.9× the wall-clock of a single patch (2,463 s vs. 419 s) and 1.9× the peak training VRAM, purely because it reprocesses the entire corpus. The decisive effect, though, is cumulative: with every new update the monolithic retrain must again cover the whole grown corpus, while each PnR patch stays at the cost of one increment.

Strategy	Total training steps over 6 updates	vs. PnR
PnR (one patch per update)	318	1.00×
Monolithic (retrain cumulative)	1,099	3.46×

Table 6: Cumulative training cost over six CounterFact relation-family updates (53 steps per increment). PnR's per-update cost is flat ($O(\text{increment})$); monolithic retrains on the full grown corpus each time.

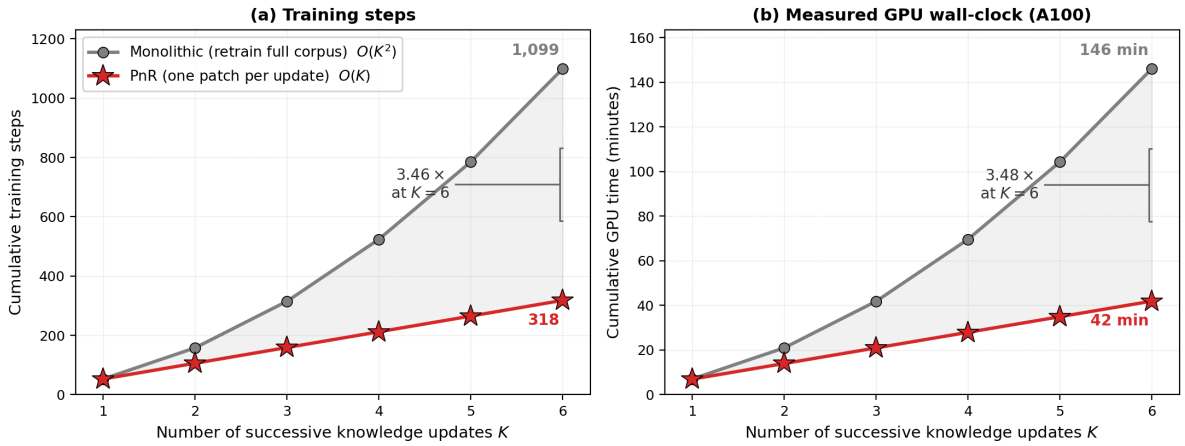

Figure 5: Cumulative training cost as the knowledge base grows, in abstract training steps (a) and measured A100 GPU wall-clock (b). PnR adds one patch per update, so its per-update cost is flat and the cumulative cost grows linearly ($O(K)$); the monolithic baseline retrains on the whole accumulated corpus at every update, so its cumulative cost grows quadratically ($O(K^2)$). The shaded region is the redundant recomputation the monolithic path pays; it widens with every update, reaching 3.46× at $K = 6$ (Table 6), or roughly 146 vs. 42 GPU-minutes. Panel (b) rescales each curve by its measured seconds-per-step from the matched cost runs in Table 5 (MLflow `train_runtime`).

The cost ratio is $(K + 1)/2$ in the number of updates K, measured at 3.46× for $K = 6$ (Table 6) and widening steadily as more updates arrive; Figure 5 plots the two cumulative-cost curves directly. This is the structural efficiency case for the modular design: the marginal cost of an update is decoupled from the size of the accumulated knowledge base. The monolithic baselines cannot do that. They pay twice over, with the catastrophic forgetting of Table 2 and this linearly-growing retraining cost.

6.4 Comparison of the Two Routing Realisations

We evaluated the architecture on two conflict-resolution realisations: a hard winner-takes-all Time-Aware Centroid Router ("PnR Routing") and a generate-then-synthesise Parallel-Orchestrator (Section 4.3). They sit on the same frontier: in matched long-form QM strict ESR they are only 5.4pps apart (62.4% hard-routing vs. 57.0% ensemble) and both have 0.6% FR. The implication is that routing into separate parametric experts two stages down is the architecture, not the conflict resolution. This architecture class provides the trade-off curve; hard selection versus ensemble synthesis is the efficiency–robustness trade-off that follows (the Parallel is more robust to routing confusion but incurs 849 ms/q of ensemble evaluation costs from Table 4 compared with 457 ms/q for hard routing.) We set hard routing ("PnR Routing") as the default. The two realizations tie on stability, within a few points on every edit cell (the Parallel-Orchestrator outperforms the short, atomic CounterFact and SituatedQA metrics by up to 3.1 pps, and hard routing

outperforms the long-form QM strict ESR by 5.4 pps). Therefore the choosing factors are pragmatic rather than accuracy-based: hard routing almost halves inference latency (457 vs 849 ms/q), restricts maximum memory usage to one active expert, and selects exactly one expert whose output can be targeted by the verbatim-recall extension described in Section 9. Parallel-Orchestrator has advantages if route robustness is paramount and the latency penalty of evaluating an ensemble is acceptable.

6.5 The Retrieval-Cache Oracle: a Recall Ceiling, Not a Competing Architecture

Beyond the declared scope of the exposé, we also evaluate a verbatim retrieval-cache oracle. It is no new architecture: what we evaluate here is a textbook k-nearest-neighbour (kNN) retrieval cache, a non-parametric key–value memory of the kind the kNN-LM (Khandelwal et al., 2020) and retrieval-augmented generation (Lewis et al., 2020) lines introduced. Concretely, the evaluated path is a single retrieval step with no learning in the loop: (i) a sentence encoder embeds the query; (ii) cosine similarity picks the nearest record in a store whose keys are fact embeddings and whose values are the stored answer strings; (iii) if the top-1 similarity clears a confidence threshold, the cache returns that record's value verbatim and never runs the decoder, otherwise it abstains to the frozen base. This is therefore 1-nearest-neighbour retrieval with a reject option, pushed to its degenerate copy-the-neighbour limit: where RAG conditions generation on the retrieved item, the cache simply returns it. Since the store is seeded with the evaluation facts themselves, every query's gold answer is already a key, so a near-exact self-match clears the threshold and the gold string is copied straight out. We include it for completeness, and because two of its results support the central thesis. Its headline number, however, requires careful reading.

Table 7 gives the oracle ablation. That headline 98.3% CounterFact ESR is **not a routing result**: on a high-similarity retrieval hit, the verbatim-return path returns the stored answer string and never runs the LLM. Because the store is seeded from the evaluation facts, the number measures lossless retrieval recall against an oracle store, conceptually a verbatim-return variant of RECIPE's Knowledge Sentinel rather than anything the surrounding machinery learned. The timing agrees: on a bypass hit the oracle returns in $\sim$169 ms/q, faster than any generating system in Table 4, because it does a dictionary lookup instead of running the decoder at all. The same table carries the proof. The no-bypass ablation, which forces every answer through the activated specialist, scores 0.4% CF ESR, and that $\sim$98-point gap shows the learned path adds essentially nothing to the bypass number.

Read carefully, the oracle reinforces the case for parametric experts on two fronts. First, the **no-bypass result is the real "oracle-as-a-learning-system" number**: at 0.4% CF ESR, retrieval-conditioned context on its own cannot steer the frozen model to override its parametric belief, which is exactly why parametric experts (as in PnR) are needed on atomic factual conflicts. Second, the **bypass-versus-RECIPE contrast isolates the failure mode of non-parametric editing**: under the same retrieval,

Routing signal	Bypass	CF ESR	QM Strict ESR	FR ↓
τ_{low} centroid	on	98.3%	81.0%	0.6%
τ_{low} centroid	off	0.4%	55.6%	0.6%
MLP classifier	on	85.4%	—	6.1%
MLP classifier	off	0.5%	—	6.1%

Table 7: Retrieval-cache oracle ablation. `bypass` rows return the stored answer verbatim without running the LLM (retrieval-recall ceiling); `no-bypass` rows force generation through the activated specialist. '—' = configuration not run for QM.

RECIPE's continuous-prompt injection yields 0.3% CF ESR where verbatim substitution yields 98.3%. The failure is therefore not retrieval quality; prompt-injection does not override, and literal substitution does. The gap that remains, between routing to a parametric expert (PnR, 62.4% QM strict ESR) and perfectly recalling a stored edit (the ∼81–98% bypass ceiling), is what drives the most concrete extension in Section 9: adding lossless verbatim recall on store-confirmed edits to the PnR routing path.

