DualPRAG: Dual-Channel Adaptive Reasoning Control for Parametric Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-augmented generation improves the factual reliability of large language models by providing external evidence in the input prompt, but it does not explicitly control how such evidence is used during intermediate reasoning. We propose **Dual-Channel Parametric RAG** (*DualPRAG*), a latent retrieval-augmented reasoning framework that extends parametric retrieval-augmented generation from knowledge injection to reasoning control. DualPRAG decomposes retrieved evidence into two roles: factual access, implemented through document-conditioned memory slots, and reasoning modulation, implemented through a control vector that routes trainable reasoning primitives. The resulting latent signals are injected into a frozen LLM via gated MLP-input residual updates, enabling lightweight hidden-state modulation without altering the backbone self-attention computation. We evaluate DualPRAG on multi-hop question answering and mathematical reasoning benchmarks across multiple LLM backbones. Results show that DualPRAG achieves effective and efficient reasoning improvements, and ablation studies confirm the contribution of memory slots, reasoning primitives, encoder depth, and the dual-channel injection design. Our code and data are released in `https://anonymous.4open.science/r/DualPRAG-BFD4/`

1 Introduction

Large language models (LLMs) (Dubey et al., 2024; Yang et al., 2025; Guo et al., 2025; Yu et al., 2025) have demonstrated strong capabilities in language understanding, knowledge-intensive generation, and multi-step reasoning. Despite these advances, they still suffer from several limitations when solving knowledge-intensive reasoning tasks. First, their parametric knowledge can become outdated or incomplete (De Cao et al., 2021; Zheng et al., 2023). Second, they are prone to hallucination when the required evidence is not reliably encoded in the model parameters (Shi et al., 2024). Third, it remains challenging for large language models to handle tasks that require integrating multiple pieces of external evidence through intermediate reasoning steps (Ho et al., 2020; Yang et al., 2018).

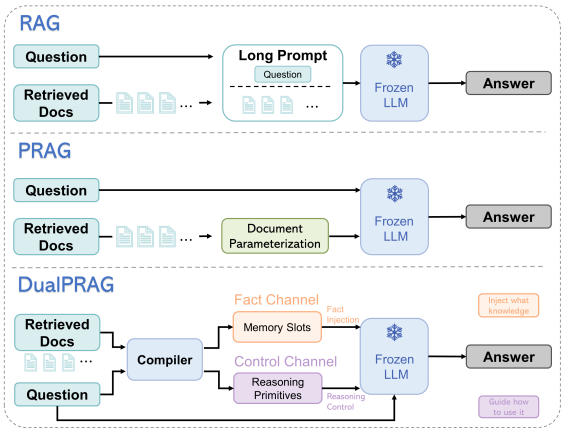

Figure 1: RAG augments the prompt with retrieved documents, while PRAG injects retrieved documents into the model's internal representation or parameters. DualPRAG further introduces a dual-channel mechanism that separates factual access from reasoning control, enabling retrieved evidence to guide both what knowledge is used and how it influences intermediate reasoning.

As shown in Figure 1, retrieval-augmented generation (RAG) (Lewis et al., 2020) has become a widely used solution for these issues by retrieving relevant documents and appending them to the input prompt. This allows the model to condition its generation on external evidence instead of relying solely on internal parametric knowledge. However, prompt-level retrieval augmentation has two important limitations. First, long retrieved passages consume context budget and may introduce irrelevant, redundant, or noisy information (Liu et al., 2024). Second, simply placing retrieved evidence in the prompt does not explicitly ensure that the model fully attends to and properly uses such

evidence (Shi et al., 2024). Parametric retrieval-augmented generation (PRAG) (Su et al., 2025a) alleviates this issue by internalizing retrieved evidence into the model through parameterization, yet it mainly focuses on knowledge injection and still lacks explicit guidance on how the injected evidence should influence intermediate reasoning.

This motivates the central question of this work: *can we inject external knowledge into a frozen LLM while also guiding how the model uses that knowledge during reasoning?* To address this question, we study a latent retrieval-augmented reasoning framework in which retrieved evidence is not merely appended as textual context, but is transformed into compact latent controls that modulate the hidden computation of a frozen language model. This perspective extends parametric retrieval-augmented generation (PRAG) by going beyond knowledge injection alone. The model should not only access external knowledge, but also learn how to use it for reasoning.

We propose DualPRAG, a framework that extends PRAG from knowledge injection to internal reasoning control. The core idea is to decompose the role of retrieved evidence into two complementary channels: *factual access* and *reasoning modulation*. The fact channel encodes retrieved evidence into document-conditioned memory slots, enabling token-level hidden states to access relevant external knowledge. The control channel produces a global control vector that routes a mixture of reasoning primitives, thereby guiding how intermediate representations are transformed. This dual-channel design allows DualPRAG to model not only what knowledge should be injected, but also how the injected knowledge should influence reasoning.

Instead of augmenting only the textual prompt as in standard RAG, DualPRAG performs hidden-state-level retrieval augmentation. Retrieved evidence is compressed by a lightweight compiler, which reuses the frozen LLM embedding layer and applies a trainable encoder to produce latent document representations. These representations are injected into selected decoder layers through a gated MLP-input mechanism. Specifically, DualPRAG adds two independently gated residual updates to the feed-forward sublayer input, corresponding to the fact and control channels. Since the backbone self-attention and feed-forward parameters remain frozen, DualPRAG functions as a lightweight reasoning adapter that modulates the frozen LLM without altering its original computations.

The main contributions of this work are threefold:

- We propose DualPRAG (*Dual-Channel Parametric RAG*), which extends PRAG by enabling both external knowledge injection and internal reasoning control.
- We introduce a gated MLP-input injection mechanism with separate fact and control channels to support factual access and reasoning modulation.
- We evaluate DualPRAG on multiple LLM backbones and benchmark datasets, showing its effectiveness and efficiency through main results and ablation studies.

2 Related Works

2.1 Retrieval-Augmented and Parametric Knowledge Injection

Retrieval-augmented generation (Lewis et al., 2020) was first introduced to improve the factual reliability of language models by retrieving external documents and appending them to the input context. Later studies improved this paradigm through stronger retriever (Glass et al., 2022; Wang et al., 2024; Zhang et al., 2025), better evidence aggregation (Izacard and Grave, 2021; Yu et al., 2024), and Self-reflection (Jiang et al., 2023; Asai et al., 2024). These methods mainly operate at the prompt or input level, where retrieved passages are provided as textual evidence for generation.

More recent works (Su et al., 2025a; Tan et al., 2025; Tang et al., 2025; Su et al., 2025b) move beyond prompt-level retrieval and explores parametric or latent knowledge injection. Instead of directly concatenating retrieved passages (Lewis et al., 2020), these methods compress external evidence into model parameters, hidden representations, or lightweight modules, reducing the dependence on long contexts. PRAG (Su et al., 2025a) first proposes to inject retrieved documents into the feed-forward parameters of LLMs through document parameterization, reducing the cost of long-context RAG. DyPRAG (Tan et al., 2025) improves this paradigm with a lightweight parameter translator that dynamically generates parametric knowledge at test time. PolyPRAG (Su et al., 2025b) further reduces storage and training overhead by representing documents with combinations of shared LoRA adapters through latent routing. DualPRAG

builds on this progression by extending PRAG from knowledge injection to reasoning control, enabling retrieved knowledge to guide both factual access and intermediate reasoning.

