Decoupled Mixture-of-Experts for Knowledge Injection

Anonymous ACL submission

Abstract

We introduce **Decoupled Mixture-of-Experts (DMoE)**, an innovative architectural framework designed to address the challenges of knowledge injection in large language models (LLMs). Traditional knowledge injection methods, such as Retrieval-Augmented Generation (RAG) and Supervised Fine-Tuning (SFT), suffer from issues such as shallow integration, catastrophic forgetting, and interference during continual updates. Inspired by how MoE architectures isolate knowledge by splitting computation across experts, DMoE achieves extreme isolation by decoupling both experts and router from the base model. Each expert in DMoE represents a modular unit of knowledge that can be added or updated independently, while the router activates relevant experts when the current model lacks sufficient knowledge during generation, as detected by token-level uncertainty exceeding a predefined threshold. Experiments show that DMoE is broadly competitive and often improves answer quality over retrieval- and adapter-based baselines, while substantially reducing latency and memory compared with dynamic retrieval methods that invalidate KV-cache. This work presents a promising direction for integrating external knowledge into LLMs in an effective and efficient way.

1 Introduction

Large Language Models (LLMs) have demonstrated exceptional generalization capabilities across a wide range of tasks (Brown et al., 2020; Chowdhery et al., 2023). However, they often struggle to capture domain-specific or time-sensitive knowledge, resulting in hallucinations or outdated responses at inference time (Song et al., 2025; Xu et al., 2024). To mitigate this issue, a growing body of research has explored **knowledge injection** techniques that incorporate external information during or after model training (Ovadia et al., 2023; Lauscher et al., 2020), such as Retrieval-Augmented Generation (RAG) (Borgeaud et al., 2022; Lewis et al., 2020), Supervised Fine-Tuning (SFT) (Wang et al., 2022; Mishra et al., 2021).

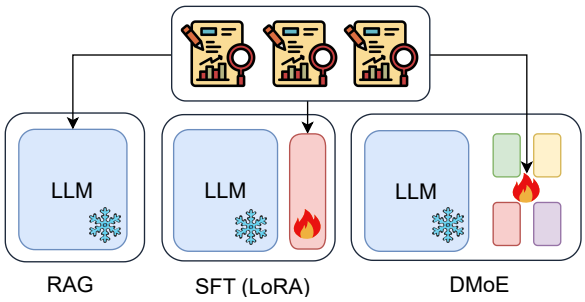

Figure 1: **Comparison of different paradigms for knowledge injection.** (a) RAG relies on external context, leading to shallow integration. (b) Fine-tuning injection overwrites existing parameters, often causing catastrophic forgetting and knowledge conflict. (c) Our proposed DMoE framework decouples knowledge modules from the base model, enabling effective and efficient knowledge integration.

As illustrated in Figure 1, RAG-based methods augment the model input with retrieved documents, yet the incorporated knowledge remains at the prompt level without integration into the model's parameter space, providing only shallow knowledge augmentation. In contrast, SFT and its parameter-efficient variants, e.g., LoRA (Hu et al., 2022) directly adjust model parameters to encode new information. While effective in the short term, such parameter-level updates often cause catastrophic forgetting and knowledge conflict by overwriting previously learned knowledge within a shared parameter space.

These challenges point to a deeper architectural bottleneck: traditional dense LLMs entangle heterogeneous knowledge within a single monolithic context or parameter space, offering no explicit mechanism for knowledge isolation, routing, or incremental expansion. This observation motivates

a shift toward modular designs that can store, access, and update knowledge in a structured and interpretable manner.

Inspired by the Mixture-of-Experts (MoE) principle of conditional computation and specialization (Jacobs et al., 1991; Shazeer et al., 2017; Fedus et al., 2022), we propose **Decoupled Mixture-of-Experts (DMoE)**: an architectural paradigm that *fully decouples* experts and routers from the base model (right of Figure 2). DMoE converts external knowledge corpora into independent adaptive experts, each acting as a parameterized representation of a specific knowledge unit, while a lightweight router dynamically activates the relevant experts at inference time.

To remain efficient for real-world autoregressive generation, DMoE attaches experts only to the *final-layer feed-forward network*, enabling full KV-cache reuse during inference. Consequently, when the knowledge corpus changes, only the corresponding experts are modified or newly added, without perturbing other irrelevant knowledge parameters. At inference time, whenever the model lacks sufficient internal knowledge, it routes to the most relevant experts to support more specialized reasoning.

We evaluate DMoE on a suite of knowledge-intensive benchmarks with an emphasis on effectiveness and efficiency. Across datasets, DMoE improves answer quality (e.g., EM/F1) over dense-model baselines, while also offering a substantial improvement in inference cost. In addition, our ablation studies consolidate the rationale behind DMoE's design. They show that (i) a decoupled expert mechanism is preferable to conventional coupled MoE variants when conducting knowledge injection; (ii) DMoE's routing behavior is robust to the triggering threshold; and (iii) expert attachment must be cache-compatible, where restricting experts to the final-layer FFN enables effective KV-cache reuse and yields the best overall performance.

In summary, we make the following key contributions:

- **A modular DMoE architecture for knowledge injection.** We propose Decoupled Mixture-of-Experts (DMoE), which fully decouples experts and router from base model.
- **Final-layer FFN expert integration for KV-cache reuse.** We design a paradigm that experts are attached only to the final-layer FFN so that KV-cache reuse is preserved, enabling efficient reasoning.
- **Effectiveness–efficiency evaluation with design justification.** We provide comprehensive empirical results on knowledge-intensive benchmarks, demonstrating consistent gains in answer quality and inference efficiency over baselines.

2 Related Work

2.1 Knowledge Injection Methods

RAG RAG (Lewis et al., 2020; Borgeaud et al., 2022) empowers LLMs to access external knowledge at inference time by retrieving relevant documents from a background corpus and incorporating them into the input prompt. While this approach enhances factual accuracy and expands the model's accessible knowledge, the retrieved text is appended to the prompt rather than integrated into model's internal parameterization, which limits how deeply the model can exploit injected facts (Alghisi et al., 2024; Balaguer et al., 2024; Soudani et al., 2024). Besides, most RAG systems rely on a single round of retrieval based solely on the initial query. Such static retrieval strategies often prove insufficient for tasks that require multi-step reasoning or long-form generation, where the model's information needs updating throughout the generation process (Yang et al., 2025; Joshi et al., 2024). Two broad directions have been proposed to mitigate these limitations. Parametric RAG (Su et al., 2025) converts external knowledge into parameterized representations, enabling deeper integration between retrieval and generation; however, it still fails to support continuous knowledge injection. Dynamic RAG approaches, such as FLARE (Jiang et al., 2023) and DRAGIN (Su et al., 2024), determine when to retrieve during generation by monitoring token-level uncertainty or attention weights, providing a principled criterion for the timing of knowledge injection. Nevertheless, context-level injection disrupts KV-cache in transformer decoding, reducing efficiency in practical applications.