6.6 Open-Stream Routing Stress Test: Measuring the Closed-World Boundary

The stability result of Section 6.2 holds by construction, and that construction rests on a premise: every query the gate ever sees belongs to one of the four classes $\{cf, sqa, qm, ood\}$ the Stage-1 classifier was trained on. The motivating enterprise setting breaks that premise constantly, since an open update stream presents queries from domains the gate has never seen. This subsection measures what happens at that boundary. It is a post-hoc stress test of the routing protocol, separate from the evaluation in Section 5.4, and we present it as such.

Setup. We assemble 1,000 held-out queries, 200 from each of five domains the classifier never saw in training: biomedical research (PubMedQA, Jin et al., 2019), legal consumer-contract QA (LegalBench, Guha et al., 2023), financial filings (financial-QA-10K, virattt, 2024), science exams (SciQ, Welbl et al., 2017), and open-domain factoids (Natural Questions, Kwiatkowski et al., 2019). They run through the unchanged production pipeline (the same checkpoints, the same routing thresholds, greedy decoding, and seed used in every $\mathcal{D}_{\text{eval}}$ run), with no retraining and no new class. Natural Questions is a boundary probe by design: it is general-knowledge factoid QA that the gate could legitimately send to the frozen base through the `ood_trivia` class, so every headline number appears both overall and with it excluded; the principal claim rests on the four clearly out-of-distribution domains.

The gate leaks roughly a third of held-out queries. Table 8 reports, per domain, how often a query slips past the frozen base and loads an expert adapter (the routing

leak, the only path that can change an answer). The headline is 31.2% across the four distinct domains (38.1% with the boundary probe included). The closed-world guarantee does not carry over: faced with genuinely novel input, the gate passes about one query in three through to an expert adapter. The leak rate tracks distance from the trained partition. SciQ science questions, which read like general knowledge, route correctly to the base (3.0%), whereas finance (50.0%) and legal (44.0%), with their distinctive professional phrasing, get pulled into the experts.

Held-out domain	n	Stage-1 confident	Routing leak [95% CI]
SciQ (science)	200	2.5%	3.0% [1.4, 6.4]
PubMedQA (medical)	200	35.5%	28.0% [22.2, 34.6]
LegalBench (legal)	200	52.5%	44.0% [37.3, 50.9]
financial-QA-10K (finance)	200	45.5%	50.0% [43.1, 56.9]
Natural Questions (probe)	200	56.5%	65.5% [58.7, 71.7]
Overall, excl. NQ	800	34.0%	**31.2% [28.1, 34.5]**
Overall	1,000	38.5%	38.1% [35.1, 41.2]

Table 8: Open-stream routing leak on 1,000 held-out queries from five domains the Stage-1 classifier never saw in training. Stage-1 confident = argmax $\neq$ `ood_trivia` at $p \geq 0.7$; Routing leak = the full pipeline loads an expert adapter. Wilson 95% confidence intervals in brackets.

The failure is at Stage 1, and it is structural. The leak does not refute the gating concept; it localises to one replaceable component, and within it to a single design assumption. Of the 381 leaked queries, 364 are confident Stage-1 misclassifications (argmax $\neq$ `ood_trivia` at $p \geq 0.7$); only 17 slip in through the low-confidence centroid fallback. The root cause is the classifier's label space. It was trained only on questions from four content distributions (CounterFact, SituatedQA, German quality-management, and TriviaQA), and `ood_trivia` is the last of these, a particular factoid distribution rather than a "none-of-the-above" reject option. A four-way softmax with no abstention class has no choice but to map every novel query onto one of four known domains; "outside all of them" is simply not a state it can express. MiniLM duly places these professional-domain questions nearer the $cf/sqa/qm$ training populations than TriviaQA-style general knowledge, and the softmax head says so with confidence. Stage 2 then compounds the error instead of catching it: a confident Stage-1 prediction does more than narrow the candidate pool, it relaxes the centroid veto from the global $\tau = 0.45$ to the production fallback of 0.30, loosening the one geometric safety net at the precise moment the classifier has committed in error. Neither bound was tuned on a held-out sweep. We set $\tau = 0.45$ just below the band where in-domain queries land (around 0.50) so genuine matches are not turned away, and the drop to 0.30 rested on the assumption that Stage 1 had already removed the out-of-domain queries, leaving Stage 2 free to be more permissive. The open-stream result is exactly the falsification of that assumption:

when Stage 1 is confidently wrong, the relaxed bar lands on the very queries it was presumed to have filtered out. The leaked centroid similarities run from a floor of exactly 0.300 (the relaxed threshold) up to a median of 0.451, essentially the un-relaxed one, so the within-domain selector is itself well calibrated; those queries should never have been allowed to reach it under a loosened veto. The boundary probe makes the point most cleanly: Natural Questions, the one held-out set the gate could defensibly have sent to the base, leaks more (65.5%) than any genuinely distinct domain, which says the classifier has learned the surface of four distributions, not an "in-domain versus out-of-domain" boundary.

Most leaks are benign, but the contract is still violated. A routing leak only matters if it changes the answer. Table 9 sets, for every leaked query, the routed answer against the frozen-base answer on the same query. Only 44.1% of leaks (168/381) actually change it, so the effective answer-corruption rate is 13.6% excluding the probe (16.8% overall), well under the routing-leak figure. The damage clusters in the adapters whose training data most directly overrides general knowledge: the counterfactual (69.7%) and quality-management (66.3%) experts rewrite the answer far more often than the situated-QA family (33.2%). One caveat applies. An unchanged answer means the leak caused no output corruption, not that the system behaved correctly. The design contract for an out-of-domain query is frozen-base equivalence, and loading any adapter breaks it, whether or not the sampled string happens to match.

Leaked-into family	Changed / leaked	Change rate [95% CI]
cf (counterfactual edits)	23 / 33	69.7% [52.7, 82.6]
qm (quality-management)	59 / 89	66.3% [56.0, 75.3]
sqa (situated facts)	86 / 259	33.2% [27.8, 39.1]
All leaks	**168 / 381**	**44.1% [39.2, 49.1]**

Table 9: Conditional damage on the 381 leaked queries: share whose routed answer differs from the frozen base, by adapter family. Wilson 95% confidence intervals in brackets.

Reading. This clarifies the cutoff that Section 6.2 left vague. Out of distribution, the gate locks and FR drops to the frozen-base floor; out of distribution, the rationale for the build breaks at some defined rate (a 31% routing leak which degrades $\sim 14\%$ of held-out answers) and remains contained within the Stage-1 classifier, not routing. Two adjustments arise for diagnosis. The cheaper one is to stop relaxing the centroid veto when the Stage-1 decision is confident; this is deferred to Section 9. The more fundamental one is to give the gate a genuine reject capability (an open-set score, since `ood-trivia` is silent on this; Section 6.7 addresses it, and the resulting leak-versus-recall trade-off is quantified by the α-sweep in Appendix Table 13), which is what we proceed

to evaluate. The detailed open-stream analysis (per domain, including for flowed-in adapters) is Table 12 in the Appendix.

6.7 Mitigating the Open-Stream Leak: an Open-Set Gate

The stress test isolated the cause to one failure modality: all 364 of the 381 leaks are confident misclassifications. Therefore a max-softmax or a margin reject cannot help (since the detector is already convinced), meaning the signal must be from external information. To address this, we introduce a per-class Mahalanobis distance to the trained class manifolds in the gate's embedding space (Lee et al., 2018); a confidently identified intra-adapter-domain example is instead dispatched to the frozen base if its query example is distant from the manifold of the identified class. This detector is additive and switchable; disabling this module cleanly reverts the system to its initial production behavior, which yields the reported before/after differences. See Appendix 9.6 for details of method and calibration.

The evaluation uses disjoint splits by construction. We use the training split to estimate the classifier's statistics for the detector. Validation statistics are then used to calibrate the threshold, before performing a single evaluation pass on a freshly held-out test set (at an allowed false-reject budget of 5%). The detector is then evaluated on three fresh out-of-distribution domains that it has not encountered during fitting or calibration: 600 queries in total drawn from MMLU-Pro (professional exam questions), MATH-500 (competition math), and Belebele `deu_Latn` (German reading comprehension, bilingual). The five diagnostic domains of the open-stream stress test (Section 6.6) are reused only as a separate consistency check on the same frozen detector (Appendix Table 14). We pre-commit the false-reject budget to $\alpha = 5\%$, the single tuning parameter, which we set before touching the test set.

On the English domains, the regime the leak concerns, the veto removes about two thirds of it (17.2% $\rightarrow$ 5.8%, Table 10), catching exactly the confident-but-far queries at an in-domain recall cost of 2.7%, well within budget. Running on the original 5 diagnostic domains the same frozen detector reproduces the published leak perfectly in the before pass and removes it from 31.2% to 13.3% at the same cost (Table 14 in Appendix), and a sweep over α puts 5% at the knee of the trade-off (Table 13 in Appendix). The one domain which is largely unaffected is German (97.5% $\rightarrow$ 96.5%, a nearly perfectly clean removal to 0.1

7 Discussion

The results in Section 6 are reported and read locally, next to the tables that produce them. In this chapter we move away from the individual numbers and ask what they mean collectively: what kind of claim the framework supports, how far that claim generalises, and where it stops holding. We then make explicit the threats to that reading.