2.2 Reasoning Control in Large Language Models

Early reasoning methods (Wei et al., 2022; Hao et al., 2024) mainly rely on prompting strategies, such as generating intermediate reasoning steps before the final answer. Subsequent work improves this paradigm through self-consistency (Wang et al., 2022), problem decomposition (Fu et al., 2021; Zhong et al., 2023), and Self-reflection (Asai et al., 2024), allowing language models to perform more reliable multi-step inference.

Beyond explicit textual reasoning, recent studies (Sinii et al., 2025; Chen et al., 2025; Ye et al., 2026; Li et al., 2026) have explored latent reasoning control through steering vectors, modular controllers, and hidden-state interventions. Bias-only adaptation (Sinii et al., 2025) controls reasoning by training only one lightweight steering vector per layer while freezing the backbone. SEAL (Chen et al., 2025) calibrates CoT reasoning through training-free representation intervention with latent reasoning steering vectors. RISER (Ye et al., 2026) adaptively composes reusable reasoning vectors with a lightweight router to steer model activations for each input. These methods show that reasoning can be controlled not only by generating additional tokens, but also by modulating internal model computation. DualPRAG follows this perspective by introducing a retrieval-conditioned control channel that selects reasoning primitives to regulate how external knowledge affects intermediate reasoning states.

3 Methodology

As shown in Figure 2, DualPRAG mainly consists of dual-channel signal generation § 3.1, followed by fact injection § 3.2.1 and reasoning-control guidance § 3.2.2 within the frozen LLM.

3.1 Dual-Channel Compiler

The compiler first tokenizes x^{enc} and obtains token embeddings from the frozen LLM backbone embedding layer (Emb_ϕ). Let the backbone hidden dimension be H and the encoder dimension be E. Since the compiler does not update the backbone parameters, the embedding outputs are detached and projected from $\mathbb{R}^H$ to $\mathbb{R}^E$ by a learnable input projection $W_{\text{in}} \in \mathbb{R}^{E\times H}$.

$$\mathbf{U} = \text{Emb}_\phi(x^{\text{enc}})W_{\text{in}}^\top \in \mathbb{R}^{T_e\times E} \tag{1}$$

where x^{enc} denotes the natural-language form of the injected knowledge, and T_e is the corresponding token length. A lightweight Transformer encoder then computes contextual evidence representations.

$$\begin{aligned}\mathbf{S} &= \text{TransformerEncoder}_\theta(\mathbf{U}) \\ &= [\mathbf{s}_1, \ldots, \mathbf{s}_{T_e}] \in \mathbb{R}^{T_e\times E}\end{aligned} \tag{2}$$

The proposed method uses attention pooling to aggregate token-level evidence representations. For each token representation $\mathbf{s}_t \in \mathbf{S}$, a scalar pooling score is computed as

$$a_t = \mathbf{W}_{p_{out}}^\top \tanh(\mathbf{W}_{p_{in}} \text{Norm}(\mathbf{s}_t)) \tag{3}$$

where $\mathbf{W}_{p_{out}}$ and $\mathbf{W}_{p_{in}}$ are learnable parameters of the attention-pooling. The normalized attention weight is then

$$\alpha_t = \frac{\exp(a_t)}{\sum_{j=1}^{T_e}\exp(a_j)} \tag{4}$$

and the pooled evidence vector is obtained by

$$\bar{\mathbf{s}} = \sum_{t=1}^{T_e} \alpha_t \mathbf{s}_t \tag{5}$$

The pooled vector is then projected into two latent channels through two separate heads, f_θ and g_θ.

$$\mathbf{z}_D^f = \text{reshape}(f_\theta(\bar{\mathbf{s}})) \in \mathbb{R}^{K\times H} \tag{6}$$

$$\mathbf{z}_D^c = g_\theta(\bar{\mathbf{s}}) \in \mathbb{R}^H \tag{7}$$

where K denotes the number of fact memory slots. The fact memories $\mathbf{z}_D^f$ serve as token-addressable evidence for knowledge injection, while the control vector $\mathbf{z}_D^c$ parameterizes layer-wise routing over reasoning primitives. For more detailed structural information about the Compiler, please refer to Appendix A.1.

3.2 MLP-Input Injection

Let $\mathbf{h}_l^{\text{MLP}_{\text{in}}} \in \mathbb{R}^{T\times H}$ denote the input representation to the MLP submodule of decoder layer l. In a standard pre-norm transformer decoder layer, this representation is obtained after self-attention, residual addition, and the subsequent normalization

