DOCPATCH: CONTINUAL KNOWLEDGE INTERNALIZATION WITH CONFLICT-AWARE LORA MEMORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) can acquire external knowledge either by repeatedly conditioning on retrieved context or by updating model parameters. However, retrieval-based methods incur recurring inference-time context costs, while parameter-based updates require additional optimization for new knowledge and may interfere with previously acquired information. We propose DOCPATCH, a continual parametric knowledge framework that internalizes documents as modular and reusable LoRA memories while keeping the base language model frozen. Given a collection of documents, DOCPATCH uses a pretrained Doc-to-LoRA hypernetwork to convert short document chunks into knowledge LoRAs, which are generated offline and stored in a persistent knowledge bank. At inference time, a query-dependent router retrieves relevant LoRAs using their cached parametric representations, allowing the model to answer from a question-only prompt without reintroducing the original document text into the context window. To support evolving knowledge, DOCPATCH preserves LoRAs from different document versions and associates them with source, version, and conflict metadata, enabling newer conflicting knowledge to override outdated information while retaining earlier non-conflicting knowledge. We further introduce a lightweight global reasoning adapter that teaches the frozen LLM to follow activated knowledge LoRAs over conflicting model priors and to combine complementary LoRAs for multi-step reasoning. By separating knowledge storage, retrieval, conflict resolution, and reasoning, DOCPATCH provides a modular framework for continually extending and updating the parametric knowledge of frozen language models without repeatedly retraining the base model.

1 INTRODUCTION

Large language models (LLMs) encode substantial knowledge in their parameters, but efficiently incorporating new and evolving information remains challenging. Retrieval-augmented generation (RAG) addresses this problem by retrieving relevant documents and inserting them into the context window (Lewis et al., 2020). While flexible, RAG requires repeatedly processing retrieved text at inference time. Model editing instead incorporates new knowledge through parameter updates (De Cao et al., 2021; Mitchell et al., 2022a; Meng et al., 2022; 2023), but each update typically requires additional optimization, and repeated edits may introduce interference, locality issues, or degradation of previously acquired knowledge. Recent work on Doc-to-LoRA (D2L) provides a different direction by converting textual contexts directly into LoRA adapters (Charakorn et al., 2026). This enables document knowledge to be internalized parametrically while keeping the base language model frozen. However, maintaining a large and continuously evolving collection of such parametric memories introduces additional challenges: the system must identify which memories are relevant to a query, combine complementary information across multiple memories, and resolve conflicts between newly introduced and previously stored knowledge.

We propose DOCPATCH, a framework for *continual parametric knowledge patching*. DOCPATCH converts document chunks into knowledge LoRAs using a pretrained Doc-to-LoRA hypernetwork and stores them in a persistent knowledge bank. At inference time, a query-dependent router retrieves a small set of relevant knowledge LoRAs using their cached representations. The selected adapters are then composed and applied to a frozen base LLM, allowing the model to answer from a question-only prompt without inserting the original documents into its context window.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

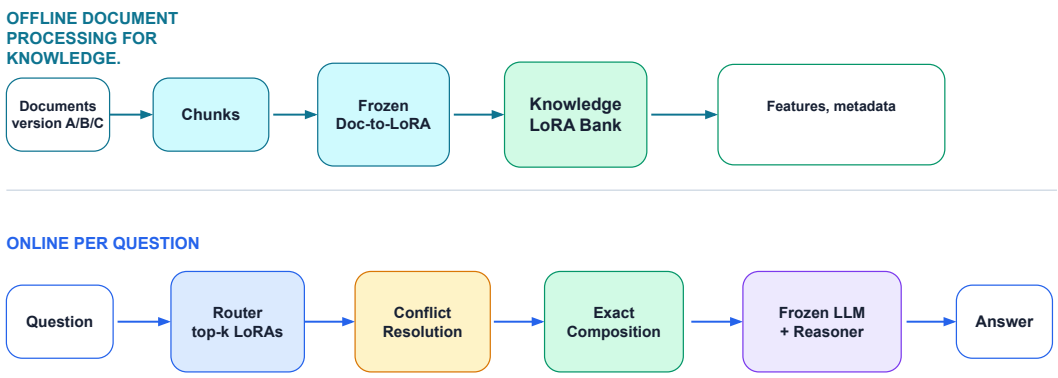


Figure 1: Overview of the proposed framework.

A key property of DocPatch is that new knowledge is added rather than overwriting existing knowledge. When an updated document arrives, DocPatch generates new knowledge LoRAs while retaining previous versions. Version and conflict metadata are used to suppress outdated adapters when their knowledge conflicts with newer information, while earlier non-conflicting knowledge remains available. We further introduce a lightweight global reasoning adapter that teaches the frozen LLM to follow activated knowledge LoRAs over conflicting model priors and to combine complementary knowledge from multiple adapters.

By separating knowledge storage, retrieval, conflict resolution, and reasoning, DocPatch provides a modular approach to continually extending the knowledge of a frozen language model. Our main contributions are:

- We introduce a parametric knowledge framework that converts document chunks into reusable LoRA memories and retrieves them without placing the original documents in the inference context.
- We formulate continual knowledge updates as additive, versioned LoRA patches, enabling newer conflicting knowledge to override outdated information while preserving non-conflicting earlier knowledge.
- We introduce a global reasoning adapter for reliably using and combining retrieved knowledge LoRAs while keeping both the knowledge modules and the base LLM frozen.