SFT SFT (Mishra et al., 2021; Ouyang et al., 2022; Taori et al., 2023) injects knowledge by fine-tuning the model on synthetic or human-annotated question-answer pairs derived from external sources. SFT can effect deeper, parameter-level incorporation of new information, but updating base model's weights inevitably risks catas-

trophic forgetting and knowledge conflict (Goodfellow et al., 2013; Kemker et al., 2018). To reduce cost and mitigate interference, a variety of parameter-efficient fine-tuning (PEFT) techniques (Houlsby et al., 2019; Han et al., 2024) have been developed that freeze the majority of base parameters and train only a small set of adapters, e.g., LoRA (Hu et al., 2022), Prompt Tuning (Lester et al., 2021), Prefix Tuning (Li and Liang, 2021). Although PEFT methods effectively reduce training cost and alleviate interference with the knowledge of base model, the newly injected knowledge still cannot be modularized or isolated, so both knowledge conflict and catastrophic forgetting remain unresolved under knowledge updates.

2.2 Previous Explorations in Decoupling MoE

Several works have attempted to decouple components of MoE to improve efficiency or training stability. Read-ME (Cai et al., 2024) introduces a pre-gating router partially decoupled from the MoE backbone to enable expert-aware batching and caching. EvoMoE (Nie et al., 2021) decouples expert training from sparse gating through a progressive dense-to-sparse evolution scheme. StableMoE (Dai et al., 2022) distills and then freezes a router decoupled from the backbone to stabilize routing. DeMo (Wang et al., 2025) designs a feature-level decoupled MoE for multi-modal object re-identification, focusing on modality-specific feature weighting. Unlike previous works that decouple either the router or the training phases, our approach *fully* decouples both experts and router from the base model, isolating each unit of knowledge and enabling a scalable routing mechanism.

3 Methodology

3.1 Preliminaries and Task Formulation

We first characterize the effect of knowledge injection through its impact on the model's predictive uncertainty, following prior uncertainty-aware retrieval and routing methods (Jiang et al., 2023; Su et al., 2024). At each decoding step t, the model produces a softmax distribution p_t over the vocabulary. We define the token uncertainty (TU) at step t as the entropy of this distribution:

$$\mathrm{TU}_t = -\sum_v p_t(v) \log p_t(v). \quad (1)$$

During auto-regressive generation, the model does not have access to the correct next token and thus cannot directly assess whether its prediction is accurate or whether additional knowledge is required. TU therefore serves as a practical inference-time signal that reflects how uncertain the model is about its next-token distribution: a larger TU indicates that the model lacks sufficient internal knowledge for the current input and is more likely to benefit from external knowledge injection.

Formally, let $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_N\}$ denote a set of external knowledge units, each representing a semantically coherent piece of information (e.g., a fact, a document, or a reasoning pattern). The objective of knowledge injection is to find an augmented model θ' that incorporates a subset of knowledge $\mathcal{K}^* \subseteq \mathcal{K}$ to minimize TU on knowledge-relevant samples $\mathcal{D}_{\text{know}}$:

$$\theta' = \arg\min_{\theta'} \mathbb{E}_{x \sim \mathcal{D}_{\text{know}}} \left[\mathrm{TU}(x; \theta' \cup \mathcal{K}^*) \right]. \quad (2)$$

Existing approaches differ in how they realize this objective. In RAG and other context-level methods, the model parameters remain unchanged, i.e., $\theta' = \theta$, and the external knowledge $\mathcal{K}^*$ is injected only through the input context, resulting in a shallow form of knowledge injection. In contrast, parameter-level knowledge injection methods such as SFT modify θ directly to encode $\mathcal{K}^*$, enabling the model to internalize new knowledge. However, such direct modification frequently leads to catastrophic forgetting or knowledge conflicts, as updates to θ overwrite or interfere with existing knowledge. These observations motivate the need for a mechanism that can update θ to minimize TU while preserving isolation between newly injected and existing knowledge.

3.2 Model Architecture

As illustrated in Figure 2, **DMoE** architecture consists of three main components: a frozen *base model*, a *router*, and a large set of *experts*. Unlike conventional MoE architectures that replace the feed-forward network (FFN) layers of a dense model with a jointly trained gating network and multiple experts, DMoE decouples both router and experts from the base model.

3.2.1 Experts

Expert architecture. In DMoE, each expert represents a semantic unit of knowledge, which is practically realized via lightweight adapters (e.g., LoRA (Hu et al., 2022) or other PEFT variants) attached to the FFN layers, which are known to