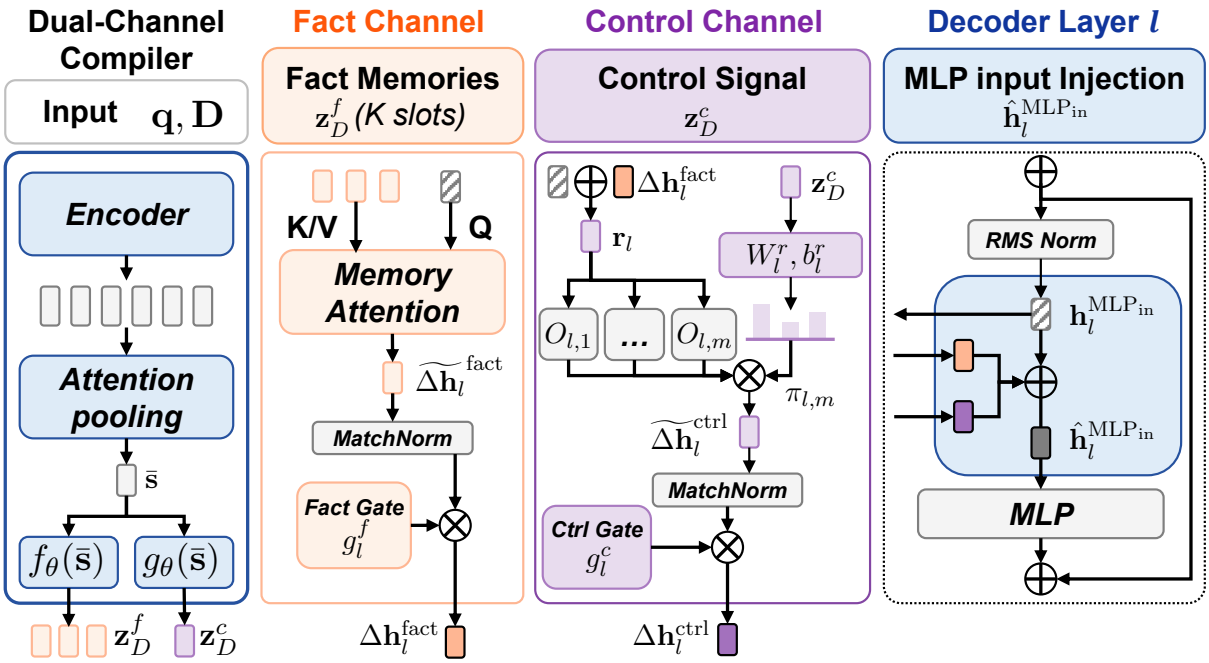

Figure 2: Overview of the proposed DualPRAG architecture. Given a question and retrieved documents, a shared dual-channel compiler encodes the external context into fact memories z_D^f and a control vector z_D^c. At each injected decoder layer, the fact channel performs memory attention over z_D^f to obtain a factual update Δh_l^{fact}, while the control channel uses z_D^c to route among reasoning primitives and produces a control update Δh_l^{ctrl} conditioned on the fact-updated MLP input. The two updates are gated, norm-matched, and added to the original MLP input representation $h_l^{\text{MLP}_{\text{in}}}$, yielding $\hat{h}_l^{\text{MLP}_{\text{in}}}$ for the frozen LLM. Importantly, the self-attention sublayer and the MLP parameters remain unchanged. DualPRAG only modulates the MLP input through lightweight layer-wise adapters.

before the MLP. DualPRAG registers a forward pre-hook on the selected MLP modules and replaces the original MLP input with a modulated representation.

$$\hat{\mathbf{h}}_l^{\text{MLP}_{\text{in}}} = \mathbf{h}_l^{\text{MLP}_{\text{in}}} + \Delta\mathbf{h}_l^{\text{fact}} + \Delta\mathbf{h}_l^{\text{ctrl}} \tag{8}$$

Only $\hat{\mathbf{h}}_l^{\text{MLP}_{\text{in}}}$ is passed to the frozen MLP submodule. The self-attention computation, residual path, and MLP parameters of the backbone decoder layer remain unchanged. Thus, DualPRAG does not overwrite the full decoder hidden state; instead, it performs lightweight, layer-wise modulation at the MLP input.

3.2.1 Fact Channel

The fact channel injects document-grounded evidence into the MLP input representation through memory attention over the fact slots. Let $\mathbf{h}_l^{\text{MLP}_{\text{in}}} \in \mathbb{R}^{T\times H}$ denote the original MLP input at decoder layer l, and let $\mathbf{z}_D^f \in \mathbb{R}^{K\times H}$ denote the fact memories produced by the compiler. For each injected layer, the fact memories are projected into layer-specific keys and values:

$$\mathbf{K}_l^f = \mathbf{z}_D^f W_l^K, \qquad \mathbf{V}_l^f = \mathbf{z}_D^f W_l^V \tag{9}$$

where $W_l^K, W_l^V \in \mathbb{R}^{H\times H}$.

The raw fact update is computed by attending from the MLP input representation to the fact memory slots.

$$\widetilde{\Delta\mathbf{h}}_l^{\text{fact}} = \text{softmax}\left(\frac{\mathbf{h}_l^{\text{MLP}_{\text{in}}}(\mathbf{K}_l^f)^\top}{\sqrt{H}}\right)\mathbf{V}_l^f \tag{10}$$

To stabilize the magnitude of the injected signal, the update is matched to the token-wise norm of the original MLP input.

$$\text{MatchNorm}(\Delta, \mathbf{h})_t = \Delta_t \cdot \frac{\|\mathbf{h}_t\|_2}{\|\Delta_t\|_2 + \epsilon} \tag{11}$$

where t indexes tokens. The final fact update added to the MLP input is then obtained by applying a learnable layer-wise gate g_l^f.

$$\Delta\mathbf{h}_l^{\text{fact}} = g_l^f \cdot \text{MatchNorm}\left(\widetilde{\Delta\mathbf{h}}_l^{\text{fact}}, \mathbf{h}_l^{\text{MLP}_{\text{in}}}\right) \tag{12}$$

The gate is initialized with a small value, e.g., $g_l^f \approx 0.01$, so that the frozen backbone is only weakly perturbed at the beginning of training.

3.2.2 Control Channel

The control channel provides a second residual update that modulates the MLP input according to document-conditioned reasoning signals. Each injected layer maintains a library of M trainable reasoning primitives, the structural details of the reasoning primitives are provided in Appendix A.2.

$$\mathcal{O}_l = \{O_{l,1}, \ldots, O_{l,M}\} \tag{13}$$

where each primitive maps a hidden representation in $\mathbb{R}^H$ to another representation in the same space.

Rather than operating on the original MLP input alone, the control channel is conditioned on the fact-updated representation.

$$\mathbf{r}_l = \mathbf{h}_l^{\mathrm{MLP_{in}}} + \Delta\mathbf{h}_l^{\mathrm{fact}} \tag{14}$$

A layer-specific router maps the control vector $\mathbf{z}_D^c$ to a distribution over reasoning primitives.

$$\boldsymbol{\pi}_l = \mathrm{softmax}\left(W_l^r \mathbf{z}_D^c + \mathbf{b}_l^r\right), \sum_{m=1}^{M} \pi_{l,m} = 1 \tag{15}$$

where, $W_l^r \in \mathbb{R}^{M\times H}$, $\mathbf{b}_l^r \in \mathbb{R}^M$.The raw control update is computed as a mixture of primitive outputs.

$$\widetilde{\Delta\mathbf{h}}_l^{\mathrm{ctrl}} = \sum_{m=1}^{M} \pi_{l,m} O_{l,m}(\mathbf{r}_l) \tag{16}$$

As in the fact channel, the control update is norm-matched to the reference MLP input and modulated by a learnable layer-wise gate.

$$\Delta\mathbf{h}_l^{\mathrm{ctrl}} = g_l^c \cdot \mathrm{MatchNorm}\left(\widetilde{\Delta\mathbf{h}}_l^{\mathrm{ctrl}}, \mathbf{h}_l^{\mathrm{MLP_{in}}}\right) \tag{17}$$

As in the fact channel, the control update is norm-matched to the reference MLP input and modulated by a learnable layer-wise gate, which is initialized to a small value, e.g., $g_l^c \approx 0.01$, following the same initialization strategy as the fact gate g_l^f. The final modulated MLP input is therefore

$$\hat{\mathbf{h}}_l^{\mathrm{MLP_{in}}} = \mathbf{h}_l^{\mathrm{MLP_{in}}} + \Delta\mathbf{h}_l^{\mathrm{fact}} + \Delta\mathbf{h}_l^{\mathrm{ctrl}} \tag{18}$$

This formulation decouples document-conditioned factual access from document-conditioned reasoning control while keeping both mechanisms residual, gated, and parameter efficient.