2 Related Work

Retrieval and Parametric Context Internalization. Retrieval-augmented generation (RAG) incorporates external knowledge by retrieving relevant passages and placing them in the model context (Lewis et al., 2020). This allows the external knowledge source to be updated without modifying model parameters, but retrieved documents must be processed again for each query. An alternative direction is to internalize context into model parameters. Doc-to-LoRA (D2L) (Charakorn et al., 2026) uses a hypernetwork to directly generate LoRA adapters from textual contexts, allowing subsequent queries to access the internalized information without repeatedly consuming the original context. DocPatch builds on this idea but targets a different setting: rather than internalizing a single context for subsequent queries, we maintain a growing bank of document-derived LoRAs and study how to retrieve, compose, and update these parametric memories as the underlying documents evolve.

Knowledge Editing and Continual Updates. A large body of work studies how to modify factual knowledge stored in language models. ROME (Meng et al., 2022) performs targeted rank-one updates to model weights, while MEMIT (Meng et al., 2023) extends this approach to editing many factual associations at once. MEND (Mitchell et al., 2022a) learns an auxiliary network that produces efficient model updates, while SERAC (Mitchell et al., 2022b) instead stores edits in an explicit memory and retrieves them when relevant. More recent work has considered lifelong editing,

where large sequences of edits must be incorporated without disrupting previous knowledge. WISE (Wang et al., 2024), for example, separates pretrained and edited knowledge into different parametric memories and introduces routing mechanisms for selecting between them. These methods primarily formulate knowledge updates as edits to facts or model behavior. In contrast, DOCPATCH treats document chunks as persistent parametric knowledge units and retains different document versions explicitly, allowing new knowledge to be added without overwriting previous LoRA memories. Conflicts between versions can therefore be resolved explicitly while preserving earlier non-conflicting knowledge.

Modular Adapters and Dynamic Routing. Parameter-efficient adaptation provides a natural mechanism for storing knowledge in modular form. LoRA (Hu et al., 2022) represents model updates using low-rank parameter modifications while leaving the base model frozen. AdapterFusion (Pfeiffer et al., 2021) demonstrates that independently learned adapters can be combined without destructively modifying their original parameters. Related model-editing approaches also use dynamically selected parameter modules. MELO (Yu et al., 2024), for instance, organizes edits into LoRA blocks and dynamically activates them using a retrieval index. Such methods motivate the use of modular parameter updates and query-dependent activation. DOCPATCH differs in that its LoRAs are generated directly from document chunks rather than learned from individual editing examples, and routing operates over a persistent collection of document-derived parametric memories. Moreover, DOCPATCH explicitly considers multi-LoRA composition, document versioning, and conflicts between newly added knowledge, previously stored knowledge, and the pretrained model prior. Overall, DOCPATCH lies at the intersection of retrieval, model editing, and parameter-efficient adaptation. It combines the updateability of an external knowledge store with a parametric representation of document knowledge, while separating knowledge storage, query-dependent retrieval, conflict resolution, and knowledge utilization.

3 METHOD

3.1 OVERVIEW

We propose DOCPATCH, a continual parametric knowledge framework that represents an evolving document collection as a set of modular LoRA-based memories. The framework consists of four main components: (1) an offline knowledge bank constructed from document chunks, (2) a query-dependent router that retrieves relevant knowledge LoRAs, (3) a conflict-resolution mechanism that determines how conflicting knowledge should be used, and (4) a global reasoning adapter that enables a frozen language model to reliably use and combine the retrieved knowledge. Given a document collection, we first split each document into short chunks and convert each chunk into a knowledge LoRA using a pretrained Doc-to-LoRA hypernetwork (Charakorn et al., 2026). These knowledge LoRAs are generated only once and cached together with compact representations and metadata. At inference time, a question is mapped to a query representation, and the router retrieves a small set of relevant LoRAs from the knowledge bank. The retrieved LoRAs are then passed through conflict resolution and composed into a query-specific knowledge update. The final answer is generated by the frozen base LLM augmented with the composed knowledge LoRA and a trainable global reasoning adapter. The original document text is not included in the inference prompt, making the inference fast and efficient.

3.2 OFFLINE KNOWLEDGE BANK

Let a document collection be divided into chunks $c_i{}_{i=1}^{N}$. A frozen Doc-to-LoRA hypernetwork H maps each chunk c_i to a knowledge LoRA

$$\mathcal{L}_i = H(c_i) = \left\{ \left(A_i^{(\ell)}, B_i^{(\ell)} \right) \right\}_{\ell \in \mathcal{T}}, \tag{1}$$

where $\mathcal{T}$ denotes the set of adapted linear layers. $A_i^{(\ell)} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B_i^{(\ell)} \in \mathbb{R}^{d_{\text{out}} \times r}$ denote the rank-r LoRA matrices generated for layer ℓ.

For each layer ℓ, the corresponding LoRA update is $\Delta W_i^{(\ell)} = B_i^{(\ell)} A_i^{(\ell)}$. For retrieval, we additionally construct a compact representation z_i (See A.3) from the generated LoRA parameters. Specifically, we first extract a fixed-dimensional feature vector summarizing the LoRA factors across the

adapted layers, and then map this feature vector into the router representation space using a learned LoRA-feature. The resulting representation z_i is cached together with the corresponding knowledge LoRA and is used for query-dependent retrieval. Each knowledge LoRA is stored together with metadata μ_i, including its source, chunk identifier, document version, and conflict information. The resulting knowledge bank is: $\mathcal{M} = \{(\mathcal{L}_i, \mathbf{z}_i, \mu_i)\}_{i=1}^{N}$. When a new document or document version is introduced, its chunks are converted into additional knowledge LoRAs and appended to the knowledge bank. Existing LoRAs are retained rather than overwritten. In this way, DOCPATCH maintains explicit parametric snapshots of knowledge across document versions.