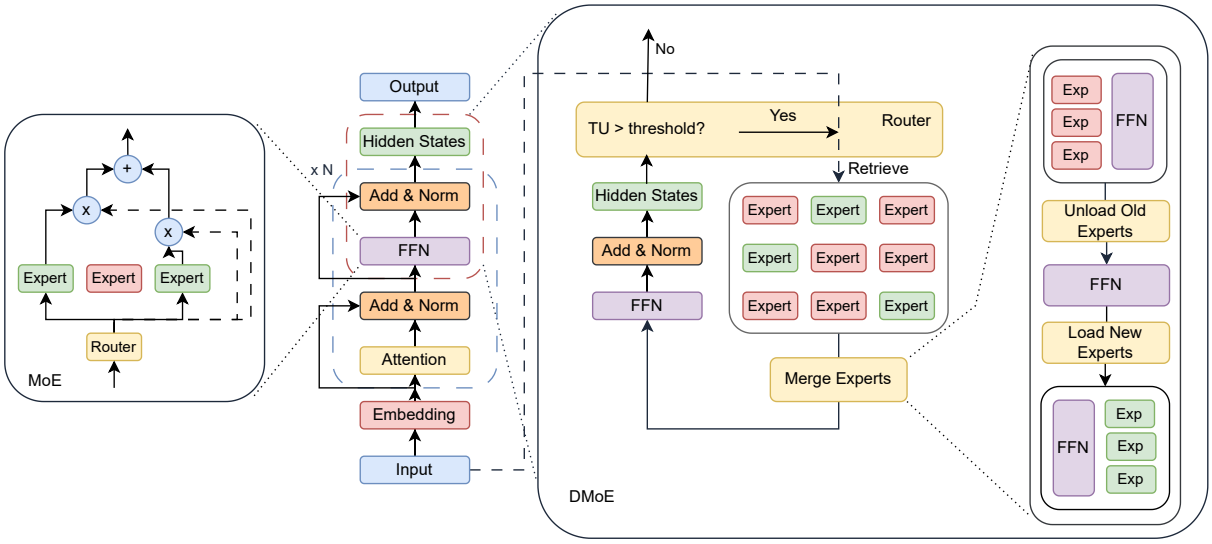

Figure 2: **Comparison of architectures** between the dense model, traditional Mixture-of-Experts (MoE), and the proposed Decoupled Mixture-of-Experts (DMoE). In the MoE architecture, the feed-forward layers of a dense model are replaced with a coupled router-expert network. In contrast, DMoE decouples both the router and experts from the base model. During inference, DMoE retrieves and updates relevant experts when knowledge injection is required, enabling adaptive and non-destructive knowledge integration.

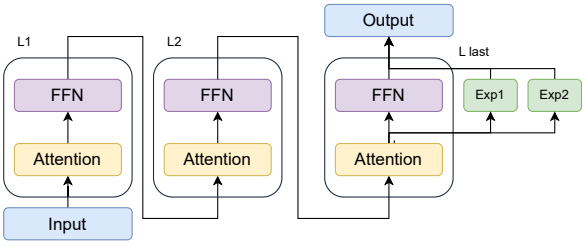

Figure 3: **Illustration of expert placement in DMoE.** Experts are attached only to the FFN of the final Transformer layer so that the hidden states feeding earlier attention blocks remain unchanged. Placing experts at intermediate layers would modify representations consumed by subsequent attention modules, preventing efficient reuse of KV-cache during decoding.

store factual and semantic knowledge (Geva et al., 2021b; Nanda et al., 2023; Yu and Ananiadou, 2024). To maintain practical KV-cache reuse during decoding, we attach experts only to the final-layer FFN (Figure 3); a detailed analysis of this design choice is provided in Section 4.4.1.

Expert generation. Since our focus is architectural, we adopt a simple and transparent expert construction pipeline based on document augmentation and training, following the same recipe as PRAG (Su et al., 2025). This yields a set of experts $\{\Delta\theta_i\}$, each specialized to a knowledge unit.

3.2.2 Router

Triggering. Routing is activated only when the current model exhibits insufficient knowledge of the input. Since our goal is to reduce TU, there is no need to involve additional experts when the model is already confident. At each decoding step t, we trigger routing only when TU is above a given threshold:

$$\text{Trigger}_t = \mathbb{I}\big[\text{TU}_t > \tau\big], \tag{3}$$

where τ is a given threshold. When $\text{Trigger}_t = 1$, the router deactivates outdated experts and activates relevant experts; otherwise, decoding proceeds with the current model.

Routing. To make knowledge injection both effective and scalable, the router should (i) select experts that most likely reduce token uncertainty (TU) under the current context, while (ii) remaining fully *decoupled* from the base model so that experts can be added/removed/updated without retraining any neural router. We therefore adopt a *BM25-based lexical router* implemented via an external inverted index.

Each expert E_i is trained on a knowledge unit $\mathcal{K}_i$ and is associated with a text surrogate $\mathcal{D}_i$ (the document plus its augmentations used in expert training; see Section 4.1.3). At decoding step t, we form a routing query q_t from the task input and the current generation context, and retrieve experts whose associated texts best match q_t under BM25: When a

high-TU position triggers routing, q_t is constructed from the generated prefix up to, but excluding, that position, so the retrieved experts are conditioned on exactly the context already represented in the KV cache.

$$E_{\text{sel}} = \text{Top-k}_{i \in [N]} \ \text{BM25}(q_t, \mathcal{D}_i). \quad (4)$$

This design is training-free, incrementally updatable (adding an expert only inserts $\mathcal{D}_i$ into the index), and avoids introducing an additional neural encoder that would partially re-couple routing with model parameters. Finally, we activate E_{sel} and update the effective parameters as

$$\theta_{t+1} = \theta_t + \sum_{E_i \in E_{\text{sel}}} \Delta\theta_i, \quad (5)$$

and keep this set of experts active until the next triggering event to avoid redundant expert swaps at every decoding step.

4 Experiments

4.1 Experimental Setup

4.1.1 Benchmarks and Metrics

We conduct comprehensive evaluations on a diverse suite of knowledge-intensive benchmarks, each representing a distinct aspect of the knowledge injection problem. **HotpotQA** (Yang et al., 2018) focuses on multi-hop reasoning, where the model must integrate evidence from multiple documents to answer complex questions that require reasoning across facts. **ComplexWebQuestions** (Talmor and Berant, 2018) extends this setting to an open-domain environment, challenging the model to perform compositional reasoning and inject relevant knowledge from large-scale web content. **Quasar-T** (Dhingra et al., 2017) further evaluates open-domain retrieval and reasoning by requiring models to locate and synthesize factual knowledge from a large corpus to answer trivia-style questions. **StrategyQA** (Geva et al., 2021a) assesses implicit multi-hop reasoning, where the required knowledge is not explicitly stated in the question and must be inferred or retrieved based on world knowledge.

For evaluation, we primarily assess the effectiveness of knowledge injection methods in the main results. Effectiveness is measured using EM and F1 (ACC for StrategyQA), which reflect how well a model integrates and utilizes injected knowledge. We analyze efficiency separately in Section 4.3 using the average inference time and average GPU memory consumption per sample.

4.1.2 Baselines

We compare DMoE against representative baselines spanning the major paradigms of knowledge injection: Basic-RAG, FLARE, PRAG, and SFT-LoRA, all built on *Llama-3.2-1B-Instruct* and *Qwen2.5-1.5B-Instruct*. For an equitable comparison, all retrieval-based methods share the same retrieval corpus of 27,613 passage-level documents, the same retriever (Elasticsearch BM25), retrieval budget ($k = 3$), and greedy decoding; retrieval queries are truncated to 128 tokens, and the corpus is not additionally chunked because each item is already a short single-topic passage of approximately 100 words. Basic-RAG performs one-shot retrieval and appends the retrieved passages to the prompt, FLARE retrieves during decoding and therefore cannot reuse the KV-cache, PRAG combines retrieved passages with a corresponding parametric adapter using the same augmentation-and-adapter recipe as our expert construction pipeline, and SFT-LoRA trains a single LoRA adapter on supervision merged across the full corpus.