Fresh OOD domain	Routing leak before	after
MATH-500 (math)	10.0	**0.5**
MMLU-Pro (professional)	24.5	**11.0**
Belebele DE (german)	97.5	96.5
English-only (math + prof.)	**17.2**	**5.8**
Overall	44.0	36.0

Table 10: Open-set veto on the newly held-out OOD test (not encountered during fitting or calibration), at the pre-committed $\alpha = 5\%$. A routing leak is defined as the fraction of queries that load an expert adapter; in-domain recall incurs a cost of 2.7% as a result. The complete α-sweep and the diagnosis-domain consistency check can be found in Appendix Tables 13 and 14. Every value is in %.

7.1 From Solving Forgetting to Relocating It

The main finding is that Patch-and-Route reaches a joint edit-success and forgetting-rate frontier that none of the four established baselines reach (Section 6.1). The broader point is not that the framework lessens interference within a shared parameter set; it is that it avoids entering a state where interference happens altogether. Section 6.2 stated this in terms of measurement: stability is maintained by design since the frozen base serves as the reference and every update exists in its own isolated expert. The broader implication is that the stability-plasticity dilemma (French, 1999) has not been resolved on its own terms. Instead, it has been replaced with a different issue.

This trade merits attention. Continuous parametric learning constitutes a balancing act: every gradient step that acquires new knowledge risks degrading prior knowledge, and both pressures act on the same weights. The issue that Patch-and-Route addresses (determining, for any incoming query, whether it should be directed to a registered expert or to the fixed base) is an open-set recognition problem. This substitution is beneficial for three reasons, which hold regardless of the benchmark scores. First, the new problem is bounded: it concerns only one routing choice per query rather than the entire model's state. Second, it is localised to one replaceable component, the Stage-1 gate, rather than spread across the whole parameter set. Third, it is measurable: the open-stream stress test in Section 6.6 could pinpoint exactly where the idea breaks down, which cannot be done as readily for the catastrophic forgetting that occurs within shared weights. The challenge of continual learning has thus shifted, from a difficult optimisation problem to a more tractable recognition problem, and that shift, rather than a single benchmark success, is what the framework demonstrates.

This reframing is not specific to Patch-and-Route. In Section 2, RECIPE's diagnosed weakness, a Knowledge Sentinel which allows a corrupted continuous prompt to flow into its output, is in essence identical in structure to our own Stage-1 gate's failure to catch and filter out out-of-distribution queries (Section 6.6). It is in fact the two sides of the link that related work points to meeting: that by seeing how our own gate leaked, we can

observe that RECIPE's sentinel failure is our routing leak appearing in a new context. That this failure appears in a model as different as a retrieval-enhanced prompt writer also explains why we frame the open-set recognition challenge as the common unsolved issue for this genre.

7.2 The Architecture, Not the Resolution Mechanism, Closes the Curve

As established in Section 6.4, the two implementations of routing-the single-expert Time-Aware Centroid Router and the ensemble Parallel-Orchestrator-achieve identical frontiers despite very different means of resolving conflicts. Conversely, every baseline that abandons discrete routing into isolated experts-soft-gated blending, shared-weight fine-tuning, and non-parametric prompt injection-fails to reach that frontier (Section 6.1). Together, these results have a broader implication: The general phenomenon closing the trade-off is a characteristic of the class of models that keep their bases static and choose amongst isolated experts by way of discrete selection, not a peculiarity of either the time-aware centroid router or the parallel orchestrator. The key aspect is discrete choice, the choice to commit to one expert, rather than one that blends experts-which is the dimension on which the soft-gated baseline was designed to differ from a standard selection process. This brings the empirical findings into alignment with the architectural intuition that drove this paper: that the right path toward effective continual adaptation lies in separating context into individual, isolated streams, as opposed to continually updating one master copy, and therefore should be understood as an example of a broader family of design patterns (Behrouz et al., 2025). Because the two realisations are equivalent on the frontier, the choice between them is operational rather than one of accuracy: we recommend the hard-routing centroid variant as the default for its lower latency and its single, inspectable expert selection, and reserve the ensemble orchestrator for deployments where robustness to routing confusion outweighs its inference cost.

This same separation then recasts the baseline failures not as three separate phenomena but rather a single one. Monolithic LoRA and LoRA+RAG write to the same shared weights, and thus forget; the continuous mixture blends where it should override, never fully trusting the newest expert; and the non-parametric writer inserts into the prompt, lacking control over the parametric pathways. Despite the varied surface phenomena in Section 6.1, these can all be understood as symptoms of the same problem: knowledge storing and knowledge selection are tied together in the same part of the model, and Patch-and-Route provides the mechanism by which they are disconnected.

7.3 Scalability and the Single Point of Failure

The efficiency results in Section 6.3 show that Patch-and-Route shifts the cost of continuous updating away from the three axes where monolithic retraining struggles. The per-update training cost remains constant as the knowledge base expands, peak memory stays at the level of just one loaded expert, and inference latency is close to the frozen

base. The principal implication is that the framework is, on the dimensions that usually make continuous learning costly, largely unconstrained: the expert pool can grow without requiring additional hardware or compute. These numbers, however, come from a single-stream research setup on one device: they report the per-update and per-query costs, not the behaviour under multi-tenant loads. Paging hundreds of adapters in and out of GPU memory at high frequency is a separate systems problem. This is precisely what dedicated multi-LoRA serving stacks such as S-LoRA and LoRAX (Sheng et al., 2024; Predibase, 2023) are designed to address; the routing aspect discussed here is complementary to that infrastructure and would layer on top, so evaluating costs in realistic multi-user scenarios requires deployment-scale testing.

Describing the framework as "scalable" without qualification would be the wrong reading. The efficiency results show only that the residual risk has been collapsed onto one dimension. The stability guarantee from Section 6.2 and the wide range of newly admissible domains depend on the same component (the Stage-1 gate directing out-of-domain requests correctly to the frozen base), and the open-stream leak from Section 6.6 is what occurs if that component fails. The stability limitation of the framework and its knowledge-breadth scaling limitation are thus not two distinct failings, but a single one viewed from both perspectives, namely the open-set classification challenge of Section 7.1. This is a stronger and more informative assertion than that the system scales well along some directions and badly along others; it means there is a single point of failure for which the system may need strengthening. The mitigation experiment in Section 6.7 is a first step toward showing that this one component can be improved independently of everything else.

A final design cost is conceptual rather than empirically derived: it does not appear next to any of the tables. Since the underlying parameters are always anchored, the system does not consolidate them into a unified connected model; it retrieves individual specialists rather than synthesising them. In other words, the system is explicitly built as an efficient librarian rather than a synthesiser. If the goal, as this thesis claimed, is a conflict-resolving system, that cost is acceptable. The implications for building a system that can learn over long spans of time are addressed in Section 9.

7.4 Threats to Validity

Construct validity. The forgetting rate is close to zero largely by definition, not entirely by empirical invention: the control set is constructed so that the frozen base always answers all items correctly, and the frozen base serves as the reference, so the low forgetting rate is in part a constructed effect of the set's construction (Section 6.2). This is an intentional design trade off to make non-zero forgetting a clean signal of routing, but it does mean that the stability result should be interpreted as "routing does not inject forgetting into this structurally-grounded zero" not a more robust empirical claim that the mechanism successfully overcomes forgetting. We have chosen to represent success through three views of edit-success precisely so that a specific parsing or harness artefact does not present itself as a result.

External validity. The evaluation covers three different types of updates, yet it relies on one frozen base model and a single closed world of registered domains. That closed world is the principal threat. The open-stream test (see Section 6.6) and the residual that mitigation could not address (see Section 6.7) both point to the same issue: conclusions made within the trained partition do not carry over once the query stream exits it. The bilingual residual illustrates this problem most clearly. In this case, German queries that were out-of-distribution could not be distinguished from the similar in-domain class. This is a blind spot that the gate inherits from its label space, not something the routing principle can resolve. Generalising to other base models, to a broader range of domains, and to truly open query streams is therefore a design argument rather than a concrete measurement at this stage.