3.3 QUERY-DEPENDENT KNOWLEDGE ROUTING

Given a question q, a learned query encoder E that leverage the base LLM and a learnable query projector P_q (See A.2) produces a query representation $e_q = E(q)$. The router R compares the query representation with each candidate LoRA representation z_i, which is calculated by a learned LoRA projector P_{LoRA} (See A.3) and produces a relevance score

$$s_i = R(e_q, z_i). \tag{2}$$

For a candidate set $\mathcal{C}_q$, these scores define a probability distribution over candidate knowledge LoRAs:

$$p_i = \frac{\exp(s_i)}{\sum_{j \in \mathcal{C}_q} \exp(s_j)}. \tag{3}$$

A question may have multiple supporting chunks. Let $\mathcal{G}_q \subseteq \mathcal{C}_q$ denote the set of ground-truth knowledge LoRAs. We use a normalized multi-hot target,

$$y_i = \begin{cases} \frac{1}{|\mathcal{G}_q|}, & i \in \mathcal{G}_q, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

and optimize the chunk-level classification objective

$$\mathcal{L}_{\text{chunk}} = -\sum_{i \in \mathcal{C}_q} y_i \log p_i. \tag{5}$$

This objective encourages the router to assign high probability to all supporting knowledge LoRAs rather than only a single positive memory. We additionally supervise source-level discrimination. Let $c(i)$ denote the source of chunk i, and let d^+ denote the ground-truth source. The probability assigned to source d is

$$p_d = \sum_{\substack{i \in \mathcal{C}_q \\ c(i)=d}} p_i, \tag{6}$$

and the source-level objective is

$$\mathcal{L}_{\text{source}} = -\log p_{d^+}. \tag{7}$$

The final routing objective is

$$\mathcal{L}_{\text{router}} = \mathcal{L}_{\text{chunk}} + \lambda_{\text{source}} \mathcal{L}_{\text{source}}. \tag{8}$$

During the optimization of $\mathcal{L}_{\text{router}}$, the router R, the query projector P_q and the LoRA projector P_{LoRA} are trainable. At inference time, the top-k knowledge LoRAs ranked by p_i are retained for subsequent conflict resolution and composition.

3.4 CONFLICT-AWARE KNOWLEDGE RESOLUTION

Retrieved knowledge LoRAs may encode conflicting information across document versions. In our version-aware setting, conflict relationships are provided during knowledge-bank construction: incompatible alternatives of the same underlying information share a conflict-group identifier, while version metadata specifies their precedence.

Conflict resolution is applied only to the retrieved set $\mathcal{S}_q$; all LoRAs remain stored in the knowledge bank. For each $i \in \mathcal{S}_q$, we define

$$m_i = \begin{cases} 0, & \text{if a higher-precedence retrieved LoRA belongs to the same conflict group,} \\ 1, & \text{otherwise.} \end{cases} \tag{9}$$

Thus, only the highest-precedence retrieved member of each conflict group contributes to the current composition, while compatible memories from different groups may contribute jointly.

Let $\alpha_i = \mathrm{Softmax}(s_i)$ denote the normalized routing weight derived from the router score s_i (see Eq. 2). We define the final composition weight as

$$g_i = \frac{\alpha_i m_i}{\sum_{j \in \mathcal{S}_q} \alpha_j m_j}. \tag{10}$$

The router determines relevance, whereas conflict resolution determines applicability under version precedence. This allows DOCPATCH to resolve conflicts locally while composing complementary knowledge globally. Our current formulation assumes conflict groups and version precedence are available from the document-update process; automatic conflict identification in arbitrary unversioned collections is outside the scope of this work.

3.5 MULTI-LORA KNOWLEDGE COMPOSITION

After conflict resolution, the selected LoRAs must be combined into a single query-specific parameter update. At layer ℓ, the desired composed knowledge update is

$$\Delta W^{(\ell)}_{\text{know}} = \sum_{i \in \mathcal{S}_q} g_i B^{(\ell)}_i A^{(\ell)}_i. \tag{11}$$

For all LoRAs with nonzero composition weight, we construct

$$A^{(\ell)}_q = \mathrm{concat}_{\text{row}} \left(g_1 A^{(\ell)}_1, \ldots, g_k A^{(\ell)}_k \right), B^{(\ell)}_q = \mathrm{concat}_{\text{col}} \left(B^{(\ell)}_1, \ldots, B^{(\ell)}_k \right),$$

where the same ordering of LoRAs is used in both concatenations. This gives

$$B^{(\ell)}_q A^{(\ell)}_q = \sum_{i=1}^{k} g_i B^{(\ell)}_i A^{(\ell)}_i, \tag{12}$$

which exactly recovers the desired composed update. This composition mechanism allows multiple complementary knowledge LoRAs to contribute simultaneously while respecting the decisions made by the conflict-resolution module.

3.6 GLOBAL REASONING ADAPTER AND REASONING TRAINING

Retrieving the correct knowledge LoRAs does not necessarily guarantee that the frozen LLM will use them appropriately. Activated knowledge may conflict with pretrained model priors, and some questions require integrating complementary information distributed across multiple LoRAs. We therefore introduce a global reasoning LoRA shared across all documents and queries. Unlike the document-specific knowledge LoRAs, this adapter is not intended to store factual knowledge; instead, it learns general behaviors for using the parametric knowledge supplied by the activated memories.

For an adapted linear layer ℓ, let $\mathbf{h} \in \mathbb{R}^{d_{\text{in}}}$ denote the input activation. The layer computation is

$$\mathbf{h}' = W^{(\ell)}_0 \mathbf{h} + \Delta W^{(\ell)}_{\text{know}} \mathbf{h} + \Delta W^{(\ell)}_{\text{reason}} \mathbf{h}, \tag{13}$$

where $W^{(\ell)}_0$ is the frozen base-model weight, $\Delta W^{(\ell)}_{\text{know}}$ is the query-specific update obtained from the composed knowledge LoRAs, and $\Delta W^{(\ell)}_{\text{reason}}$ is the update produced by the global reasoning adapter. Equivalently, the adapted layer applies

$$W^{(\ell)}_0 + \Delta W^{(\ell)}_{\text{know}} + \Delta W^{(\ell)}_{\text{reason}}. \tag{14}$$

The knowledge update therefore provides query-specific factual information, while the reasoning adapter provides a shared mechanism for utilizing that information.