4.1.3 DMoE Setup

Before training experts, each base model undergoes a one-epoch held-out format-stabilization warmup. Concretely, warmup uses the same task and system instructions as the main evaluation, but only evaluation-disjoint instances, and is intended to stabilize output formatting consistency rather than inject factual or benchmark-specific knowledge. This stage is retained to match the PRAG recipe and maintain a controlled comparison; its impact is analyzed in Section A.1.

The external knowledge corpus used to construct experts is drawn from the Wikipedia corpus used in DPR (Karpukhin et al., 2020). This pool is used by all retrieval-based baselines and by DMoE, so comparisons are controlled under the same accessible knowledge source. Importantly, corpus construction and expert training do not use answer strings, gold supporting-document annotations, or any manually labeled rationales. Each document is augmented only from its own passage text by: (1) a single paraphrased rewrite that preserves its factual content, and (2) three question-answer pairs generated from the document. These augmentations enrich the supervision signals for expert training and help each expert better capture the underlying knowledge unit.

For each knowledge unit, we train a dedicated expert implemented as a LoRA adapter with rank 4,

Table 1: **Main results on four knowledge-intensive benchmarks.** We report EM/F1 or ACC across datasets and base models. Best values are in bold.

METHOD	CWQ		HOTPOTQA		QUASAR-T		STRATEGYQA
	EM	F1	EM	F1	EM	F1	ACC
BASE MODEL: LLAMA3.2-1B							
BASIC-RAG	0.1633	0.2384	0.1700	0.2463	0.2800	0.3513	0.4333
FLARE	0.2400	0.3154	0.0733	0.1303	0.1867	0.2190	0.5367
PRAG	**0.2500**	0.3284	0.0733	0.1427	0.2200	0.2514	0.5600
SFT-LORA	0.2167	0.3092	0.0767	0.1325	0.2133	0.2481	0.5533
DMOE (OURS)	0.2467	**0.3479**	**0.1800**	**0.2553**	**0.3133**	**0.3658**	**0.5667**
BASE MODEL: QWEN2.5-1.5B							
BASIC-RAG	0.1233	0.1888	**0.1567**	**0.2446**	0.1833	0.2549	0.3667
FLARE	0.1200	0.1806	0.1000	0.1717	0.1733	0.2242	0.2500
PRAG	**0.2167**	0.3050	0.1033	0.1583	0.1733	0.2312	0.5833
SFT-LORA	0.1967	0.3092	0.1033	0.1595	0.1767	0.2292	0.5733
DMOE (OURS)	**0.2167**	**0.3285**	0.1533	0.2375	**0.2333**	**0.2795**	**0.6133**

scaling factor $\alpha = 16$, and learning rate 1×10^{-5}. Experts are trained for one epoch and inserted only into the final-layer feed-forward network to ensure full compatibility with KV-cache reuse during autoregressive decoding.

During inference, we use greedy decoding and activate the top-k experts ($k = 3$ by default). We use BM25 to construct expert and context representations. All efficiency measurements (inference time and GPU memory usage) are conducted on a single NVIDIA A100 80GB GPU.

4.2 Overall Results

Table 1 summarizes overall effectiveness across all baselines. We set test corpus size to 300 for all benchmarks.

4.2.1 Effectiveness

Table 1 shows a general advantage for DMoE. Across the 14 reported effectiveness metrics, DMoE obtains the best or tied-best score on 11 metrics, including the strongest F1 on CWQ for both base models, both Quasar-T metrics for both base models, and StrategyQA ACC for both base models. This pattern suggests that DMoE is usually the strongest method in the reported comparison. The exceptions are concentrated rather than scattered. On CWQ, PRAG slightly exceeds DMoE in EM for Llama3.2-1B and ties it for Qwen2.5-1.5B, although DMoE has the higher CWQ F1 in both cases. On HotpotQA with Qwen2.5-1.5B, Basic-RAG is marginally better than DMoE on both EM and F1, while DMoE is best on both HotpotQA metrics with Llama3.2-1B. Thus, Table 1 supports a measured conclusion: DMoE is broadly competitive and most often best among the evaluated methods, but the results do not establish dominance on every dataset, base model, or metric.

4.3 Efficiency Analysis

As shown in Table 2, static baselines such as Basic-RAG, PRAG, and SFT-LoRA can incur lower per-sample cost because they do not modify retrieval or routing decisions during decoding. Among methods that adapt generation online, however, DMoE is roughly $3\times$ faster than FLARE while also using substantially less memory. The core reason is KV-cache reuse. FLARE-style dynamic RAG interleaves generation with retrieval and prompt rewriting, which grows the context over time and invalidates cache reuse; consequently, attention over an expanding prefix is repeatedly recomputed, inflating both latency and memory.

In contrast, DMoE keeps the prompt state stable and performs adaptation by conditionally activating lightweight final-layer experts. Since experts are confined to the last FFN layer, the KV-cache remains fully valid throughout decoding, largely

Table 2: **Efficiency analysis.** We report all available average inference latency and GPU memory measurements per sample. Lower is better, and best values within each base-model block are in bold.

METHOD	TIME (S)	GPU (GB)
BASIC-RAG	1.8900	**2.5400**
FLARE	9.2643	13.9718
PRAG	**1.3600**	4.8300
SFT-LORA	1.6700	4.8200
DMOE (OURS)	2.6656	7.2352

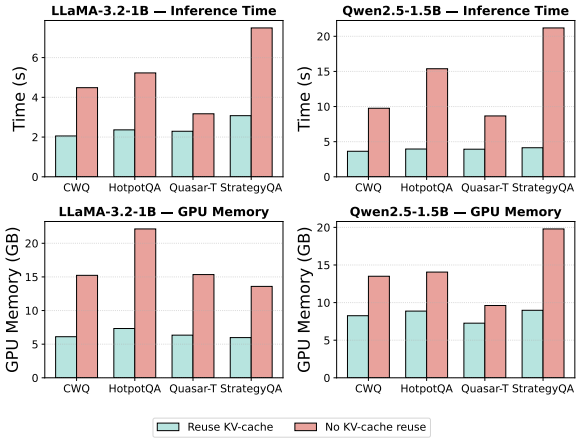

Figure 4: **KV-cache reuse substantially improves autoregressive inference efficiency.** We report the inference latency (top) and GPU memory footprint (bottom) for two base models (Llama-3.2-1B and Qwen2.5-1.5B) across four benchmarks, comparing *Reuse KV-cache* vs. *No KV-cache reuse*. KV-cache reuse consistently reduces both time and memory across all datasets.

avoiding the context-growth penalty and redundant attention computation. This cache-safe design translates into consistent time and memory advantages in practice: DMoE reduces GPU memory usage by roughly $1.6\times$–$1.9\times$ compared to FLARE.