Internal validity. Two design parameters were set based on reasoned heuristics instead of through a search process. First, there are the LoRA ranks for base and conflict experts (see Section 4). The second parameter is the routing similarity threshold τ, which is chosen to prioritize the recall of genuine domain queries rather than being fine-tuned on a separate grid (refer to Section 4.3). Both parameters are noted where they are established, and they influence how to interpret the results: the scores reported reflect one valid operating point, not an optimised condition. The recall-favouring τ also aligns with the charge for the open-stream leak later on. The retrieval-cache oracle (in Section 6.5) provides a perspective from the other side, distinguishing how much of the remaining edit-success headroom can be attributed to routing and generation rather than retrieval quality.

8 Conclusion

This thesis aimed to determine whether a modular architecture could help a large language model handle conflicting, domain-specific updates without serious forgetting, at a cost suited to continual enterprise use rather than one-time retraining. The issue was a gap that neither parametric fine-tuning nor retrieval alone has managed to bridge: shared-weight editing forgets important information, while non-parametric retrieval cannot ensure that a new fact will override an old, entrenched one.

The main finding is that this is achievable by separating where knowledge is kept from how it is chosen. The Patch-and-Route framework keeps the base model unchanged, places each update in its own LoRA expert, and resolves conflicts with a discrete two-stage routing instead of mixing or rewriting weights. Across three different update types and both routing realizations, this design achieves a balance of edit success and forgetting rates that none of the four established baselines reaches, maintaining the forgetting rate at the frozen base level while staying within about seven percent of the base inference speed and keeping the training cost per update steady as the knowledge base increases. A recurring observation is that it is the architectural family, not merely the specific conflict resolution method, that shapes the trade-off curve.

However, there is a key limitation that the thesis has acknowledged. Patch-and-Route

does not prevent interference within a shared parameter set; it avoids the setting in which interference occurs, so its stability rests on the routing gate correctly channeling out-of-domain queries to the frozen base. The open-stream stress test revealed this premise weakening once queries ventured outside the trained partition: the gate allows a substantial share of genuinely new inputs into the experts, and a portion of those leaks corrupt the answers.

These limitations define the result without undermining it. The leak was traced back not to the routing principle but to one replaceable part, the Stage-1 classifier, which lacks a real reject option, while the within-domain selector performed as expected. The major findings regarding efficiency and stability hold within the operational context the framework was designed for, and this boundary has been measured rather than assumed. What the thesis shows is therefore more limited than "catastrophic forgetting solved" and more robust than a single benchmark win: with proper routing, ongoing integration of conflicting knowledge can occur with almost no forgetting and minimal inference overhead, and the conditions for that guarantee are now laid out clearly.

That shift, from a difficult plasticity-stability optimisation problem to a more tractable open-set recognition one, is the contribution worth carrying forward. The next chapter pursues its two most immediate consequences: giving the gate a genuine option to abstain, and combining the routing decision with lossless recall on confirmed edits to realise the headroom exposed by the retrieval-cache limit.

9 Future Work

The analysis singled out a single piece of mechanism that affects both the stability guarantee and the knowledge-breadth scaling of Patch-and-Route: the routing gate and its associated closed-world assumption. The most natural paths for future work then are ones that augment this mechanism, recuperate some of what the single-expert selection is currently sacrificing, and address operating concerns only obvious after years of system usage, not mere benchmark runs. The former two result from limitations we evaluated in Section 6, and the latter result from the architecture design itself.

9.1 A Reject-Capable Routing Gate

The open-stream leak (Section 6.6) points to a Stage-1 classifier with no "none of the above" option, that thus sends confident mass into registered domains even if its input is in none of the classes. Section 6.7's open-set gate is a step in the right direction: the next logical step is a gate that is explicitly trained on yielding abstention rather than applied after-the-fact as a distance veto and explicitly measured on how well it separates a linguistically "in-domain" structural class from a linguistically "out-of-domain" language (the bilingual residual we failed to overcome with the present mitigation). Since it is a self-contained building block, improvements here benefit the entire system without re-engineering experts or the resolution mechanism.

9.2 Lossless Recall on Store-Confirmed Edits

As described in Section 6.5, the retrieval-cache oracle indicated a headroom in recall (on which routing is based) over actual verbatim retrieval. Routing combined with a lossless recall path on store-confirmed edits would mitigate this while still leveraging the parametric experts, which give the framework its ability to override ingrained facts, effectively acting as a high-recall, high-coverage expert without outright replacing the base model. The key missing piece here is how to handle the handover to the two routes, which we suspect would have implications on the joint frontier of Section 6.1, and how the additional store cost influences latency: a hit on the store bypasses decoding altogether, a miss incurs the store latency on top of falling back to the router, which in turn adds decode latency unless it too incurs a store hit. The overall speed gain is dependent on how often such a store hit can be achieved during operation.

9.3 Governing Patch Proliferation and Centroid Crowding

The framework that answers each update with a new singleton expert accrues the expertise for each knowledge unit. Over the long term, this leads to three side-effects. One, the set of experts for a fact turns into a surface that one could use to audit changes or govern versions; tracing back a mistaken prediction to the patch that introduced it is a system operation problem we do not run here. Two, in case updates concerning a single fact occur multiple times, the centroid space will eventually be so densely packed that any two subsequent versions no longer overlap: in these cases, the decision is entirely made by the time-based tiebreaker described in Section 4.3. Three, we do not implement a system to manage the update process, assume for example directional updates to facts and that only one source will attempt to patch a fact simultaneously. In a real enterprise environment, this is a more complex issue and should be properly addressed by some form of ownership and decision process rather than being left aside as an assumption. In addition, it would be interesting to study the effect of increasing updates on within-domain decision quality, investigate how patches that have been superseded by newer ones could be retired or aggregated and when forgetting could be initiated without reverting the history of a fact.

9.4 The Router as Its Own Continual-Learning Problem

The Stage-1 gate is trained on a static label space. Any genuinely new domain means that we need to add and retrain a new gate. This means the continual learning task is moved from the base model to the router. This is precisely the way the framework makes the “original” problem “more favorable to solve” rather than “solve” it (Section 7.1). A gate that could include new domains continually while retaining the previous ones would fully “relocate” it.

9.5 Toward Autonomous Knowledge Orchestration

Every approach above still presumes a human is the director of the pipeline: an observer notices a piece of factual information is out-of-date, collects the patch training data, and decides if a router extension is warranted. The governance assumption in Section 9.3 makes this precisely that. A more ideal end-state removes the human from the inner loop and turns update assimilation into an autonomous 3-step process. The first is conflict detection: an agent which monitors the document stream to detect incoming content contradicting what is currently encoded in the experts, ideally differentiating real context–memory conflicts from merely contradictory sources along the lines of the taxonomy in Xu et al. (2024). The infrastructure already includes the required primitive: the LLM-judged contradiction detection used to build the AIT QM conflict set in Section 5 is identical to that operation, just repurposed from training data collection to monitoring. Because this judge served as a construction and scoring tool rather than an independently validated component, its detection reliability would itself have to be established before it could be trusted to drive autonomous updates; otherwise the autonomy layer would inherit an unverified component at its most consequential point. The second is autonomous patch generation: if a conflict is confirmed, the system prepares the patch training data (re-using the semi-synthetic data generation framework from QM), initiates the adapter training, and tests the candidate patch against held-out test data before allowing it into the adapter set. The third, is autonomous expansion of the Stage 1 gate to adapt to the discovery of new topics and categories: the self-expanding-gate problem described in Section 9.4.

What such an end-state cannot eliminate is human responsibility for what the model claims. A system that trains and uses its own "truths" without oversight takes on the full burden of validation and auditability. The patch-provenance concerns from Section 9.3 shift from operational to legal requirements. The realistic goal is thus not total autonomy but changing the human role from creator to approver. A human-on-the-loop would oversee and approve machine-suggested patches, handling the work of detection, data gathering, and training while ensuring a person is accountable for every accepted change.