We train the global reasoning adapter using answer-level supervision from the training sets of Constitution QA and HotpotQA. During this stage, the base LLM, document-specific knowledge LoRAs, query encoder, and trained router remain frozen. Given a question q, activated knowledge LoRAs $\mathcal{K}$, and target answer $y = (y_1, \ldots, y_T)$, we minimize the autoregressive answer loss

$$\mathcal{L}_{\text{answer}} = -\sum_{t=1}^{T} \log p_{\theta_{\text{reason}}} \left(y_t \mid q, y_{<t}; \mathcal{K} \right), \tag{15}$$

where θ_{reason} contains only the parameters of the global reasoning LoRA. Thus, gradients from $\mathcal{L}_{\text{answer}}$ update only the reasoning adapter, while the base model and document-specific knowledge remain unchanged. This separation encourages the adapter to learn how to follow activated knowledge when it differs from the model prior, combine complementary memories, and avoid relying on irrelevant activations.

Training follows a staged curriculum. We first activate ground-truth knowledge LoRAs, allowing the reasoning adapter to learn how to use document-derived knowledge independently of retrieval errors. The training set also includes counterfactual pairs in which the same question is paired with different activated LoRAs and corresponding target answers. These examples encourage the model to condition on the active parametric memory rather than memorizing a fixed question–answer mapping. We then train with multiple complementary ground-truth LoRAs to improve multi-memory reasoning. Finally, we introduce examples with versioned conflicts after metadata-based conflict resolution, teaching the model to answer according to the resulting conflict-resolved knowledge state. At inference time, the complete DocPatch pipeline is

$$\begin{aligned} q \rightarrow \text{knowledge routing} \rightarrow \text{conflict resolution} \rightarrow \text{LoRA composition} \\ \rightarrow \text{frozen LLM + reasoning adapter} \rightarrow \text{answer}. \end{aligned}$$

By keeping knowledge storage, retrieval, conflict resolution, and reasoning as separate components, DocPatch can continually incorporate new document knowledge while preserving previously stored parametric memories and selectively resolving how different knowledge sources should contribute to each query.

4 Experiments

We evaluate DocPatch along four dimensions: (1) document question answering with retrieved parametric knowledge, (2) multi-LoRA reasoning over complementary memories, (3) continual knowledge updating under versioned conflicts, and (4) inference-time efficiency. We additionally evaluate the knowledge router and conduct ablations on the reasoning adapter, LoRA selection and weighting, and counterfactual training.

4.1 Experimental Setup

We evaluate end-to-end question answering, multi-hop reasoning, and knowledge-LoRA routing on **HotpotQA** (Yang et al., 2018), where supporting passages are independently converted into knowledge LoRAs. The router is trained on our Constitution QA corpus, containing 4,908 chunks from seven constitutions, 10,182 training questions, and 3,563 development questions with source- and chunk-level annotations, then frozen and evaluated on HotpotQA. The global reasoning adapter is trained separately on the training sets of Constitution QA and HotpotQA. We further construct document histories with successive revisions to evaluate whether the model follows activated LoRA knowledge while preserving valid and unrelated behavior. Conflicting LoRAs are annotated with version and conflict metadata. We use **Qwen3-4B** (Yang et al., 2025) as the frozen backbone with its pretrained Doc-to-LoRA checkpoint. The router retrieves the top-5 LoRAs. The reasoning adapter is trained for 40K steps using AdamW with a learning rate of 5×10^{-5}. Evaluation experiments are conducted on an NVIDIA H100 GPU and averaged over five random seeds. For question answering, we report EM and token-level F1; for routing, Top-5 recall, F1, and full-coverage accuracy.

4.2 MAIN DOCUMENT QUESTION ANSWERING

We first evaluate whether document knowledge can be accessed from the parametric knowledge bank without providing the original document text in the inference context. We compare DOCPATCH with **Frozen LLM**, **RAG** (Lewis et al., 2020), and **Doc-to-LoRA** (Charakorn et al., 2026), including Top-1 and ground-truth variants. For the former, the selected LoRA is the the highest-ranking LoRA selected by the router, while the latter uses LoRA that represents the ground-truth knowledge. The ground-truth variants therefore provide an upper bound on the performance of DocPatch. As shown in Table 1, DOCPATCH achieves the best performance among the non-ground-truth methods, reaching 69.25 EM and 75.78 F1. Compared with RAG, the strongest non-ground-truth baseline, DOCPATCH improves EM by 2.47 points and F1 by 5.46 points. Compared with D2L (Top-1), the improvements are 7.22 EM and 11.02 F1. The global reasoning adapter also provides a consistent improvement over the same routed knowledge LoRAs without reasoning training, increasing EM from 65.49 to 69.25 and F1 from 71.65 to 75.78. This indicates that retrieving relevant parametric knowledge alone is insufficient; the frozen language model must also learn how to reliably use the activated knowledge.

Method	**EM ↑**	**F1 ↑**
Frozen LLM	48.34	52.12
RAG (Lewis et al., 2020)	66.78	70.32
D2L (Top-1) (Charakorn et al., 2026)	62.03	64.76
D2L (Ground Truth)	71.54	77.38
DOCPATCH w/o Reasoning		
Adapter	65.49	71.65
DOCPATCH	**69.25**	**75.78**

Table 1: Main question answering results on HotpotQA.