Figure 4 further confirms the centrality of KV caching: across all benchmarks and both base models, enabling KV caching yields a $1.3\times$–$5.1\times$ latency speedup and a $1.2\times$–$2.5\times$ reduction in GPU memory usage, motivating cache-compatible expert placement.

4.4 Ablation Study

To better understand the contribution of individual architectural components in DMoE, the main paper focuses on two architectural questions: (1) where experts should be placed to preserve KV-cache compatibility, and (2) whether DMoE's gains can be explained by simply using a conventional MoE backbone. Additional ablations, including warmup, sensitivity to router top-k, TU-triggered routing analyses, LoRA hyperparameter sensitivity, and retriever choice, are reported in Appendix A.

4.4.1 Experts: Impact of Expert Layer Placement

A practical requirement for DMoE is that expert activation must preserve KV-cache reuse during auto-regressive decoding. In a standard transformer decoder, each layer ℓ computes

$$h_{\ell+1} = \mathcal{F}_\ell(h_\ell), \tag{6}$$

where h_ℓ is the hidden state, and the keys and values for layer ℓ's attention are linear projections of h_ℓ. Introducing an expert at any intermediate layer k modifies the hidden state to

$$\tilde{h}_{k+1} = h_{k+1} + \Delta(h_k), \tag{7}$$

which propagates forward to all subsequent layers $j > k$. Because the KV-cache stores $K_j^{\text{cached}} = W_K h_j$ and $V_j^{\text{cached}} = W_V h_j$ computed from the unmodified trajectory, any modification $\tilde{h}_j \neq h_j$ requires recomputing

$$K_j^{\text{new}} = W_K \tilde{h}_j, \qquad V_j^{\text{new}} = W_V \tilde{h}_j, \tag{8}$$

breaking KV-cache compatibility and causing substantial latency overhead.

This constraint is avoided only when experts are applied at the **final FFN layer**. In a decoder-only model with L layers, the output of the last FFN (h_{L+1}) is projected directly to logits and does not feed into any further attention computation. Thus, placing experts exclusively at $k = L$ guarantees

$$\forall j \leq L, \quad \tilde{h}_j = h_j, \tag{9}$$

ensuring that all cached keys and values remain valid.

Table 3 empirically confirms this analysis. When experts are inserted into earlier or multiple layers, performance drops sharply due to KV-cache mismatch. In contrast, attaching experts only to the last FFN layer is generally competitive and often best.

These findings demonstrate a key property of DMoE: minimal-intrusive expert placement at the final layer preserves fast decoding while still enabling effective knowledge injection. Together with adaptive triggering, this design provides an effective and efficient mechanism for integrating new knowledge during inference.

4.4.2 DMoE vs. Conventional MoE Backbones

To clarify whether our gains are simply an artifact of adopting an MoE-style backbone, we include a coupled MoE model as a structural control. Specifically, we evaluate *allenai/OLMoE-1B-7B-0125-Instruct* and adapt it with the same SFT-LoRA recipe used in our dense baselines. We choose it because its activated parameter scale is comparable to DMoE, i.e., both systems perform sparse conditional computation with a similar amount of per-token active capacity. This design makes the

Table 4: **DMoE vs. a coupled MoE backbone.** We compare DMoE against a finetuned MoE model (OLMoE-1B-7B) under SFT-LoRA. DMoE achieves higher effectiveness (EM/F1/ACC) while using markedly lower inference time and GPU memory. Higher is better for EM/F1/ACC; lower is better for Time/GPU. Best values are in bold.

METHOD	CWQ		HOTPOTQA		QUASAR-T		STRATEGYQA	EFFICIENCY	
	EM	F1	EM	F1	EM	F1	ACC	TIME (S)	GPU (GB)
DMoE (LLAMA3.2-1B)	**0.2467**	**0.3479**	**0.1800**	**0.2553**	**0.3133**	**0.3658**	0.5667	**2.6656**	**7.2352**
DMoE (QWEN2.5-1.5B)	0.2167	0.3285	0.1533	0.2375	0.2333	0.2795	**0.6133**	3.9138	8.3415
MoE (SFT-LoRA, OLMoE-1B-7B)	0.1500	0.2638	0.1033	0.1790	0.2100	0.2579	0.5900	20.0217	26.0843

Table 3: Performance (EM) when reusing KV-cache while introducing experts at different depths. Only placing experts in the final-layer FFN maintains KV-cache compatibility and yields the best accuracy across all datasets.

SETTING	CWQ	HOTPOTQA	QUASAR-T	STRATEGYQA
LLAMA3.2-1B				
25%	0.2167	0.1600	0.2967	0.5200
50%	0.2433	0.1567	0.3067	0.5333
75%	0.2100	0.1792	0.3100	0.5233
100%	0.2133	0.1633	0.3000	0.5533
ONLY-LAST	**0.2467**	**0.1800**	**0.3133**	**0.5633**
QWEN2.5-1.5B				
25%	0.2033	0.1367	0.2200	0.5633
50%	0.1767	0.1500	**0.2400**	0.5467
75%	0.1800	0.1333	0.2233	0.5400
100%	0.1800	0.1333	0.2200	0.5400
ONLY-LAST	**0.2167**	**0.1533**	0.2333	**0.6133**

comparison informative: it helps factor out the confounding effect of increasing total parameters, and tests whether the improvements come from DMoE's architectural decoupling rather than parameter count.

Table 4 shows that DMoE consistently outperforms the coupled MoE backbone on effectiveness while being substantially more efficient. On CWQ, HotpotQA, and Quasar-T, both DMoE variants achieve higher EM/F1 than OLMoE under SFT-LoRA, and DMoE also matches or exceeds OLMoE on StrategyQA accuracy. More importantly, the system costs differ dramatically: DMoE reduces average inference time from 20.0 s (OLMoE) to 2.7–3.9 s, and cuts GPU memory from 26.1 GB to 7.2–8.3 GB.

These results support our main design rationale for knowledge injection. A conventional MoE backbone offers conditional computation, but its experts are heavy and must remain resident during inference, and its router/expert parameters are coupled to the base model, which makes knowledge updates harder to localize and incurs high runtime footprint. In contrast, DMoE treats experts as external, lightweight knowledge modules and invokes them only when needed, while keeping the base model intact. Therefore, DMoE's gains cannot be attributed to merely increasing model capacity; they arise from a decoupled, cache-safe modularization that yields a strictly better effectiveness-efficiency trade-off than simply switching to a standard MoE architecture.