9.6 Scheduled Consolidation and the Synthesis Trade-Off

Finally, the structural cost from Section 7.3 (that a frozen base never synthesizes across the experts it routes among) opens the question of whether the system might periodically consolidate by halting and baking approved patches into the base weights, both to reinstate cross-domain reasoning and prevent the expert pool from growing indefinitely. This connects to the biological motivations we presented in the introduction; isolated LoRA experts act as the fast, hippocampal store of new information within the architecture, while the frozen base acts as long term neocortical storage. The missing component is the process of consolidation by which in the complementary-learning-systems account, sleep helps mediate transfer of new experiences from the hippocampus to the neocortex by replaying them into long-term memory storage (McClelland et al., 1995; Schapiro et al., 2018). If it were to be implemented, the systems own cycle of artificial sleep would

be one of scheduled consolidation, baking a recently settled pool of patches back into the neocortical base. However, consolidation is far from a free maintenance process. A naive consolidation that simply re-trains the base on the merged patches re-enters the regime of shared weights that Patch-and-Route was explicitly designed to avoid, and therefore re-introduces precisely the risk of forgetting which the architecture bypasses. The model-merging techniques discussed in Section 2 offer a more controlled route: rather than averaging patches into the shared weights, a TIES-style trim-elect-sign-and-merge or a DARE prune-and-rescale step (Yadav et al., 2023; Yu et al., 2024) could fold a settled pool of experts into the base while explicitly resolving the sign and magnitude conflicts that naive retraining leaves to chance. This is the natural endpoint of the additive-patch rationale established in the related work: the same grounding that motivated treating each update as a discrete, additive patch also indicates how those patches might later be consolidated without reverting to blended weight updates. The research question thus becomes not whether to consolidate, but whether the benefit of an integrated world model outweighs the cost of deliberately re-engaging the stability–plasticity dilemma on a controlled schedule, and how far a merging-based consolidation can reduce that cost relative to naive retraining.

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154.

Anderson, M. C. and Green, C. (2001). Suppressing unwanted memories by executive control. *Nature*, 410(6826):366–369.

Araujo, V., Moens, M.-F., and Tuytelaars, T. (2024). Learning to route for dynamic adapter composition in continual learning with language models. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 687–696, Miami, Florida, USA. Association for Computational Linguistics.

Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., and Abdelrazek, M. (2024). Seven failure points when engineering a retrieval augmented generation system.

Behrouz, A., Razaviyayn, M., Zhong, P., and Mirrokni, V. (2025). Nested learning: The illusion of deep learning architectures. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Buehler, E. L. and Buehler, M. J. (2024). X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design.

Chen, Q., Wang, C., Wang, D., Zhang, T., Li, W., and He, X. (2025a). Lifelong knowledge editing for vision language models with low-rank mixture-of-experts. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9455–9466.

Chen, Q., Zhang, T., He, X., Li, D., Wang, C., Huang, L., and Xue, H. (2025b). Lifelong knowledge editing for llms with retrieval-augmented continuous prompt learning.

Chen, W., Zhou, Y., Du, N., Huang, Y., Laudon, J., Chen, Z., and Cui, C. (2023). Lifelong language pretraining with distribution-specialized experts. In *International Conference on Machine Learning*, pages 5383–5395. PMLR.

Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. (2022). Toy models of superposition. *Transformer Circuits Thread*.

Fan, Y., Wang, Y., Liu, L., Tang, X., Sun, N., and Yu, Z. (2025). Research on the online update method for retrieval-augmented generation (rag) model with incremental learning.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2024). Retrieval-augmented generation for large language models: A survey.

Gao, Y., Xiong, Y., Wu, W., Li, B., Zhong, Y., and Wang, H. (2026). U-niah: Unified rag and llm evaluation for long context needle-in-a-haystack. *ACM Transactions on Information Systems*, 44(3):1–30.

Guha, N., Nyarko, J., Ho, D. E., Ré, C., et al. (2023). LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

Gururangan, S., Lewis, M., Holtzman, A., Smith, N. A., and Zettlemoyer, L. (2021). Demix layers: Disentangling domains for modular language modeling. *arXiv preprint arXiv:2108.05036*.

Gutierrez, B. J., Shu, Y., Qi, W., Zhou, S., and Su, Y. (2025). From rag to memory: Non-parametric continual learning for large language models.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huang, C., Liu, Q., Lin, B. Y., Pang, T., Du, C., and Lin, M. (2024). Lorahub: Efficient cross-task generalization via dynamic lora composition.

Huang, Y. and Huang, J. X. (2026). A survey on retrieval-augmented text generation for large language models. *ACM Computing Surveys*, 58(12):1–38.

Interrante-Grant, A., Varela-Rosa, C., Narayan, S., Connelly, C., and Reuther, A. (2025). Scaling performance of large language model pretraining.

Jin, Q., Dhingra, B., Liu, Z., Cohen, W., and Lu, X. (2019). PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Barzilay, R. and Kan, M.-Y., editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2020). Generalization through memorization: Nearest neighbor language models.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., et al. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics (TACL)*, 7:453–466.

Lazari, A., Salvan, P., Cottaar, M., Papp, D., Rushworth, M. F., and Johansen-Berg, H. (2022). Hebbian activity-dependent plasticity in white matter. *Cell Reports*, 39(11).

Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411.

Lee, K., Lee, K., Lee, H., and Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Levy, B. J. and Anderson, M. C. (2002). Inhibitory processes and the control of memory retrieval. *Trends in Cognitive Sciences*, 6(7):299–305.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Lhoest, Q., Villanova del Moral, A., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., et al. (2021). Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184.

Liang, Y.-S. and Li, W.-J. (2024). Inflora: Interference-free low-rank adaptation for continual learning.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Luong, H.-C. and Chen, L. (2026). Why lora fails to forget: Regularized low-rank adaptation against backdoors in language models. *arXiv preprint arXiv:2601.06305*.

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. (2022). Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. (2023). Locating and editing factual associations in gpt.

Meng, K., Sharma, A. S., Andonian, A., Belinkov, Y., and Bau, D. (2022). Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Nanda, N. (2022). counterfact-tracing. Dataset of 21,919 factual relations with true/false targets, adapted from CounterFact (ROME) and designed for tracing/editing; recommends measuring logit difference between true and false targets.

Pan, H., Wang, X., Cao, Y., Shi, Z., Yang, X., Li, J., and Wang, M. (2025). Precise localization of memories: A fine-grained neuron-level knowledge editing technique for llms. *arXiv preprint arXiv:2503.01090*.

Piochon, C., Kano, M., and Hansel, C. (2016). Ltd-like molecular pathways in developmental synaptic pruning. *Nature neuroscience*, 19(10):1299–1310.

Predibase (2023). Lorax: Multi-lora inference server. `https://github.com/predibase/lorax`.

Rathore, D., Kumar, V., Bansal, C., and Moitra, A. (2025). How much is too much? exploring lora rank trade-offs for retaining knowledge and domain robustness. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 1003–1013.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Saha, S., Levy, O., Celikyilmaz, A., Bansal, M., Weston, J., and Li, X. (2024). Branch-solve-merge improves large language model evaluation and generation.

Schapiro, A. C., McDevitt, E. A., Rogers, T. T., Mednick, S. C., and Norman, K. A. (2018). Human hippocampal replay during rest prioritizes weakly learned information and predicts memory performance. *Nature communications*, 9(1):3920.

Sheng, Y., Cao, S., Li, D., Hooper, C., Lee, N., Yang, S., Chou, C., Zhu, B., Zheng, L., Keutzer, K., Gonzalez, J. E., and Stoica, I. (2024). S-lora: Serving thousands of concurrent lora adapters.

Shi, H., Xu, Z., Wang, H., Qin, W., Wang, W., Wang, Y., Wang, Z., Ebrahimi, S., and Wang, H. (2024). Continual learning of large language models: A comprehensive survey.

Soman, S. and Roychowdhury, S. (2024). Observations on building rag systems for technical documents.

Song, M., Liu, R., Wang, X., Jiang, Y., Xie, P., Huang, F., Zhou, J., Herremans, D., and Poria, S. (2025). Demystifying deep search: a holistic evaluation with hint-free multi-hop questions and factorised metrics.

Su, W., Tang, Y., Ai, Q., Yan, J., Wang, C., Wang, H., Ye, Z., Zhou, Y., and Liu, Y. (2025). Parametric retrieval augmented generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1240–1250.

Sun, Y., Ming, Y., Zhu, X., and Li, Y. (2022). Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning (ICML)*.

virattt (2024). financial-qa-10k. A dataset of 7,000 financial question-answer pairs derived from 10-K filings, keyed by company ticker and filing year.

Wang, F., Wan, X., Sun, R., Chen, J., and Arik, S. O. (2025). Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models.

Wang, J., Wang, J., Athiwaratkun, B., Zhang, C., and Zou, J. (2024a). Mixture-of-agents enhances large language model capabilities.

Wang, L., Zhang, X., Su, H., and Zhu, J. (2024b). A comprehensive survey of continual learning: Theory, method and application.

Wang, Y., Feng, S., Wang, H., Shi, W., Balachandran, V., He, T., and Tsvetkov, Y. (2023). Resolving knowledge conflicts in large language models. *arXiv preprint arXiv:2310.00935*.

Welbl, J., Liu, N. F., and Gardner, M. (2017). Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text (W-NUT)*, pages 94–106.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing.