3.3 Training Objective

During training, the backbone LLM receives the generation-side sequence x^{gen}, which consists of the target question followed by the gold answer.

$$x^{\mathrm{gen}} = [\texttt{Question:}]\ q\ [\texttt{Answer:}]\ y \tag{19}$$

The training loss is computed only on the gold answer tokens, while the question tokens are masked out.

$$\mathcal{L}_{\mathrm{ans}} = -\sum_{t=1}^{|y|} \log p_{\phi,\theta}\left(y_t \mid y_{<t}, x^{\mathrm{gen}}, x^{\mathrm{enc}}\right) \tag{20}$$

The training objective can additionally include a diversity regularizer over the reasoning primitives. For layer l, let

$$\mathbf{v}_{l,m} = O_{l,m}(\mathbf{r}_l) \tag{21}$$

denote the output of primitive m given the fact-conditioned representation $\mathbf{r}_l$. The diversity regularizer penalizes excessive similarity among different primitive outputs.

$$\mathcal{L}_{\mathrm{div}} = \frac{1}{|\Omega|} \sum_{(l,m,m')\in\Omega} \cos\left(\mathbf{v}_{l,m}, \mathbf{v}_{l,m'}\right), m \neq m' \tag{22}$$

where Ω denotes the set of compared primitive pairs. This term encourages different primitives to specialize in distinct transformations rather than collapsing to similar updates. The final training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda_{\mathrm{div}}\mathcal{L}_{\mathrm{div}} \tag{23}$$

where λ_{div} controls the strength of the diversity regularization. Only the shared encoder, fact projection head, control projection head, layer injectors, routers, gates, and primitives are optimized. The backbone LLM is frozen throughout training.

3.4 Inference

At inference time, the system first applies BM25 (Robertson and Zaragoza, 2009) retrieval over the document collection and selects the top-k retrieved documents as external evidence. The retrieved documents are concatenated with the question to construct the encoder input x^{enc}, which is processed by the compiler to produce the fact memories and control vector used for MLP-input injection.

Base LLM	Method	2WQA					HotPotQA		
		Compare	Bridge	Inference	Compose	Total	Compare	Bridge	Total
LLaMA3.2-1B	Base	42.89	24.17	16.91	7.87	22.52	40.26	13.25	18.79
	Standard RAG	41.23	26.78	22.51	10.21	23.12	42.46	21.38	27.14
	PRAG	50.20	24.34	19.11	8.24	27.73	40.90	13.65	21.50
	PRAG+	40.50	31.30	22.85	9.77	30.30	41.55	22.56	28.31
	DyPRAG	51.25	48.15	17.35	7.54	25.31	43.90	14.05	19.97
	DyPRAG+	52.13	46.19	22.54	**12.60**	29.18	43.78	22.05	26.58
	DualPRAG	48.21	49.34	18.85	11.94	29.84	58.37	17.89	25.54
	DualPRAG+	**62.39**	**55.83**	**27.59**	11.41	**35.46**	**62.36**	**30.46**	**36.49**
Qwen2.5-1.5B	Base	45.74	39.06	17.04	7.27	26.87	39.46	12.18	17.76
	Standard RAG	38.75	38.84	11.87	5.68	24.31	37.13	16.19	20.73
	PRAG	44.96	43.96	19.29	11.14	27.55	40.42	13.27	18.42
	PRAG+	40.50	44.00	16.30	8.17	27.49	36.49	18.86	23.10
	DyPRAG	43.03	47.20	17.04	8.55	26.46	41.39	13.72	19.67
	DyPRAG+	35.83	44.89	14.81	8.64	25.18	41.25	21.56	27.57
	DualPRAG	51.75	52.01	17.34	8.49	29.83	61.43	16.30	24.83
	DualPRAG+	**54.04**	**55.01**	**22.79**	**13.23**	**33.51**	**63.32**	**30.57**	**36.76**
LLaMA3-8B	Base	54.90	55.20	24.59	14.43	33.02	45.63	19.00	21.29
	Standard RAG	58.43	47.77	19.20	11.07	34.55	42.10	19.68	24.23
	PRAG	57.78	58.93	27.61	19.17	39.19	65.88	33.68	38.08
	PRAG+	60.13	56.69	32.71	20.91	40.55	68.22	39.41	44.84
	DyPRAG	57.39	56.43	25.33	18.88	37.80	58.59	24.85	28.56
	DyPRAG+	66.00	59.46	**35.78**	**26.90**	**50.24**	57.93	33.37	38.35
	DualPRAG	63.12	63.98	24.84	14.70	38.55	68.94	26.22	34.29
	DualPRAG+	**69.27**	**72.33**	32.32	22.94	46.24	**72.18**	**45.42**	**50.48**

Table 1: Performance comparison on multi-hop QA using three frozen backbone models. All methods use BM25 top-3 retrieval, and results are reported as F1 scores (%).

To distinguish purely latent injection from its combination with prompt-level evidence, we consider two inference variants: **DualPRAG** and **DualPRAG+**. In DualPRAG, the LLM receives only the question q as its prompt, while the retrieved documents influence generation solely through the latent controller. In DualPRAG+, the top-k retrieved documents are additionally serialized and prepended to the LLM prompt together with the question. Thus, DualPRAG evaluates purely latent document injection, whereas DualPRAG+ combines latent injection with conventional prompt-level document conditioning.

4 Experiments

4.1 Experimental Setup

We conduct experiments on three datasets covering two reasoning tasks: multi-hop question answering and mathematical reasoning. We use Llama3.2-1B-Instruct[1] (Meta AI, 2024), Qwen2.5-1.5B-Instruct[2] (Qwen et al., 2025), and Llama3-8B-Instruct[3] (Dubey et al., 2024) as frozen LLM backbones. Only the DualPRAG controller is trained, including the compiler, fact channel, control channel, gates, and reasoning primitives. For retrieval, we use BM25 (Robertson and Zaragoza, 2009) to obtain external evidence for each input question. We evaluate multi-hop question answering on 2WikiMultihopQA (2WQA) (Ho et al., 2020) and HotpotQA (Yang et al., 2018), and mathematical reasoning on GSM8K-Aug (Deng et al., 2023). Details of the datasets, metrics, and hyperparameter settings are provided in Appendix B.2.

4.2 Main Results

We evaluated the method on two tasks: multi-hop question answering and mathematical reasoning. Detailed descriptions of the compared methods are provided in Appendix B.3.

[1] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct

[2] https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct

[3] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

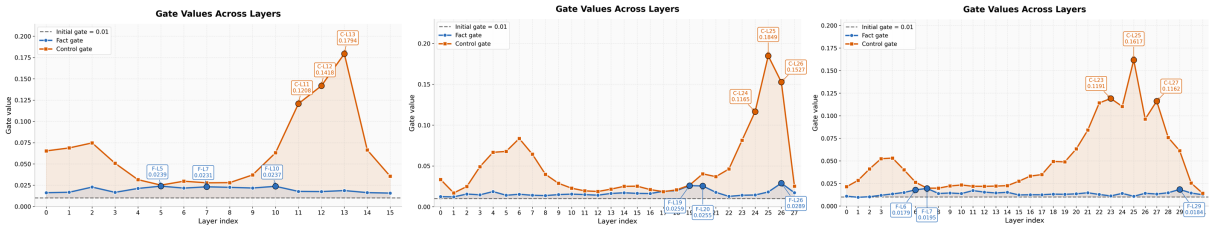

Figure 3: Layer-wise fact and control gate values after training on 2WikiMultihopQA for three backbone models: LLaMA3.2-1B, Qwen2.5-1.5B, and LLaMA3-8B. The dashed line denotes the initial gate value of 0.01.