5 Conclusion

We introduce DMoE, a fully decoupled Mixture-of-Experts framework for knowledge injection. By treating experts as independent knowledge units and activating them only when needed, DMoE enables modular knowledge integration without interfering with the base model. Experiments across knowledge-intensive benchmarks show that DMoE is competitive in both effectiveness and efficiency, while ablations confirm the importance of its decoupled design and adaptive routing. DMoE offers a practical foundation for building LLMs that can incorporate and update external knowledge flexibly, suggesting promising directions for future scalable LLMs.

6 Limitations

While DMoE offers an effective and efficient framework for knowledge injection, several limitations remain. First, TU can indicate missing knowledge but may miss outdated-but-confident errors; nevertheless, it is a practical proxy because the ground-truth next token is unavailable at inference time. Second, we do not compare with knowledge editing methods such as ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b), which target specific factual associations in model parameters rather than external knowledge-module injection for broader QA. Third, experts are trained offline and individually, which limits the framework's ability to support fully online or dynamically evolving knowledge sources. Although document augmentation is performed only from passage text and does

not use benchmark answers or gold supporting annotations, this candidate-pool construction remains benchmark-conditioned and should be evaluated at larger unrestricted-corpus scale in future work. In addition, when multiple top-k experts are activated, we currently use simple linear merging and have not explored alternative merging strategies that may better mitigate conflicts among selected experts. Finally, our comparisons do not include more advanced MoE variants with learned or hierarchical routing, leaving open the question of how DMoE performs relative to these architectures.

References

Simone Alghisi, Massimo Rizzoli, Gabriel Roccabruna, Seyed Mahed Mousavi, and Giuseppe Riccardi. 2024. Should we fine-tune or rag? evaluating different techniques to adapt llms for dialogue. *arXiv preprint arXiv:2406.06399*.

Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo O Nunes, Rafael Padilha, and 1 others. 2024. Rag vs fine-tuning: pipelines, tradeoffs, and a case study on agriculture. *arXiv preprint arXiv:2401.08406*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ruisi Cai, Yeonju Ro, Geon-Woo Kim, Peihao Wang, Babak Ehteshami Bejnordi, Aditya Akella, and Zhangyang Wang. 2024. Read-me: Refactorizing llms as router-decoupled mixture of experts with system co-design. *arXiv preprint arXiv:2410.19123*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Stablemoe: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021a. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021b. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

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

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Ashutosh Joshi, Sheikh Muhammad Sarwar, Samarth Varshney, Sreyashi Nag, Shrivats Agrawal, and Juhi Naik. 2024. Reaper: Reasoning based retrieval planning for complex rag systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4621–4628.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Anne Lauscher, Olga Majewska, Leonardo FR Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020. Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. *arXiv preprint arXiv:2005.11787*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Neel Nanda, Senthooran Rajamanoharan, Janos Kramar, and Rohin Shah. 2023. Fact finding: Attempting to reverse-engineer factual recall on the neuron level. In *Alignment Forum*, page 6.

Xiaonan Nie, Xupeng Miao, Shijie Cao, Lingxiao Ma, Qibin Liu, Jilong Xue, Youshan Miao, Yi Liu, Zhi Yang, and Bin Cui. 2021. Evomoe: An evolutional mixture-of-experts training framework via dense-to-sparse gate. *arXiv preprint arXiv:2112.14397*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2023. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Zirui Song, Bin Yan, Yuhan Liu, Miao Fang, Mingzhe Li, Rui Yan, and Xiuying Chen. 2025. Injecting domain-specific knowledge into large language models: a comprehensive survey. *arXiv preprint arXiv:2502.10708*.

Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 2024. Fine tuning vs. retrieval augmented generation for less popular knowledge. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 12–22.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. Dragin: dynamic retrieval augmented generation based on the information needs of large language models. *arXiv preprint arXiv:2403.10081*.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. 2025. Parametric retrieval augmented generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1240–1250.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, and 1 others. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Yuhao Wang, Yang Liu, Aihua Zheng, and Pingping Zhang. 2025. Decoupled feature-based mixture of experts for multi-modal object re-identification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 8141–8149.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Diji Yang, Linda Zeng, Jinmeng Rao, and Yi Zhang. 2025. Knowing you don't know: Learning when to continue search in multi-round rag through self-practicing. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1305–1315.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Zeping Yu and Sophia Ananiadou. 2024. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3267–3280.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

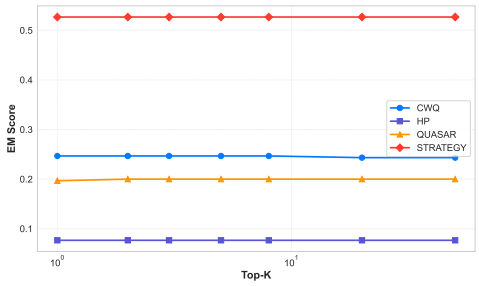

Figure 5: **Impact of the number of activated experts.** Varying top-k across the tested range yields nearly unchanged downstream performance on all four benchmarks, indicating that DMoE is robust to this routing hyperparameter.

A Additional Ablation Studies

A.1 Warmup: Definition and Impact

Since the focus of this work is the DMoE architecture itself, we do not include dedicated instruction-following benchmarks such as IFEval (Zhou et al., 2023) to directly measure the effect of warmup on instruction following. Instead, we assess its impact indirectly through downstream performance on four QA benchmarks. Warmup yields modest gains of about +1.0 EM on CWQ, +0.3 EM on HotpotQA, and +1.3 EM on Quasar-T, but slightly lowers StrategyQA by about 1.0 ACC. This mixed pattern suggests that warmup does not provide a uniform performance benefit and does not indicate a broad degradation in instruction following. The main setting retains warmup to match the PRAG recipe and maintain a controlled comparison.

A.2 Router: Impact of Top-k Experts

Figure 5 studies sensitivity to the number of activated experts while keeping the remaining setup fixed. The curves remain nearly flat across CWQ, HotpotQA, Quasar-T, and StrategyQA, showing that DMoE is highly robust to the choice of top-k. This behavior suggests that the router identifies a sufficiently relevant expert set even with a small activation budget, while increasing the number of activated experts brings only marginal changes in downstream quality. Therefore, the default choice $k = 3$ provides a simple and stable operating point without requiring careful tuning.