Xu, R., Qi, Z., Guo, Z., Wang, C., Wang, H., Zhang, Y., and Xu, W. (2024). Knowledge conflicts for LLMs: A survey. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8541–8565, Miami, Florida, USA. Association for Computational Linguistics.

Xu, Z., Liu, Z., Yan, Y., Wang, S., Yu, S., Zeng, Z., Xiao, C., Liu, Z., Yu, G., and Xiong, C. (2026). Thinknote: Enhancing knowledge integration and utilization of large language models via constructivist cognition modeling.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. (2023). Ties-merging: Resolving interference when merging models.

Yang, Y., Zhou, J., Ding, X., Huai, T., Liu, S., Chen, Q., Xie, Y., and He, L. (2024). Recent advances of foundation language models-based continual learning: A survey.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. (2024). Language models are super mario: Absorbing abilities from homologous models as a free lunch.

Zhang, M. J. and Choi, E. (2021). SituatedQA: Incorporating extra-linguistic contexts into QA. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2026). A survey of large language models.

Zhao, Y., Du, X., Hong, G., Gema, A. P., Devoto, A., Wang, H., He, X., Wong, K.-F., and Minervini, P. (2024). Analysing the residual stream of language models under knowledge conflicts. *arXiv preprint arXiv:2410.16090*.

Appendix

base_qm (trains on outdated facts)	patch_qm_current (trains on current facts)
Q: What requirements apply for the management of records in the LKR if they are not specified in another regulation?	Q: What requirements apply for the management of records in the LKR if they are not specified in another regulation?
A: For records that are not specified in any other regulation, the management of records according to M03 (M03-AIT-LKR) applies. This requirement defines the processes, responsibilities, and archiving requirements for all records in the LKR.	A: For records that are not specified in any other regulation, the management of records according to M04 (M04-AIT-LKR) applies. This requirement defines the processes, responsibilities, and archiving requirements for all records in the LKR.
base_qm (trains on outdated facts)	patch_qm_current (trains on current facts)
Q: Wer ist die erste Ansprechperson im LKR bei einer unangekündigten Kontrolle durch das Arbeitsinspektorat (AI) in Wien?	Q: Wer ist die erste Ansprechperson im LKR bei einer unangekündigten Kontrolle durch das Arbeitsinspektorat (AI) in Wien?
A: Die erste Ansprechperson ist der Betriebsrat.	A: Die erste Ansprechperson ist die Sicherheitsfachkraft (SFK).
base_qm (trains on outdated facts)	patch_qm_current (trains on current facts)
Q: Where does the initial registration of visitors take place at Giefinggasse?	Q: Where does the initial registration of visitors take place at Giefinggasse?
A: The initial registration of visitors takes place at the reception of the GG1 building. This reception serves as the first point of contact and orientation aid for customers, guests, suppliers, and partners.	A: The initial registration of visitors takes place at the reception of the GG4 building. This reception serves as the first point of contact and orientation aid for customers, guests, suppliers, and partners.

Figure 3: Three AIT QM conflict pairs shown from both adapter training perspectives. Each row shares the same question; only the highlighted attribute differs between base_qm (outdated, red) and patch_qm_current (current, blue). Attribute types: norm identifier (qm_conflict_00328, EN), responsible role (qm_conflict_00149, DE), and location (qm_conflict_00257, EN). All pairs were verified as CONTRADICT by Qwen3.5-35B-A3B. Generation prompt: Appendix 9.6.

	SituatedQA			CounterFact				AIT QM					System	
Method	ESR	F1	J	ESR	F1	TF	J	ESR	F1	TF	Str	J	FR↓	J_{ctrl}
Frozen Base	0.0	8.7	30.3	0.0	0.1	13.0	0.8	1.4	16.5	37.2	1.2	4.4	0.6‡	98.8
X-LoRA	0.4	10.4	21.6	0.0	0.4	29.3	3.6	0.0	7.1	32.8	0.0	0.4	74.8	60.8
RECIPE	19.8	39.9	60.3	0.3	0.6	14.1	1.5	52.6	49.0	68.8	50.0	56.2	47.8	86.2
Monolithic LoRA	20.1	38.0	45.9	0.0	0.3	41.7	1.6	25.2	38.2	69.2	23.4	39.8	100.0	13.3
LoRA + RAG	29.2	40.3	42.3	7.7	14.6	87.3	35.1	22.0	33.7	91.6	21.4	33.6	99.5	12.3
Parallel Orchestrator	**86.6**	89.2	**88.2**	**33.5**	**33.5**	**76.2**	**33.6**	61.6	63.7	**91.6**	57.0	90.2	0.6	98.8
PnR	86.4	**89.5**	**88.2**	30.4	30.4	75.0	30.8	**66.8**	**69.2**	90.2	**62.4**	**92.2**	**0.6**	**98.8**
Retrieval-Cache Oracle (τ, bypass)†	89.3	91.1	91.0	98.3	98.3	57.7	98.3	86.2	100.0	88.8	81.0	100.0	0.6	98.8
Retrieval-Cache Oracle (τ, no-bypass)	89.3	91.1	91.0	0.4	7.6	57.7	32.7	59.2	63.7	88.8	55.6	88.6	0.6	98.8

All values in %. † retrieval-cache oracle ceiling (Section 6.5), not an architecture result.
‡ Frozen-base FR/J_{ctrl} measured on the rebuilt $\mathcal{D}_{\text{control}}$ (`tc_/qz_` build) used by the Phase-5 routing systems, so the column is like-for-like. On that set the frozen base misses the same 6 of 1,000 records as PnR and the Parallel Orchestrator (identical predictions), confirming the proposed architectures add zero routing-induced forgetting. Of those 6, four are exact-match artefacts on essentially-correct answers (e.g. "Cave" vs. gold "in cave"; "Saint Louis" vs. alias "St Louis"; "Vector" vs. "vector quantity") and only two are genuine base-model errors (Pope John XXIII instead of John Paul I; a misattributed quotation). All 6 satisfy the teacher-forcing check (the gold answer scores highest), so even these are decoding divergences rather than erased knowledge.

Table 11: Complete per-metric results matrix for all evaluated systems. ESR = generation edit-success; TF = teacher-forcing log-prob ESR; Str = strict containment ESR; J = Gemma-4 judge. `bypass` rows return the stored answer verbatim (retrieval-recall ceiling); `no-bypass` rows are the learned-path numbers. '—' = not applicable / not scored. All values in %.

Domain	n	Routing leak [95% CI]	Leaked-into ($cf/sqa/qm$)	Damage
SciQ (science)	200	3.0% [1.4, 6.4]	1 / 0 / 5	3/6 (50.0%)
PubMedQA (medical)	200	28.0% [22.2, 34.6]	5 / 38 / 13	13/56 (23.2%)
LegalBench (legal)	200	44.0% [37.3, 50.9]	11 / 61 / 16	30/88 (34.1%)
financial-QA-10K (finance)	200	50.0% [43.1, 56.9]	4 / 42 / 54	63/100 (63.0%)
Natural Questions (probe)	200	65.5% [58.7, 71.7]	12 / 118 / 1	59/131 (45.0%)
Overall, excl. NQ	800	**31.2% [28.1, 34.5]**	21 / 141 / 88	109/250 (43.6%)
Overall	1,000	38.1% [35.1, 41.2]	33 / 259 / 89	168/381 (44.1%)

Of the 381 leaks, 364 are confident Stage-1 misclassifications (argmax $\neq$ `ood_trivia`) and only 17 enter via the low-confidence centroid fallback; median leaked centroid similarity is $0.451 \approx \tau = 0.45$. See Section 6.6.

Table 12: Open-stream routing stress test, full per-domain breakdown. Leaked-into = adapter-family distribution of leaked queries ($cf/sqa/qm$). Damage = share whose routed answer differs from the frozen base. Eff. corrupt. = changed answers over all 200 domain queries. Wilson 95% confidence intervals throughout.

Open-set gate: method and calibration. The detector (Section 6.7) models the three adapter classes $\{cf, sqa, qm\}$ as Gaussian manifolds in the MiniLM embedding space the Stage-1 gate already uses; `ood_trivia` is not modelled, since it is already routed to the frozen base and adding it would only make the gate more permissive. The shared (tied) covariance is estimated with Ledoit–Wolf shrinkage (Ledoit and Wolf, 2004), a per-class covariance being infeasible (qm has $\sim$450 fit samples in 384 dimensions). A single global threshold proved unfair across classes (it rejected 32% of genuine qm calibration queries against $\sim$5% for the larger classes), so the threshold is calibrated per class, keyed on the Stage-1 prediction, restoring a per-class in-domain false-reject of {cf 5.0%, qm 8.0%, sqa 4.8%} at $\alpha = 5\%$. A non-parametric k-NN score (Sun et al., 2022) over the same embeddings is available as a robustness alternative.