4.3 KNOWLEDGE-LORA ROUTING

We evaluate the router independently from answer generation. The router operates only on the question representation and cached LoRA-derived features, without accessing the original document chunks. Since a query may require multiple supporting chunks, it can correspond to multiple ground-truth knowledge LoRAs. For each query, let $\mathcal{G}$ denote the set of ground-truth knowledge LoRAs and $\mathcal{P}$ the top-5 LoRAs returned by the router. We define

$$\text{Recall} = \frac{|\mathcal{G} \cap \mathcal{P}|}{|\mathcal{G}|}, \ \text{Accuracy} = \mathbf{1}[\mathcal{G} \subseteq \mathcal{P}]\,. \tag{16}$$

F1 is the harmonic mean of precision and recall. Recall measures the fraction of required knowledge LoRAs retrieved in the Top-5 set, while accuracy is a stricter query-level metric that is counted as correct only when all required LoRAs are retrieved. All metrics are averaged over the evaluation queries.

Method	**Top-5 F1 ↑**	**Top-5 Recall ↑**	**Accuracy ↑**
Random	19.03	30.65	23.48
Similarity Retrieval	52.46	63.78	58.45
DOCPATCH Router	**76.41**	**89.67**	**82.44**

Table 2: Knowledge-LoRA routing performance on HotpotQA.

As shown in Table 2, DOCPATCH achieves a Top-5 F1 of 76.41 and a Top-5 recall of 89.67, substantially outperforming similarity retrieval at 52.46 and 63.78, respectively. It further achieves an accuracy of 82.44, indicating that the router retrieves all required knowledge LoRAs for a large fraction of queries using only cached LoRA-derived representations.

4.4 MULTI-LORA AND MULTI-HOP REASONING

We evaluate the ability of DOCPATCH to combine complementary knowledge distributed across multiple LoRAs. For each HotpotQA question, supporting passages are independently converted into knowledge LoRAs. Table 3 compares the full model with retrieval and direct-composition baselines.

Let $\mathcal{S}_q$ denote the retrieved top-k LoRAs. In *Uniform Composition*, all retrieved LoRAs contribute equally:

$$\Delta W_{\text{uniform}} = \frac{1}{|\mathcal{S}_q|} \sum_{i \in \mathcal{S}_q} B_i A_i. \tag{17}$$

In *Router-Weighted Composition*, the same retrieved LoRAs are weighted according to their router probabilities:

$$\Delta W_{\text{router}} = \sum_{i \in \mathcal{S}_q} \frac{p_i}{\sum_{j \in \mathcal{S}_q} p_j} B_i A_i. \tag{18}$$

Both baselines use the exact multi-LoRA composition described in Section 3.5, but do not use the global reasoning adapter. The *Ground-truth Knowledge LoRA* setting bypasses retrieval and directly provides the ground-truth supporting adapters, isolating performance under correct knowledge selection. As shown in Table 3, router-weighted composition improves over uniform composition from 58.14 to 65.49 EM and from 62.06 to 71.65 F1, showing the benefit of query-dependent weighting. Adding the global reasoning adapter further improves performance to 69.25 EM and 75.78 F1. The ground-truth setting reaches 71.54 EM and 77.38 F1, indicating that part of the remaining gap is attributable to imperfect knowledge selection.

4.5 Continual Knowledge Updating and Conflict Resolution

A central goal of DocPatch is to incorporate evolving knowledge without overwriting previously stored parametric memories. We therefore evaluate sequences of successive document revisions in which later versions may preserve, extend, or conflict with earlier information, while all previously generated knowledge LoRAs remain in the bank. We compare against ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), SERAC (Mitchell et al., 2022b), MELO (Yu et al., 2024), and WISE (Wang et al., 2024) (see Section 2). Results are shown in Table 4: *Unchanged* represents cases where new LoRA variants did not change the relevant knowledge pertinent to a question; *Updated* are test cases where new LoRA variants contradict the pretrained model; *Old-only* are cases where a relevant piece of knowledge was only found in an older LoRA variant; *Override* happens when new LoRA variants contradict their older counterparts; *Unrelated* happens when none of the LoRAs carry knowledge relevant to the question.

Method	EM ↑	F1 ↑
Frozen LLM	48.34	52.12
RAG	66.78	70.32
Ground-truth Knowledge LoRA	71.54	77.38
Top-5, Uniform Composition	58.14	62.06
Top-5, Router-Weighted	65.49	71.65
DocPatch	**69.25**	**75.78**

Table 3: Multi-hop question answering results on HotpotQA. Ground-truth Knowledge LoRA uses the ground-truth supporting adapters.

DocPatch performs strongest in the settings that require the model to actively follow parametric knowledge, achieving the highest updated-knowledge score (81.85) and conflict override rate (84.98). It also remains competitive on unchanged (82.94), old-only (86.07), and unrelated (80.22) knowledge. These results indicate that DocPatch is particularly effective when activated LoRA knowledge must override either the pretrained model prior or conflicting older memories. Although the strongest baselines achieve slightly higher performance on some of the preservation-oriented settings, we note that DocPatch is a significantly more efficient approach comparatively. Unlike ROME and MEMIT, which compute and apply new parameter edits, or MELO and WISE, which update parametric memories for incoming edits (Meng et al., 2022; 2023; Yu et al., 2024; Wang et al., 2024), DocPatch converts each new document chunk into a reusable knowledge LoRA through a single forward pass of the frozen Doc-to-LoRA hypernetwork, requiring no additional gradient-based training. Compared with SERAC (Mitchell et al., 2022b), which stores edits explicitly and retrieves them for a separate counterfactual model, DocPatch represents document knowledge as parametric memories that can be retrieved, composed, and resolved under version conflicts within the frozen LLM, without repeatedly exposing the original source text during inference.