A.3 Router: TU-Triggered Routing Analysis

This section analyzes token uncertainty (TU) as the trigger signal used by the router. We focus on three questions: whether TU is aligned with downstream performance, how the trigger and router components contribute separately, and how sensitive DMoE is to the triggering threshold.

Table 5: **Base-model accuracy across token-uncertainty bins.** We group 300 evaluation examples into five entropy bins using the base model's token uncertainty (TU) before expert activation. Higher TU consistently corresponds to lower EM/F1, supporting TU as a useful—though imperfect—proxy for when the model lacks sufficient prior knowledge.

Bin	Entropy range	EM	F1
1 (Low)	[0.593, 0.863]	0.333	0.381
2	[0.863, 1.015]	0.233	0.287
3	[1.017, 1.185]	0.183	0.217
4	[1.188, 1.365]	0.133	0.158
5 (High)	[1.366, 2.824]	0.117	0.142

Table 6: **Component ablation of trigger and router choices.** We disentangle when to activate experts from which experts to select. The full TU-triggered BM25 router performs best, showing that DMoE's gains are not explained by indiscriminate expert activation.

Setting	EM	F1
A. Full DMoE (TU + BM25)	**0.0767**	**0.1309**
B. Random trigger + BM25	0.0633	0.1220
C. TU + random router	0.0733	0.1162
D. Always trigger + BM25	0.0167	0.0421
E. TU + oracle-style expert router	0.0733	0.1288

A.3.1 TU and Downstream Performance

Table 5 sorts 300 evaluation examples by the base model's token uncertainty before expert activation. Low-entropy bins correspond to cases where the base model is already relatively confident, whereas high-entropy bins correspond to cases where the model has more limited prior knowledge. The degradation is monotonic: EM drops from 0.333 in the lowest-entropy bin to 0.117 in the highest-entropy bin, and F1 drops from 0.381 to 0.142. This trend indicates that TU is aligned with knowledge difficulty and is therefore a reasonable signal for deciding when to trigger expert assistance.

A.3.2 Component Ablation: Trigger and Router

Table 6 further disentangles the triggering decision from expert selection. The full setting, which combines TU-based triggering with the BM25 router, achieves the best EM and F1. Replacing TU with a random trigger hurts performance, keeping TU but replacing BM25 with a random router also degrades F1, and always triggering experts collapses

substantially. These results show that the gains are not driven by indiscriminate expert activation: both deciding when to activate experts and selecting which experts to activate matter. Finally, the oracle-style expert router is only comparable to BM25, suggesting that BM25 is already a strong practical router in this parametric-expert pipeline rather than the dominant bottleneck.

A.3.3 Impact of Triggering Thresholds

Figure 6 studies the effect of the TU triggering threshold. Increasing the threshold makes expert activation more selective, which reduces routing frequency and improves inference speed. At the same time, downstream metrics remain relatively stable across the tested range, indicating that DMoE does not require a narrowly tuned threshold to maintain performance. This behavior suggests that the TU criterion captures a useful region where expert activation is most needed, while allowing practitioners to adjust the threshold to trade off effectiveness and efficiency.

A.4 Router: Impact of Retriever Choice

Table 7 studies whether replacing the default BM25 lexical router with SGPT dense retrieval improves expert selection. SGPT yields higher accuracy on StrategyQA and slightly higher F1 on HotpotQA, but it underperforms BM25 on CWQ and Quasar-T. The efficiency profile is also mixed: SGPT is modestly faster in these runs, but consistently consumes about 0.6 GB more GPU memory because dense retrieval introduces additional neural embedding computation and storage. Since SGPT does not provide a stable effectiveness gain and increases memory usage, we use BM25 as the default router in the main experiments.

A.5 Experts: Impact of LoRA Rank and Scaling Factor

Table 8 evaluates the sensitivity of DMoE experts to LoRA rank r and scaling factor α. The default configuration $(4, 16)$ achieves the best result on HotpotQA, CWQ, and Quasar-T, while StrategyQA slightly favors $(8, 32)$. Overall, increasing rank or scaling factor does not produce a consistent improvement, suggesting that the expert adapters do not require large capacity to capture the targeted knowledge units used in our setting. We therefore use $(4, 16)$ as the default configuration because it provides the strongest overall trade-off between effectiveness and parameter efficiency.

B Scalability and Cost Analysis

B.1 Expert Count and Storage Footprint

In DMoE, the number of experts is not a fixed architectural hyperparameter as in standard MoE models. It is determined by how external knowledge is partitioned into knowledge units. In our experiments, we instantiate one expert for each passage-level knowledge unit, resulting in 27,613 corpus-aligned experts for the Llama-3.2-1B expert bank. This design makes the expert bank naturally aligned with the retrieval corpus, but it is not the only possible granularity: in deployments with stronger storage constraints, multiple passages can be grouped into a document-level or cluster-level expert; conversely, finer units can be used when routing precision is more important.

Each expert is lightweight. With the default LoRA configuration, one expert contains 122,880 trainable parameters, corresponding to approximately 481 KiB on disk. The full 27,613-expert bank occupies about 13.08 GiB of disk storage. Importantly, this bank size does not translate into GPU memory pressure during inference. DMoE keeps the expert bank on disk and loads only the top-k selected experts at triggered decoding steps; all inactive experts remain outside GPU memory. Therefore, increasing the expert bank mainly affects disk footprint and index management, while per-step GPU memory depends on the small number of active experts rather than the total number of stored experts.

B.2 Training and Update Cost

DMoE also differs from SFT-LoRA in how updates are performed. SFT-LoRA trains a single adapter on a merged corpus-scale dataset, so adding or modifying knowledge typically requires re-merging the training data and retraining or continuing training a shared adapter. In contrast, DMoE trains one small LoRA expert per knowledge unit independently. Existing expert checkpoints are skipped automatically, so when the corpus grows, only experts for newly added knowledge units need to be trained.