Base LLM	Method	GSM8K-Aug
LLaMA3.2-1B	Base	2.6
	CoT	4.4
	DualPRAG	34.8
Qwen2.5-1.5B	Base	10.1
	CoT	14.5
	DualPRAG	44.5
LLaMA3-8B	Base	8.2
	CoT	14.6
	DualPRAG	59.6

Table 2: Performance on the mathematical reasoning taskt. Results are reported as exact-match accuracy (%) based on the extracted final numerical answer.

	2WQA			
Setting	**Compare**	**Bridge**	**Inference**	**Compose**
A. Number of memory slots (K)				
$K = 1$	50.17(61.72)	48.38(57.14)	18.27(28.88)	10.30(12.97)
$\mathbf{K = 2}$	48.21(62.39)	49.34(55.83)	18.85(27.59)	11.94(11.41)
$K = 4$	48.65(62.70)	49.31(56.87)	18.28(29.40)	10.88(11.58)
$K = 8$	45.80(63.21)	45.73(53.33)	19.70(27.36)	9.31(11.32)
B. Number of reasoning primitives (M)				
$M = 1$	41.36(60.48)	48.09(52.77)	15.14(27.00)	8.41(9.17)
$M = 2$	44.68(59.06)	53.06(57.16)	19.92(27.12)	10.69(13.61)
$\mathbf{M = 4}$	48.21(62.39)	49.34(55.83)	18.85(27.59)	11.94(11.41)
C. Number of encoder layers (L_e)				
$L_e = 1$	50.01(60.11)	46.09(58.45)	18.61(27.62)	10.52(12.85)
$\mathbf{L_e = 2}$	48.21(62.39)	49.34(55.83)	18.85(27.59)	11.94(11.41)
$L_e = 4$	47.36(62.99)	51.38(61.37)	21.41(29.91)	9.38(13.48)

Table 3: Ablation study of controller capacity on 2WikiMultihopQA using LLaMA3.2-1B. We vary the number of memory slots K, reasoning primitives M, and encoder layers L_e while keeping the remaining hyperparameters fixed. Results of DualPRAG(DualPRAG+) are reported as F1 by question type.

4.2.1 Multihop QA

Table 1 reports the performance on multi-hop QA under the setting of retrieving the top-3 documents. Overall, DualPRAG achieves strong results across different backbone models and question types. Compared with standard RAG, DualPRAG better utilizes retrieved evidence by injecting it into the model's hidden computation rather than relying only on prompt-level concatenation. Compared with PRAG-style baselines, DualPRAG further improves reasoning by introducing a control channel that guides how the injected evidence is used.

The combined DualPRAG+ generally obtains the best performance, suggesting that latent evidence injection and textual evidence are complementary, which is consistent with the findings of Tang et al. (2025). These results indicate that jointly modeling factual access and reasoning control is beneficial for multi-hop question answering.

4.2.2 Mathematical Reasoning

We evaluate the mathematical reasoning task on the GSM8K-Aug dataset, and the results are shown in Table 2. DualPRAG consistently outperforms both the base model and the CoT baseline across all backbone models, showing that the proposed framework is effective not only for multi-hop QA but also for mathematical reasoning.

The improvements suggest that retrieved reasoning examples can provide useful latent guidance when encoded through the DualPRAG controller. Instead of simply prompting the model to produce reasoning steps, DualPRAG injects retrieved examples into the hidden computation and uses the control channel to guide how this information influences intermediate reasoning. These results further confirm the generality of DualPRAG across different reasoning tasks and model backbones.

4.3 Layer-wise Gate Analysis

In addition to the main QA results, we analyze the learned fact and control gate values across decoder layers. Figure 3 shows the layer-wise values of the fact gate g_l^f and control gate g_l^c for LLaMA3.2-1B-Instruct, Qwen2.5-1.5B-Instruct, and LLaMA3-8B-Instruct. Across all three backbones, the control

Base LLM	Method	2WQA					HotPotQA		
		Compare	Bridge	Inference	Compose	Total	Compare	Bridge	Total
LLaMA3.2-1B	DualPRAG(+)	48.21(62.39)	49.34(55.83)	18.85(27.59)	11.94(11.41)	29.84(35.46)	58.37(62.36)	17.89(30.46)	25.54(36.49)
	w/o Reasoning	42.13(47.67)	47.78(54.29)	16.93(24.07)	7.37(9.81)	25.91(30.56)	52.08(55.17)	17.77(28.90)	24.25(33.87)
Qwen2.5-1.5B	DualPRAG(+)	51.75(54.04)	52.01(55.01)	17.34(22.79)	8.49(13.23)	29.66(33.51)	61.43(63.32)	16.30(30.57)	24.83(36.76)
	w/o Reasoning	43.76(50.13)	50.28(47.22)	14.18(18.10)	9.99(6.71)	27.64(27.52)	56.21(49.22)	15.26(23.86)	23.00(28.65)
LLaMA3-8B	DualPRAG(+)	63.12(69.27)	63.98(72.33)	24.84(32.32)	14.70(22.94)	38.55(46.24)	68.94(72.18)	26.22(45.42)	34.29(50.48)
	w/o Reasoning	59.84(69.11)	59.61(69.64)	23.36(30.06)	12.99(21.78)	35.89(44.83)	65.11(51.77)	25.56(33.78)	33.03(37.81)

Table 4: Ablation study results of DualPRAG(DualPRAG+) on the reasoning channel across different backbone LLMs and multi-hop QA benchmarks.

gates become substantially larger than the fact gates after training, suggesting that the model relies more strongly on the control channel to modulate layer-wise reasoning dynamics. The strongest control activations tend to appear in middle-to-late or late decoder layers, while the fact gates remain comparatively small and stable across layers. This pattern indicates that DualPRAG preserves the factual injection pathway as a lightweight residual signal, while using the reasoning-primitives channel to provide more pronounced layer-specific modulation.

4.4 Ablation Study

We conduct ablation studies to examine the effects of controller capacity and the reasoning control channel.

4.4.1 Effect of Controller Capacity

We further investigate the sensitivity of our framework to three key controller hyperparameters: the number of memory slots K, the number of reasoning primitives M, and the number of encoder layers L_e. In each study, we vary one hyperparameter while keeping the remaining settings fixed. Table 3 reports the ablation results of LLaMA3.2-1B on 2WikiMultihopQA.

The results show that larger controller capacity does not always lead to better performance. A moderate configuration generally provides a better balance between representation capacity and computational efficiency. Based on these observations, we use $K = 2$, $M = 4$, and $L_e = 2$ as the default setting. Additional ablation results on Qwen2.5-1.5B and LLaMA3-8B, together with further analysis, are provided in Appendix C.

4.4.2 Ablation on Reasoning Channel

To examine the contribution of the reasoning channel, we compare the full model with a variant that removes the control-based reasoning update and only retains the fact injection pathway. As shown in Table 4, removing the reasoning channel consistently weakens the model across different backbone LLMs and datasets. This indicates that fact injection alone is insufficient for fully exploiting retrieved evidence, especially when the task requires comparison, bridging, or multi-step reasoning. The reasoning channel provides an additional mechanism for transforming and organizing injected factual information, enabling the model to better adapt the external evidence to the reasoning process. These results suggest that the reasoning channel is an important component of our framework rather than a redundant auxiliary module.