α	English OOD leak after	In-domain recall cost
before	17.2	0.0
1%	8.0	0.6
2%	7.0	1.1
5%	**5.8**	**2.7**
10%	1.8	9.2
15%	1.8	13.8

Table 13: Leak-versus-recall trade-off over the false-reject budget α on the fresh English OOD domains (companion to Section 6.7). $\alpha = 5\%$ (pre-committed) is the knee: beyond it the recall cost rises about three times faster than the leak falls. German moves only to 92.5% even at $\alpha = 15\%$, confirming its residual is structural. All values in %.

Diagnosis domain	Routing leak before	after
SciQ (science)	3.0	1.0
PubMedQA (medical)	28.0	**7.5**
LegalBench (legal)	44.0	**9.0**
financial-QA-10K (finance)	50.0	35.5
Natural Questions (trivia-adj.)	65.5	54.0
Overall	38.1	21.4
Overall, excl. NQ	**31.2**	**13.3**

Table 14: Consistency check: the same frozen open-set detector on the five diagnosis domains of Section 6.6 (dev-OOD, secondary to the fresh-set headline). The "before" column reproduces the published leak exactly, confirming the mitigation harness is faithful to the production pipeline. Same 2.7% in-domain recall cost. All values in %.

LLM-as-a-Judge Prompt Templates

The supplementary LLM-as-a-Judge evaluation (Section 5.4) uses `google/gemma-4-26B-A4B-it` (int4, Gemma 4 MoE, 26 B total / 3.8 B active parameters) as an independent judge model. Two prompt templates are applied depending on the evaluation task; placeholder fields are filled at inference time and shown in `{braces}`. For multi-alias gold answers the aliases are concatenated with `|` as a separator. Deterministic greedy decoding (`do_sample=False`) is used throughout. The model is instructed to output exactly one word, either `CORRECT` or `INCORRECT`, and the verdict is parsed with a regular-expression boundary match (`\bCORRECT\b` / `\bINCORRECT\b`); responses that match neither are counted as unparseable and excluded from the judge accuracy denominator.

```
You are an impartial evaluator of factual question-answering systems. Your
job is to decide whether a system prediction conveys the SAME FACTUAL
CONTENT as a reference answer.

Question: {question}
Reference answer(s) (any one is acceptable): {gold}
System prediction: {prediction}

Rules:
- Different wording, ordering, or formatting are acceptable. Examples:
  "August 15, 1947" matches "15 August 1947".
  "Paris" matches "the city of Paris".
  "8" matches "eight".
- Extra surrounding explanation does NOT invalidate a correct answer, as
long as the core fact is asserted somewhere in the prediction.
- A prediction that contradicts the reference is INCORRECT, even if it
sounds confident.
- A prediction that is irrelevant, evasive, refuses to answer, or is empty
is INCORRECT.
- Length, style, fluency, and politeness are irrelevant. Score ONLY on
factual correctness with respect to the reference.

Respond with EXACTLY one word: CORRECT or INCORRECT. No other text. No
punctuation. No explanation.
```

Figure 6: Factoid judge prompt (`JUDGE_PROMPT_FACTOID`, version v1.0). Applied to **SituatedQA** and **AIT QM** splits, where the gold answer is the expected correct fact. Implemented in `src/eval/external_judge.py`.

```
You are an impartial evaluator of knowledge-editing systems. The system
has been edited with a counterfactual fact, and you must decide whether
its prediction asserts the counterfactual content (NOT the original true
fact).

Question: {question}
Counterfactual target the system was edited to assert: {gold}
System prediction: {prediction}

Rules:
- The reference is the COUNTERFACTUAL target, not the real-world fact. The
system is considered CORRECT iff its prediction asserts the counterfactual
content.
- Different wording, ordering, or formatting are acceptable.
- A prediction that asserts the original true fact is INCORRECT (the edit
failed).
- A prediction that is irrelevant, evasive, or empty is INCORRECT.
- Length, style, and politeness are irrelevant.

Respond with EXACTLY one word: CORRECT or INCORRECT. No other text. No
punctuation. No explanation.
```

Figure 7: CounterFact judge prompt (`JUDGE_PROMPT_COUNTERFACT`, version v1.0). Applied to the **CounterFact** $\mathcal{D}_{\text{conflict}}$ split, where the gold answer is the injected counterfactual target rather than the true real-world fact. Implemented in `src/eval/external_judge.py`.

AIT QM Conflict Pair Generation Prompts

The semi-synthetic AIT QM conflict pairs (Section 5) are built with two sequential calls to `Qwen3.5-35B-A3B-Q4_K_M.gguf` (Qwen 3.5 MoE, 35B total / 3B active parameters, quantised to 4-bit `Q4_K_M`) per candidate: a generation call that produces the contradicting earlier-revision answer, followed by a verification call that confirms the two answers genuinely contradict. Placeholder fields (shown in `<<ALLCAPS>>`) are filled per candidate at inference time. The prompt version logged in every output record is `qm-conflict-v1`; implementation is in `scripts/build_qm_conflict_pairs.py`.

```
You are helping construct a knowledge-editing benchmark for an enterprise
Quality Management (QM) system.

You are given a CURRENT, factually correct question-answer pair extracted
from a company's QM documentation. Imagine the PREVIOUS revision of that
document (the version in force before the most recent update) and write
what the answer USED TO say.

Strict requirements:
1. Change exactly ONE concrete, checkable attribute: a date, a number, a
quantity, a threshold/limit, a responsible role or department, a location,
a standard/norm identifier, or a named requirement.
2. The OLD answer must DIRECTLY CONTRADICT the current answer on that
single attribute: a reader comparing the two must see a clear factual
disagreement.
3. Everything else stays the same: same meaning, same structure, same
length, same terminology, same formatting. Only the one attribute differs.
4. The old value must be PLAUSIBLE for an earlier revision of a real QM
document (a realistic earlier date, a realistic earlier threshold, a
real-sounding role). Never absurd or obviously fake.
5. Write the OLD answer in <<LANGUAGE>>, the same language as the
current answer.
6. Do NOT change a value, name, or entity that already appears in the
QUESTION below. The attribute you change must occur only in the answer;
otherwise the question itself gives away the current answer.

Question:
<<QUESTION>>

Current (correct) answer:
<<ANSWER>>

Respond with ONLY a single JSON object and nothing else:
{"changed_attribute": "<short description of the attribute you changed>",
"new_value": "<the value stated in the current answer>", "old_value":
"<the contradicting earlier value>", "old_answer": "<the full
earlier-revision answer>"}
```

Figure 8: QM conflict pair generation prompt (`QM_EDIT_PROMPT`, version `qm-conflict-v1`). Placeholders `<<LANGUAGE>>`, `<<QUESTION>>`, and `<<ANSWER>>` are filled per candidate at inference time from the AIT QM source dataset. Greedy decoding (`do_sample=False`, `max_new_tokens=768`). The JSON response is extracted with a regex-tolerant parser; records with malformed JSON or an empty `old_answer` field are discarded.

```
You are checking a knowledge-editing benchmark pair.

Two answers to the SAME question are given: a CURRENT answer and an
EARLIER-REVISION answer. They are supposed to DIRECTLY CONTRADICT each
other on exactly one factual point.

Question:
<<QUESTION>>

Current answer:
<<ANSWER_NEW>>

Earlier-revision answer:
<<ANSWER_OLD>>

Decide: do the two answers state DIFFERENT, MUTUALLY EXCLUSIVE values for
the same factual point, such that they cannot both be true?

Respond with EXACTLY one word: CONTRADICT or COMPATIBLE. No other text.
```

Figure 9: QM conflict pair verification prompt (`QM_VERIFY_PROMPT`, version `qm-conflict-v1`). Placeholders filled per candidate from the generation step. Greedy decoding (`do_sample=False`, `max_new_tokens=8`). The verdict is parsed with boundary-match regex; off-spec responses and `COMPATIBLE` verdicts cause the candidate to be discarded without recording.