Method	Unchanged ↑	Updated ↑	Old-Only ↑	Override ↑	Unrelated ↑
RAG	82.04	75.12	**88.31**	–	79.03
ROME (Meng et al., 2022)	79.43	81.56	84.33	82.91	76.56
MEMIT (Meng et al., 2023)	78.62	81.47	82.68	82.04	77.39
SERAC (Mitchell et al., 2022b)	74.05	79.35	77.62	79.49	70.03
MELO (Yu et al., 2024)	**83.11**	78.28	84.94	76.73	80.35
WISE (Wang et al., 2024)	81.32	79.40	81.93	83.65	**81.29**
DOCPATCH	82.94	**81.85**	86.07	**84.98**	80.22

Table 4: Continual knowledge updating under sequential document versions.

4.6 ABLATION STUDY

We ablate the major components of DOCPATCH to determine their individual contributions (Table 5). Removing the reasoning adapter reduces EM from 69.25 to 65.49 and F1 from 75.78 to 71.65, confirming the importance of explicitly training the frozen LLM to use activated parametric knowledge. Restricting the system to a single LoRA or using uniform composition further degrades performance, demonstrating the benefit of query-dependent multi-LoRA composition. Counterfactual training has the largest impact among the evaluated ablations. Removing counterfactual examples reduces EM from 69.25 to 53.46, F1 from 75.78 to 61.03, and conflict override from 84.98 to 62.44. This substantial degradation indicates that counterfactual supervision is critical for teaching the reasoning adapter to follow the currently activated parametric knowledge rather than reverting to the pretrained model prior.

4.7 EFFICIENCY

We compare inference-time cost in Table 6. Since document chunks are converted into knowledge LoRAs offline, DOCPATCH answers from a question-only prompt without repeatedly inserting retrieved source text into the context. RAG processes approximately 4000 input tokens per query, compared with about 50 for Doc-to-LoRA and DOCPATCH. Consequently, DOCPATCH reduces latency from 3048 ms to 451 ms and peak GPU memory from 27.35 GB to 26.14 GB. Compared with single-adapter Doc-to-LoRA, the additional overhead from routing, multi-LoRA composition, and the global reasoning adapter remains modest.

Variant	Multi-Hop EM ↑	Multi-Hop F1 ↑	Multi-Hop Override ↑
DOCPATCH	**69.25**	**75.78**	**84.98**
w/o Reasoning Adapter	65.49	71.65	83.13
Top-1 LoRA Only	62.13	68.22	67.45
Uniform LoRA Weights	58.14	62.06	69.76
w/o Counterfactual Training	53.46	61.03	62.44

Table 5: Ablation study of components in DOCPATCH.

5 CONCLUSION

We introduced DOCPATCH, a continual parametric knowledge framework that represents document knowledge as modular LoRA memories while keeping the base language model frozen. DOCPATCH separates knowledge storage, query-dependent retrieval, conflict resolution, and reasoning, enabling new document versions to be incorporated without overwriting previously stored knowledge. Retrieved LoRAs are composed into query-specific parameter updates, while a shared reasoning adapter helps the model follow activated knowledge, combine complementary memories, and override conflicting priors. Experiments show strong performance on question answering, multi-LoRA reasoning, and continual knowledge updating, while preserving valid and unrelated knowledge. These results support modular parametric memories as an alternative to repeated context injection or direct base-model editing. Future work will explore more flexible conflict resolution, finer-grained activation, and larger knowledge banks.

Method	Input Tokens ↓	Latency (ms) ↓	Peak GPU Memory (GB) ↓
RAG	4000	3048	27.35
Doc-to-LoRA	50	342	24.86
DOCPATCH	**50**	451	26.14

Table 6: Inference efficiency on Qwen3-4B. Offline knowledge-LoRA generation is excluded from per-query latency.

AI USE STATEMENT

In this work, we used generative AI tools to check grammar and improve the clarity of the writing. We also used AI to generate our Constitution QA dataset, by prompting LLMs to generate verifiable questions and answers towards constitutions (with amendments). We futher verify the correctness of the QA through human review.

ETHICS STATEMENT

DOCPATCH studies the internalization, retrieval, and updating of document knowledge in frozen language models. As with retrieval and model-editing systems, the framework may internalize incorrect, biased, or harmful information if such content is present in the source documents. In addition, incorrect version or conflict metadata may cause outdated or inappropriate knowledge to be activated. Our current formulation therefore assumes that document provenance, version relationships, and conflict annotations are available and reliable, and we explicitly describe these limitations rather than treating the system as an automatic verifier of factual correctness. The modular knowledge bank preserves document-specific LoRAs rather than modifying the base model directly, which provides explicit separation between stored knowledge and the underlying model, but also introduces persistent storage and provenance considerations. Future deployment should therefore consider source reliability, access control, privacy, licensing, and mechanisms for auditing or disabling inappropriate knowledge memories.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce DOCPATCH throughout the main paper and appendix. The model architecture, knowledge routing objective, conflict-resolution rule, multi-LoRA composition, and reasoning-adapter training procedure are described in Sections 3.2–3.6, while implementation details, training settings, datasets, evaluation metrics, and baseline configurations are summarized in Section 4.1. The appendix further provides the derivation of exact multi-LoRA composition, the LoRA-to-feature mapping, Constitution QA construction and router supervision, conflict-handling assumptions, and details of counterfactual reasoning training.