5 Conslusion

In this work, we proposed DualPRAG, a dual-channel framework for retrieval-augmented reasoning with frozen large language models. Unlike standard RAG, which augments the input prompt with retrieved evidence, and PRAG-style methods, which primarily internalize external knowledge, DualPRAG jointly models factual access and reasoning control. By separating these two roles, DualPRAG enables retrieved evidence to not only enter the model's hidden computation, but also guide how it influences intermediate reasoning states.

Experiments on multi-hop question answering and mathematical reasoning benchmarks show that DualPRAG consistently outperforms standard RAG and PRAG-style baselines across different backbone models. These results indicate that effective retrieval augmentation requires both access to external knowledge and mechanisms for controlling its use during reasoning. Future work will explore DualPRAG on more challenging reasoning tasks. We hope this work inspires more efficient and robust reasoning paradigms for knowledge-intensive reasoning.

Limitations

Although DualPRAG shows promising results, this work has two main limitations. First, DualPRAG still depends on the quality of retrieved evidence. Since the controller generates both fact and control signals from the retrieved documents, irrelevant or missing evidence may limit the effectiveness of latent injection and reasoning modulation. Although DualPRAG reduces the dependence on prompt-level evidence, it does not directly address retrieval failures. Future work can combine the proposed controller with stronger retrievers, rerankers, or adaptive retrieval mechanisms.

Another, the current implementation introduces additional trainable controller modules, including the compiler, projection heads, gates, and reasoning primitives. Although these modules are lightweight compared with full-model fine-tuning and do not require document-specific parameter storage, they still introduce extra training and inference computation. More efficient controller architectures and dynamic layer selection strategies could further reduce this overhead.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pages 9112–9141.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025. Seal: Steerable reasoning calibration of large language models for free. *arXiv preprint arXiv:2504.07986*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6491–6506.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2023. Implicit chain of thought reasoning via knowledge distillation. *Preprint*, arXiv:2311.01460.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Ruiliu Fu, Han Wang, Xuejun Zhang, Jun Zhou, and Yonghong Yan. 2021. Decomposing complex questions makes multi-hop qa easier and more interpretable. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 169–180.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pages 874–880.

Zhengbao Jiang, Frank 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, Singapore. Association for Computational Linguistics.

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.

Li Li, Ziyi Wang, Yongliang Wu, Jianfei Cai, and Xu Yang. 2026. Cot vectors: Transferring and probing the reasoning mechanisms of LLMs. In *The Fourteenth International Conference on Learning Representations*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Meta AI. 2024. Llama 3.2 model card. https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct. Accessed: 2026-05.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791, Mexico City, Mexico. Association for Computational Linguistics.

Viacheslav Sinii, Alexey Gorbatovski, Artem Cherepanov, Boris Shaposhnikov, Nikita Balagansky, and Daniil Gavrilov. 2025. Steering llm reasoning through bias-only adaptation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9213–9222.

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

Zhan Su, Fengran Mo, Jinghan Zhang, Yuchen Hui, Jiaao Sun, and Jian-yun Nie. 2025b. Parametric retrieval-augmented generation using latent routing of lora adapters. *arXiv preprint arXiv:2511.17044*.

Yuqiao Tan, Shizhu He, Huanxuan Liao, Jun Zhao, and Kang Liu. 2025. Dynamic parametric retrieval augmented generation for test-time knowledge enhancement. *arXiv preprint arXiv:2503.23895*.

Minghao Tang, Shiyu Ni, Jingtong Wu, Zengxin Han, and Keping Bi. 2025. The role of parametric injection-a systematic study of parametric retrieval-augmented generation. *arXiv preprint arXiv:2510.12668*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yuhao Wang, Ruiyang Ren, Junyi Li, Wayne Xin Zhao, Jing Liu, and Ji-Rong Wen. 2024. Rear: A relevance-aware retrieval-augmented framework for open-domain question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5613–5626.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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, Brussels, Belgium. Association for Computational Linguistics.

Wencheng Ye, Xiaoyang Yuan, Yi Bin, Pengpeng Zeng, Hengyu Jin, Liang Peng, and Heng Tao Shen. 2026. Riser: Orchestrating latent reasoning skills for adaptive activation steering. *arXiv preprint arXiv:2601.09269*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Peixin Cao, Kaixin Ma, Jian Li, Hongwei Wang, and Dong Yu. 2024. Chain-of-note: Enhancing robustness in retrieval-augmented language models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 14672–14685.

Zhange Zhang, Yuqing Ma, Yulong Wang, Shan He, Tianbo Wang, Siqi He, Jiakai Wang, and Xianglong Liu. 2025. Lexical diversity-aware relevance assessment for retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27758–27781.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876.

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702, Singapore. Association for Computational Linguistics.

A Structural Details

In this section, we provide additional implementation-level structural details of the DualPRAG controller. The mathematical definitions of the compiler, fact channel, and control channel are provided in the main text. Here, we focus on the internal module structures, including the shared encoder, projection heads, and reasoning primitives. Unless otherwise specified, we follow the notation defined in the main text.

A.1 Compiler

The compiler follows the dual-channel formulation described in Section 3.1. It consists of a shared encoder and two projection heads. The shared encoder produces the pooled evidence representation $\bar{\mathbf{s}}$, while the fact head and control head project $\bar{\mathbf{s}}$ into the fact memories $\mathbf{z}_D^f$ and the control vector $\mathbf{z}_D^c$, respectively.

Shared Encoder. The shared encoder reuses the frozen LLM embedding layer to embed the encoder-side input x^{enc}. The embedding outputs are detached so that gradients are not propagated into the backbone embedding parameters. If the LLM hidden size H differs from the compiler encoder dimension E, a learnable linear projection W_{in} maps the embeddings from $\mathbb{R}^H$ to $\mathbb{R}^E$; otherwise, this projection is implemented as an identity mapping.

The encoder-side input x^{enc} is constructed according to the task format. For multi-hop question answering, x^{enc} consists of the retrieved documents and the target question. Each retrieved document is assigned an explicit document identifier, allowing the encoder to distinguish document boundaries. For mathematical reasoning, x^{enc} is constructed from a retrieved example, including its question, reasoning steps, and answer, followed by the target question. This format enables the encoder to extract reasoning-relevant information from the retrieved example conditioned on the current question.

x^{enc} FOR MULTI-HOP QA

```
Documents:
[Doc 1] {retrieved document 1}
[Doc 2] {retrieved document 2}
...
[Doc k] {retrieved document k}
Question: {question}
```

x^{enc} FOR MATHEMATICAL REASONING

```
Question: {retrieved_question}
Steps: {retrieved_reasoning_steps}
Answer: {retrieved_answer}
Question: {question}
Steps:
```

The projected sequence is processed by a lightweight Transformer encoder. Each encoder layer uses multi-head self-attention, a feed-forward dimension of $4E$, GELU activation, dropout, and pre-normalization. A final layer normalization is applied to the encoder output before attention pooling. The pooled representation $\bar{\mathbf{s}}$ is then shared by both projection heads.