In our current implementation, training one new passage-level expert takes about 10 seconds on a single NVIDIA A100 GPU. Training is embarrassingly parallel because experts are independent: the 27,613-expert bank corresponds to roughly 76.7 A100 GPU-hours under this per-expert timing, and wall-clock time can be reduced by distributing experts across multiple GPUs. Updating obsolete

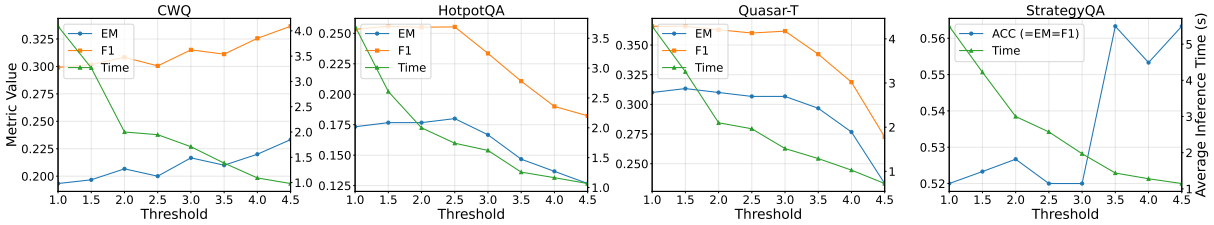

Figure 6: **Impact of triggering thresholds on effectiveness and inference speed.** We vary the TU threshold for expert activation. Larger thresholds reduce triggering frequency and improve inference speed, while downstream performance remains comparatively stable across a broad threshold range.

Table 7: **Impact of retriever choice for routing.** We compare the BM25 router used in the main experiments with SGPT-based dense retrieval. SGPT improves StrategyQA but does not consistently improve the other benchmarks, while using more GPU memory.

Dataset	Router	EM	F1	Avg. Time (s)	Avg. GPU (GB)
HotpotQA	BM25	**0.0600**	0.0813	5.65	**3.05**
HotpotQA	SGPT	**0.0600**	**0.0863**	**5.28**	3.65
StrategyQA	BM25	0.4200	0.4200	8.51	**2.62**
StrategyQA	SGPT	**0.4600**	**0.4600**	**7.52**	3.22
CWQ	BM25	**0.1600**	**0.2388**	5.57	**2.75**
CWQ	SGPT	0.1400	0.2237	**4.95**	3.36
Quasar-T	BM25	**0.1400**	**0.1530**	6.18	**2.91**
Quasar-T	SGPT	0.1200	0.1330	**5.97**	3.51

Table 8: **Impact of LoRA rank and scaling factor.** We vary the expert LoRA configuration (r, α) while keeping the remaining setup fixed. Higher rank or larger scaling does not yield consistent gains across benchmarks.

Dataset	(4,16)	(8,16)	(8,32)	(16,32)
HotpotQA (EM)	**0.0767**	0.0667	0.0733	0.0700
CWQ (EM)	**0.2467**	0.2400	0.2300	0.2433
Quasar-T (EM)	**0.2000**	0.1800	0.1800	0.1767
StrategyQA (Acc.)	0.5267	0.5367	**0.5400**	0.5367

Table 9: **Expert-bank scaling ablation.** We reduce the expert bank to different fractions of the full corpus-aligned bank and report downstream performance. The full bank is not uniformly best, indicating that DMoE is not brittle to the exact expert count.

Dataset	Full	1/2	1/5	1/10	1/100
HotpotQA	0.0767	0.0767	0.0767	**0.0800**	0.0733
StrategyQA	0.5267	0.5400	**0.5433**	0.5300	0.5367
CWQ	**0.2467**	0.2367	0.2433	0.2300	0.2300
Quasar-T	**0.2000**	**0.2000**	0.1900	0.1933	0.1967

knowledge is similarly local. One can delete the corresponding adapter directory and remove the document from the BM25 index, without retraining the backbone or unrelated experts.

B.3 Scalability Trade-off

The main scalability trade-off is therefore not the number of active parameters at inference time, but the granularity of the expert bank. A larger bank provides more fine-grained routing targets but increases disk usage and index size; a smaller bank reduces storage and management overhead but may merge heterogeneous knowledge into the same expert. This trade-off is controllable because knowledge units can be defined at different granularities before expert construction.

We consider two related but distinct dimensions of expert capacity. First, the capacity of each individual expert is controlled by the LoRA rank and scaling factor; as shown in Appendix A.5, this effect is non-monotonic, and the default $(r = 4, \alpha = 16)$ setting is competitive or best on three of the four datasets. Second, the total expert-bank size is controlled by how many knowledge units are represented as separate experts. Table 9 reports the latter ablation.

The results show reasonably stable performance across substantial reductions in bank size. HotpotQA slightly improves at 1/10 of the bank, StrategyQA is best at 1/5, while CWQ and Quasar-T favor the full bank or tie with it. This pattern indi-

cates that DMoE does not require a narrowly tuned expert count to remain functional.

C Prompt Design

This appendix lists the prompt templates used for document augmentation and evaluation. The augmentation prompts are used to construct document-specific parametric knowledge representations, while the evaluation prompt follows the few-shot reasoning format used in our experiments.

C.1 Prompts for Document Augmentation

To construct document-specific parametric knowledge representations, we employ two augmentation techniques: multi-style rewriting and QA generation.

Document Rewriting We utilize rewriting to diversify surface forms while preserving semantic content.

Prompt 1: Document Rewriting

Rewrite the following passage.
While keeping the entities, proper nouns, and key details such as names, locations, and terminology intact, create a new version of the text that expresses the same ideas in a different way. Make sure the revised passage is distinct from the original one, but preserves the core meaning and relevant information.
Passage: passage

QA Pair Generation This prompt is used to generate multiple QA pairs from a passage for downstream training.

Prompt 2: Question-Answer Generation

I will provide a passage of text, and you need to generate three different questions based on the content of this passage. Each question should be answerable using the information provided in the passage. Additionally, please provide an appropriate answer for each question derived from the passage.
You need to generate the question and answer in the following format:
[
{
"question": "What is the capital of France?",
"answer": "Paris",
"full_answer": "The capital of France is Paris."
},
]
This list should have at least three elements. You only need to output this list in the above format.
Passage: passage

C.2 Prompt for HotpotQA

Following prior work such as FLARE(Jiang et al., 2023) and DRAGIN(Su et al., 2024), we adopt a similar few-shot prompting strategy. All compared methods use the same prompt format for a fair evaluation.

We use reasoning exemplars to guide the model toward concise multi-hop answers.

Prompt 3: Few-Shot Prompt for HotpotQA

You should reference the knowledge provided below and combine it with your own knowledge to answer the question. Please follow the format of the example I provided above.
Question: Jeremy Theobald and Christopher Nolan share what profession? Answer: Jeremy Theobald is an actor and producer. Christopher Nolan is a director, producer, and screenwriter. Therefore, they both share the profession of being a producer. So the answer is producer.
Question: Were Lonny and Allure both founded in the 1990s? Answer: Lonny (magazine) was founded in 2009. Allure (magazine) was founded in 1991. Thus, of the two, only Allure was founded in 1990s. So the answer is no.
... [additional examples truncated for brevity] ...
Question: Were Scott Derrickson and Ed Wood of the same nationality?