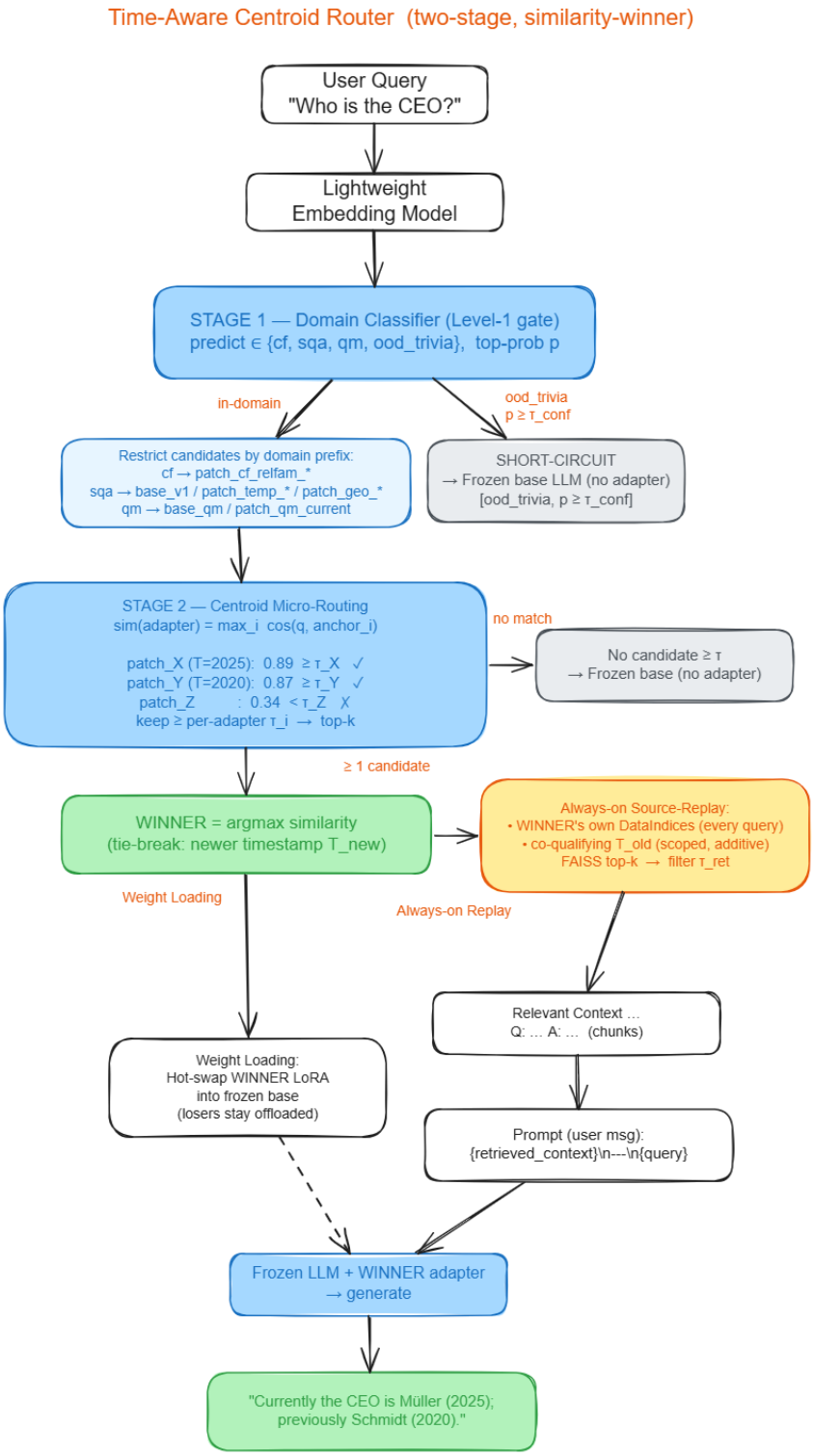

Figure 10: Time-Aware Centroid Router with Source-Replay (full-size companion to Section 4.3). The Stage-1 domain gate short-circuits out-of-domain queries to the frozen base; in-domain queries undergo centroid micro-routing, the highest-similarity expert is hot-swapped (timestamp breaking near-ties toward the newer authority), and always-on Source-Replay injects the winner's training chunks into the context window.

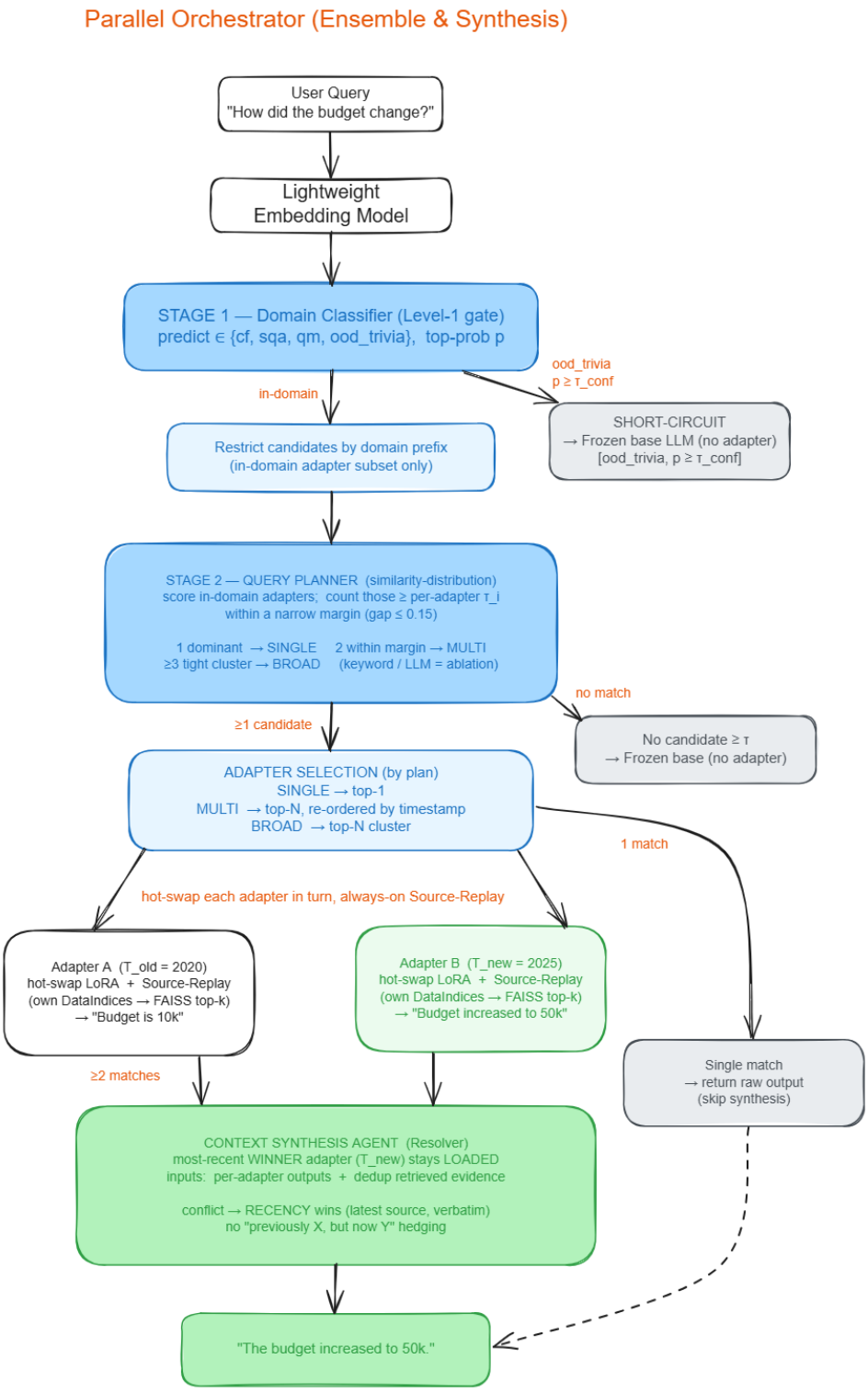

Figure 11: Parallel-Orchestrator architecture (full-size companion to Section 4.3). The similarity-distribution planner selects one or more in-domain adapters; each runs in turn with its own always-on Source-Replay context, and the Resolver (with the most-recent winning adapter kept loaded) fuses the per-adapter outputs and retrieved evidence, resolving conflicts by recency.

Erklärung zum Einsatz generativer KI-Werkzeuge

Im Rahmen dieser Masterarbeit wurden generative KI-Werkzeuge eingesetzt. Für die Implementierung des PnR-Frameworks und der zugehörigen Evaluierungsskripte wurde *Claude* (Anthropic) zur Unterstützung bei der Code-Erstellung und dem Debugging verwendet. Für das Verfassen, Strukturieren und Überarbeiten von Textabschnitten dieser Arbeit wurden *Claude* (Anthropic) und *Gemini* (Google) genutzt.
Alle KI-generierten Inhalte (sowohl Code als auch Text) wurden von mir kritisch geprüft, inhaltlich bewertet und gegebenenfalls überarbeitet. Die fachliche und inhaltliche Verantwortung für die gesamte Arbeit verbleibt bei mir.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Mainz, den June 18, 2026