Fact Head. The fact head f_θ is implemented as a two-layer MLP. It first maps the pooled representation $\bar{\mathbf{s}}$ to an intermediate vector of dimension E, applies GELU activation and dropout, and then projects the result to $K \times H$ dimensions. The output is reshaped into K fact memory slots.

$$\mathbf{z}_D^f \in \mathbb{R}^{K \times H} \tag{24}$$

These memory slots serve as token-addressable latent evidence for the fact channel.

Control Head. The control head g_θ has the same two-layer MLP structure as the fact head, but its output dimension is H rather than $K \times H$. It maps the pooled representation $\bar{\mathbf{s}}$ into the global control vector.

$$\mathbf{z}_D^c \in \mathbb{R}^H \tag{25}$$

This vector is used by the layer-wise router to compute the mixture weights over reasoning primitives.

A.2 Reasoning Primitives

Each injected decoder layer l maintains a set of M trainable reasoning primitives,

$$\mathcal{O}_l = \{O_{l,1}, \ldots, O_{l,M}\} \tag{26}$$

Each primitive is a lightweight gated bottleneck MLP that maps a hidden representation in $\mathbb{R}^H$ back

to $\mathbb{R}^H$. Following the main text, the primitives operate on the fact-conditioned representation

$$\mathbf{r}_l = \mathbf{h}_l^{\text{MLP-in}} + \Delta\mathbf{h}_l^{\text{fact}} \tag{27}$$

For each primitive $O_{l,m}$, the input is first normalized.

$$\bar{\mathbf{r}}_l = \text{LayerNorm}(\mathbf{r}_l) \tag{28}$$

The primitive then applies two parallel projections. The first branch produces a transformed hidden feature.

$$\mathbf{a}_{l,m} = \text{GELU}\left(W_{l,m}^{\text{up}}\bar{\mathbf{r}}_l\right), \tag{29}$$

while the second branch produces a multiplicative gate.

$$\mathbf{b}_{l,m} = \sigma\left(W_{l,m}^{\text{gate}}\bar{\mathbf{r}}_l\right) \tag{30}$$

The two branches are combined by element-wise multiplication and projected back to the LLM hidden dimension.

$$O_{l,m}(\mathbf{r}_l) = \text{Dropout}\left(W_{l,m}^{\text{down}}\left(\mathbf{a}_{l,m} \odot \mathbf{b}_{l,m}\right)\right) \tag{31}$$

The bottleneck dimension of each primitive is set to

$$R = \max(128, H/4) \tag{32}$$

The layer-wise router uses the control vector $\mathbf{z}_D^c$ to compute the mixture weights over primitives.

$$\boldsymbol{\pi}_l = \text{softmax}\left(W_l^r\mathbf{z}_D^c + \mathbf{b}_l^r\right) \tag{33}$$

The raw control update is then obtained as a weighted mixture of primitive outputs.

$$\widetilde{\Delta\mathbf{h}}_l^{\text{ctrl}} = \sum_{m=1}^{M} \pi_{l,m} O_{l,m}(\mathbf{r}_l) \tag{34}$$

As described in the main text, this raw update is further norm-matched and modulated by the control gate before being added to the MLP input. The reasoning primitives therefore provide a compact set of trainable transformations, while the router selects their mixture according to the retrieved evidence encoded in $\mathbf{z}_D^c$.

B Experiment Details

B.1 Implementation Details

All hyperparameters used in the experiments are shown in Table 5. Unless otherwise specified, the backbone model is frozen during training, and only the DualPRAG controller modules are optimized.

B.2 Dataset and Metric Details

We evaluate DualPRAG on both multi-hop question answering and mathematical reasoning tasks. For multi-hop question answering, we use 2WikiMultihopQA (Ho et al., 2020) and HotpotQA (Yang et al., 2018) as benchmarks. For mathematical reasoning, we use GSM8K-Aug (Deng et al., 2023) as the benchmark.

QA Datasets. The 2WikiMultihopQA dataset consists of four question types:

- **Compare**: questions requiring comparison between two or more entities with respect to an attribute.
- **Bridge**: questions requiring the identification of an intermediate entity that connects multiple evidence pieces.
- **Inference**: questions requiring an intermediate inference before the final evidence can be used.
- **Compose**: questions requiring two sequential sub-questions, where the first answer constrains the second step.

These question types provide a useful diagnostic setting for examining whether latent document injection improves factual comparison, entity bridging, implicit inference, and multi-step compositional reasoning.

HotpotQA is another multi-hop question answering benchmark. Following its standard setting, we consider two major question types: **bridge** and **comparison**. Bridge questions require identifying an intermediate entity that links the question to the final answer, while comparison questions require comparing two entities according to a specific property or relation. These two categories are closely related to the bridge and comparison types in 2WikiMultihopQA, and therefore provide an additional testbed for evaluating the robustness of DualPRAG on evidence-grounded multi-hop reasoning.

Mathematical Reasoning Dataset. For mathematical reasoning, we use GSM8K-Aug, a mathematical reasoning dataset expanded from GSM8K (Cobbe et al., 2021) by Deng et al. (2023).

Metrics. For multi-hop question answering, we report token-level F1-score between the predicted answer and the gold answer, following standard

Category	Hyperparameter	Value
Training	Epochs	4
	Batch size	1
	Gradient accumulation steps	4
	Learning rate	1×10^{-4}
	Warmup ratio	0.1
	Weight decay	0.01
	Max gradient norm	1.0
	Random seed	42
Input and retrieval	LLM maximum length	512
	Encoder maximum length	512
	Retrieved evidence budget	Top-3 documents
Compiler and controller	Encoder dimension E	512
	Encoder layers	2
	Encoder attention heads	8
	Fact memory slots K	2
	Reasoning primitives M	4
	Dropout	0.1
	Diversity weight λ_{div}	0.01
Injection	Fact gate initialization g_l^f	0.01
	Control gate initialization g_l^c	0.01

Table 5: Main hyperparameters used in the default DualPRAG experiments.

evaluation practice. For mathematical reasoning, we report accuracy, where a prediction is considered correct only if the extracted final answer exactly matches the gold numerical answer.

B.3 Other Method Details

For the multi-hop question answering task, we compare DualPRAG with several retrieval-augmented baselines:

- **Base**: The frozen backbone LLM answers the question without retrieval, relying solely on its internal parametric knowledge.
- **Standard RAG**: Retrieved documents are appended to the input context to provide factual evidence for generation.
- **PRAG** (**PRAG+**): PRAG injects retrieved knowledge through document parameterization, while PRAG+ further combines parametric injection with prompt-level retrieved evidence.
- **DyPRAG** (**DyPRAG+**): DyPRAG dynamically generates parametric knowledge from retrieved documents, while DyPRAG+ additionally uses retrieved documents in the textual prompt.