REFERENCES

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-loRA: Learning to instantly internalize contexts. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=iW1oBBO72S.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6491–6506. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.522.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.07229.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022a. URL https://arxiv.org/abs/2110.11309.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15817–15831. PMLR, 2022b. URL https://proceedings.mlr.press/v162/mitchell22a.html.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.39.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.14768.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. MELO: Enhancing model editing with neuron-indexed dynamic LoRA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19449–19457, 2024. doi: 10.1609/aaai.v38i17.29916.

A Appendix

A.1 Exact Multi-LoRA Composition

This section provides a detailed derivation of the exact multi-LoRA composition used in DocPatch. After routing and conflict resolution, let $\mathcal{S}_q^+ = \{i \in \mathcal{S}_q : g_i > 0\}$ denote the set of active knowledge LoRAs for query q. For an adapted linear layer ℓ, each knowledge LoRA produces the update

$$\Delta W_i^{(\ell)} = B_i^{(\ell)} A_i^{(\ell)}, \tag{19}$$

where $A_i^{(\ell)} \in \mathbb{R}^{r_i \times d_{\text{in}}}$ and $B_i^{(\ell)} \in \mathbb{R}^{d_{\text{out}} \times r_i}$. The desired query-specific knowledge update is the weighted sum

$$\Delta W_{\text{know}}^{(\ell)} = \sum_{i \in \mathcal{S}_q^+} g_i B_i^{(\ell)} A_i^{(\ell)}. \tag{20}$$

A seemingly natural alternative is to combine the low-rank factors independently. For example, one might construct

$$\widetilde{A}^{(\ell)} = \sum_i a_i A_i^{(\ell)}, \qquad \widetilde{B}^{(\ell)} = \sum_i b_i B_i^{(\ell)}. \tag{21}$$

However, multiplying the resulting factors gives

$$\widetilde{B}^{(\ell)} \widetilde{A}^{(\ell)} = \left(\sum_i b_i B_i^{(\ell)} \right) \left(\sum_j a_j A_j^{(\ell)} \right) = \sum_i \sum_j b_i a_j B_i^{(\ell)} A_j^{(\ell)}. \tag{22}$$

The terms with $i = j$ correspond to the original LoRA updates, but Eq. 22 additionally introduces cross-LoRA terms $B_i^{(\ell)} A_j^{(\ell)}$ for $i \neq j$. These terms were not produced by any individual document-derived LoRA and therefore change the intended parameter update. Moreover, applying the same coefficient to both factors would scale the corresponding diagonal term quadratically rather than linearly.

To preserve the weighted sum exactly, we instead concatenate the LoRA factors along their rank dimensions. Using the same ordering of active LoRAs in both factors, we define

$$A_q^{(\ell)} = \mathrm{concat}_{\mathrm{row}} \left(g_1 A_1^{(\ell)}, \ldots, g_k A_k^{(\ell)} \right), \tag{23}$$

and

$$B_q^{(\ell)} = \mathrm{concat}_{\mathrm{col}} \left(B_1^{(\ell)}, \ldots, B_k^{(\ell)} \right), \tag{24}$$

where $k = |\mathcal{S}_q^+|$. For equal-rank LoRAs with rank r, we have $A_q^{(\ell)} \in \mathbb{R}^{kr \times d_{\mathrm{in}}}$ and $B_q^{(\ell)} \in \mathbb{R}^{d_{\mathrm{out}} \times kr}$. More generally, the resulting rank dimension is $\sum_i r_i$.

By block matrix multiplication,

$$\begin{aligned} B_q^{(\ell)} A_q^{(\ell)} &= \begin{bmatrix} B_1^{(\ell)} & \cdots & B_k^{(\ell)} \end{bmatrix} \begin{bmatrix} g_1 A_1^{(\ell)} \\ \vdots \\ g_k A_k^{(\ell)} \end{bmatrix} \\ &= \sum_{i=1}^{k} g_i B_i^{(\ell)} A_i^{(\ell)} \\ &= \Delta W_{\mathrm{know}}^{(\ell)}. \end{aligned} \tag{25}$$

Thus, rank concatenation exactly implements the desired weighted sum without introducing cross-LoRA interactions.

Conflict resolution is applied before this composition step. Consequently, mutually incompatible memories from the same conflict group do not simultaneously contribute to Eq. 25; only memories with nonzero conflict-aware weights are concatenated. The composition mechanism is therefore used primarily to combine complementary parametric memories selected for the current query.

The main cost of exact concatenation is an increase in effective LoRA rank. If k rank-r LoRAs are active, the composed update has effective rank at most kr. In DocPatch, this growth is bounded by the retrieval budget because only a small top-k candidate set is considered, and conflict resolution may further reduce the number of active memories. We choose this exact construction rather than a lower-rank approximation because it preserves each document-derived parameter update and its routing weight without introducing unintended cross terms.

A.2 Query Representation Extraction

We use the frozen base LLM as the query representation extractor. Given a raw-text question q, the frozen LLM produces token-level hidden states

$$H_q = F_{\mathrm{LLM}}(q), \tag{26}$$

where only hidden states corresponding to non-padding tokens are retained. We aggregate these token representations into a fixed-dimensional query feature

$$h_q = \mathrm{Pool}\left(\{ H_{q,t} : t \in \mathcal{T}_q \} \right), \tag{27}$$

where $\mathcal{T}_q$ denotes the set of non-padding token positions.

A lightweight learned MLP projector P_q then maps this feature into the routing space:

$$e_q = P_q(h_q). \tag{28}$$

The resulting representation e_q is then taken as input of the routing module, with the cached LoRA-derived representation z_i, to compute the routing score for each candidate knowledge LoRA. During router training, the base LLM remains frozen and only the query projector and routing components are optimized. After training, the query representation module is also frozen for downstream evaluation. In the main text, we use $E(q)$ to denote the query encoding process for simplicity.