We further compare the parameter and storage costs of PRAG-style methods and DualPRAG. As shown in Table 6, PRAG incurs the largest document-level storage overhead because it maintains a separate LoRA adapter for each document. DyPRAG reduces the persistent storage cost by learning a hypernetwork to generate document-specific parameters, but its offline training still depends on document-parameter pairs. By contrast, DualPRAG does not maintain document-specific parameters. It uses a shared controller to produce fact and control updates from retrieved documents online, making its persistent storage cost independent of the corpus size.

For the mathematical reasoning task, we evaluate two input settings: a direct-answer baseline and a retrieval-augmented chain-of-thought setting. The input formats used by **Base** and **CoT** are shown in Appendix B.4. In both settings, the model prediction is evaluated only by comparing the extracted final numerical answer with the gold answer.

Method	Doc-specific Params.	Trainable parameter cost	Persistent storage cost	Corpus-size dependent
PRAG	Yes	$O(2drL\|\mathcal{T}\|)$	$O(2drL\|\mathcal{T}\|)$	Yes
DyPRAG	Yes	$O(2drL\|\mathcal{T}\| + F_\phi)$	$O(F_\phi)$	No for storage
DualPRAG	No	$O(P_{\text{enc}} + P_f + P_c + LMP_{\text{prim}})$	$O(P_{\text{enc}} + P_f + P_c + LMP_{\text{prim}})$	No

Table 6: Computational and memory cost comparison between PRAG-style baselines and DualPRAG. Here, d is the input/output dimension of LoRA weights, r is the LoRA rank, L is the number of Transformer layers, $|\mathcal{T}|$ is the number of documents, and F_ϕ denotes the DyPRAG hypernetwork. For DualPRAG, P_{enc}, P_f, P_c, and P_{prim} denote the shared encoder, fact projection head, control projection head, and one reasoning primitive, respectively.

B.4 Prompt Format

Base Prompt. In the **Base** setting, the model is directly asked to solve the target question and output only the final numerical answer.

BASE PROMPT

```
Solve the following math problem. Return
only the final number, without explanation
or units.

Question: {question}
Answer:
```

Retrieval-Augmented CoT Prompt. In the **CoT** setting, we first retrieve one similar solved example using the question as the retrieval query. The retrieved example contains a question, reasoning steps, and the final answer. The model is then prompted to solve the target question by following the reasoning style of the retrieved example. During evaluation, only the numerical value after `Answer:` is used to compute accuracy.

COT PROMPT WITH RETRIEVED EXAMPLE

```
You are given one solved math example
retrieved from a similar problem. Use the
example as a reference for the reasoning
style, but solve the target question
independently.

Your output must follow this format
exactly:
Steps:
1. ...
2. ...
Answer: <final numerical answer>

Retrieved Example:
Question: {retrieved_question}
Steps:
{retrieved_steps}
Answer: {retrieved_answer}

Target Question:
Question: {question}
Steps:
```

	2WQA			
Setting	**Compare**	**Bridge**	**Inference**	**Compose**
A. Number of memory slots (K)				
$K = 1$	45.96(55.02)	53.42(51.17)	15.13(26.05)	9.59(15.87)
$\mathbf{K = 2}$	51.75(54.04)	52.01(55.01)	17.34(22.79)	8.49(13.23)
$K = 4$	43.55(52.77)	47.21(56.47)	16.22(23.55)	10.73(14.44)
$K = 8$	43.95(53.90)	50.79(52.08)	19.15(23.80)	10.30(15.07)
B. Number of reasoning primitives (M)				
$M = 1$	43.95(56.17)	50.79(54.18)	19.15(23.99)	10.30(14.47)
$M = 2$	50.64(55.32)	50.66(55.94)	15.77(25.19)	8.16(15.27)
$\mathbf{M = 4}$	51.75(54.04)	52.01(55.01)	17.34(22.79)	8.49(13.23)
C. Number of encoder layers (L_e)				
$L_e = 1$	42.55(53.33)	47.43(52.43)	18.42(24.57)	11.66(12.03)
$\mathbf{L_e = 2}$	51.75(54.04)	52.01(55.01)	17.34(22.79)	8.49(13.23)
$L_e = 4$	43.97(52.84)	49.23(53.30)	14.27(26.30)	10.99(14.45)

Table 7: Ablation study of controller capacity on 2WikiMultihopQA using Qwen2.5-1.5B. We vary the number of memory slots K, reasoning primitives M, and encoder layers L_e while keeping the remaining hyperparameters fixed. Results of DualPRAG(DualPRAG+) are reported as F1 by question type.

C Supplement Ablation Study Results

The ablation results of controller hyperparameters on Qwen2.5-1.5B and LLaMA3-8B are shown in Table 7 and Table 8, respectively. We study three important hyperparameters: the number of memory slots K, the number of reasoning primitives M, and the number of encoder layers L_e, following the same one-factor-at-a-time setting as in the main text.

Overall, the results show a similar trend across different backbone models. Increasing the controller capacity does not necessarily produce consistent gains. For the fact memory slots, a moderate number of slots provides sufficient capacity to represent retrieved evidence while avoiding redundant memory representations and additional computation. For the reasoning primitives, a small set of primitives is generally enough to capture diverse reasoning transformations, whereas using too many primitives may introduce unnecessary routing complexity. For the compiler depth, increasing the encoder from a shallow configuration improves ev-

Setting	2WQA			
	Compare	Bridge	Inference	Compose
A. Number of memory slots (K)				
$K = 1$	60.26(74.64)	65.72(71.49)	24.64(35.44)	16.29(25.53)
$\mathbf{K = 2}$	63.12(69.27)	63.98(72.33)	24.84(32.32)	14.70(22.94)
$K = 4$	62.51(71.67)	63.98(71.16)	21.20(33.65)	15.62(22.95)
$K = 8$	63.36(69.91)	64.41(70.80)	23.49(33.67)	14.68(22.70)
B. Number of reasoning primitives (M)				
$M = 1$	64.40(75.29)	64.29(72.97)	27.71(35.11)	18.28(21.93)
$M = 2$	61.66(73.07)	62.67(70.93)	23.73(34.87)	15.01(24.28)
$\mathbf{M = 4}$	63.12(69.27)	63.98(72.33)	24.84(32.32)	14.70(22.94)
C. Number of encoder layers (L_e)				
$L_e = 1$	61.97(73.15)	65.29(71.69)	25.77(35.94)	15.34(24.23)
$\mathbf{L_e = 2}$	63.12(69.27)	63.98(72.33)	24.84(32.32)	14.70(22.94)
$L_e = 4$	61.24(70.70)	60.92(67.81)	20.94(34.84)	12.72(23.39)

Table 8: Ablation study of controller capacity on 2WikiMultihopQA using LLaMA3-8B. We vary the number of memory slots K, reasoning primitives M, and encoder layers L_e while keeping the remaining hyperparameters fixed. Results of DualPRAG(DualPRAG+) are reported as F1 by question type.

idence encoding, but deeper encoders bring limited additional benefit while increasing training and inference cost.

These observations support the default configuration used in our experiments, namely $K = 2$, $M = 4$, and $L_e = 2$. This setting offers a favorable trade-off between effectiveness and efficiency across backbone models.