A.3 LoRA-to-Feature Mapping

We assume the same configuration for original knowledge lora generated by the hypernetwork introduced in Doc-to-LoRA (Charakorn et al., 2026). Then, to construct a compact representation of lora for routing from each knowledge LoRA, we summarize its effective parameter update rather than directly flattening the LoRA factors. For an adapted layer ℓ, let

$$\Delta W_i^{(\ell)} = B_i^{(\ell)} A_i^{(\ell)}.$$

We introduce fixed projection matrices

$$P_{\text{in}}^{(\ell)} \in \mathbb{R}^{d_{\text{in}} \times p}, \qquad P_{\text{out}}^{(\ell)} \in \mathbb{R}^{d_{\text{out}} \times p},$$

with $p < d_{\text{in}}, d_{\text{out}}$, and compute a low-dimensional sketch

$$S_i^{(\ell)} = {P_{\text{out}}^{(\ell)}}^{\top} B_i^{(\ell)} A_i^{(\ell)} P_{\text{in}}^{(\ell)} \in \mathbb{R}^{p \times p}.$$

This sketch can be evaluated efficiently as

$$S_i^{(\ell)} = \left({P_{\text{out}}^{(\ell)}}^{\top} B_i^{(\ell)} \right) \left(A_i^{(\ell)} P_{\text{in}}^{(\ell)} \right),$$

without explicitly materializing the dense update $\Delta W_i^{(\ell)}$. We then flatten and aggregate the layer-wise sketches and map them into the routing space using a learned MLP projector P_{LoRA}:

$$z_i = P_{\text{LoRA}} \left(\text{Pool}_{\ell \in \mathcal{T}} \, \text{vec} \left(S_i^{(\ell)} \right) \right).$$

The resulting z_i is used only as a compact retrieval representation, while the full LoRA parameters remain the parametric knowledge memory applied after retrieval.

A.4 Constitution QA Construction

We construct Constitution QA from the original constitutional texts and subsequent amendments of seven constitutions. The corpus contains 4,908 document chunks, 10,182 training questions, and 3,563 development questions. Each example consists of a question, answer, reasoning trace, and annotations identifying the relevant source constitution and supporting chunk(s).

The corpus provides supervision at both the source and chunk levels. Source labels identify the constitution relevant to a question, while supporting-chunk annotations identify the specific knowledge units required for answering it. Constitutional amendments also introduce natural document evolution, with later provisions potentially modifying or superseding earlier ones.

We use Constitution QA to train the knowledge-LoRA router. Source annotations supervise the source-level routing objective, while supporting-chunk annotations provide chunk-level supervision. After training, the router is frozen and applied to HotpotQA without further router optimization. Thus, Constitution QA serves as the router-training corpus, while the main routing results reported in the paper evaluate cross-domain transfer on HotpotQA.

A.5 Conflict Identification and Resolution

Our current formulation separates conflict identification from conflict resolution. Conflict relationships are provided during knowledge-bank construction through document-version metadata. Knowledge items corresponding to incompatible alternatives of the same underlying information are assigned the same conflict-group identifier, while version metadata specifies their precedence. At inference time, the conflict-resolution module operates only over retrieved members of these annotated groups.

This design allows us to isolate the problem studied in this work: how evolving parametric memories should be retained and selectively activated once their version relationships are known. Automatically identifying contradictions in arbitrary unversioned document collections would require an additional semantic conflict-detection component and is outside the scope of the present work.

A.6 Role of Counterfactual Reasoning Training

A central challenge in training the global reasoning adapter is preventing it from learning a fixed mapping from questions to answers. If the same question always appears with the same target, the adapter may answer from the pretrained model prior or memorize task-specific associations without relying on the activated knowledge LoRA.

We therefore construct counterfactual training pairs in which the question remains fixed while the activated knowledge memory and target answer change. For example, an earlier-version LoRA may support y_{old}, whereas a later-version LoRA for the same question supports y_{new}. Because the question itself provides no information about which target is correct, successful prediction requires conditioning on the active parametric knowledge.

This supervision is complementary to metadata-based conflict resolution. The resolver determines which knowledge memories are active, while counterfactual training teaches the global reasoning adapter to follow the resulting knowledge state rather than reverting to the frozen model's prior.

A.7 Storage and Inference-Time Trade-offs

DocPatch shifts part of the cost of document access from inference-time context processing to offline parametric storage. Each document chunk is represented by a cached knowledge LoRA together with a compact routing representation and metadata. Consequently, the persistent knowledge bank can require more storage than a conventional dense-retrieval index, which primarily stores document embeddings and source text.

The efficiency results in the main paper therefore focus specifically on inference-time cost: input context length, answer-generation latency, and peak GPU memory. These measurements capture the benefit of avoiding repeated processing of retrieved document text, but should not be interpreted as a claim that DocPatch minimizes persistent storage. This represents an explicit trade-off between larger reusable parametric memory and reduced per-query context processing.

A.8 Multi-Level Router Supervision

The knowledge router is trained as a classification model over candidate knowledge LoRAs. For each question, the router produces a probability distribution over the candidate memories. Because a question may depend on multiple supporting chunks, we use a normalized multi-hot target that assigns equal target mass to all ground-truth knowledge LoRAs.

The chunk-level objective therefore encourages every supporting memory to receive high probability while suppressing irrelevant candidates. This is important for multi-hop questions, where retrieving only one of several required memories may be insufficient for answering correctly.

We additionally use source-level supervision by aggregating the probabilities of chunks belonging to the same source. The source-level objective provides a coarser signal for distinguishing the relevant document from unrelated sources, while the chunk-level objective resolves fine-grained ambiguity among memories within that source.

Together, the two objectives train the router to first identify the relevant document region and then assign high probability to all knowledge LoRAs required by the question.