NeuRAG: End-to-End Neural Knowledge Augmentation via Hyper-Neurons

Liwei Zheng, Xuemin Liu, Jie Liu*
College of Artificial Intelligence, Nankai University, Tianjin 300350, CHINA
{zhengliwei,1120240277}@mail.nankai.edu.cn
jliu@nankai.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) systems have become a standard approach for grounding large language models in external knowledge. However, they are constrained by a decoupled architecture: retrieval and reasoning operate as separate stages, with retrieved text merely prepended as passive context. This prevents deep integration of knowledge into the model's parametric reasoning, leading to fragmented responses for complex queries requiring multi-document synthesis or conflict resolution. To bridge this gap, we propose **NeuRAG**, an end-to-end Neuralized RAG framework that unifies knowledge retrieval and fusion through **Hyper-Neurons**—parameterized modules encoding entire documents directly into the model's parameter space, framing retrieval as dynamic reconfiguration of the model's parameter space rather than static context augmentation. In NeuRAG, each document is encoded as a lightweight LoRA module, conceptualized as a knowledge neuron. These neurons collectively form a document-adaptive **Hyper-Layer**, which dynamically activates and fuses knowledge neurons via an attention mechanism conditioned on the input hidden-state query. This enables the model to jointly retrieve and reason within a single forward pass, seamlessly integrating external knowledge into its inference pathway. Extensive experiments across multiple datasets and LLMs demonstrate NeuRAG's strong and consistent performance as a promising novel RAG paradigm.

1 Introduction

Retrieval-augmented generation (RAG) (Lewis et al., 2020; Borgeaud et al., 2022; Guu et al., 2020; Izacard and Grave, 2021) incorporates external knowledge at inference time to compensate for the inability of large language models (LLMs) (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023) to update their internal knowledge in real time, helping reduce knowledge staleness and factual hallucinations.

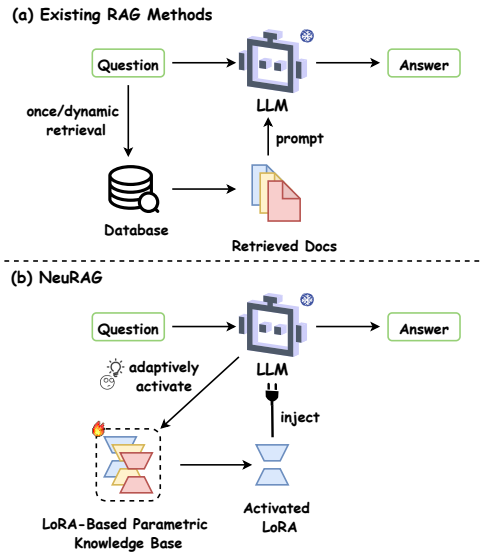

Figure 1: Comparison of inference paradigms. (a) Existing RAG methods perform retrieval as an external operation that is not informed by the model's internal reasoning state. (b) NeuRAG integrates retrieval into the decoding process through hidden-state-driven parameter updating.

Although existing RAG methods adopt diverse designs, we identify a common architectural limitation: retrieval and generation remain isolated. The retrieved texts are used as surface-level context that the model must interpret from scratch during each generation, rather than as an integral end-to-end augmentation of the model's generative computation, as is shown in Figure 1(a). This architectural choice results in a cascade of well-documented shortcomings: (1) **context window bottleneck** that limits the volume and scope of integrable knowledge; (2) **inefficient reasoning**, as the model redundantly parses retrieved text without leveraging structured prior understanding; and (3)

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 30324–30343
July 2-7, 2026

fragmented optimization, where the retriever and generator are tuned in isolation, misaligned with the end-to-end reasoning objective. As a result, such decoupled designs fail to achieve the deep, parametric integration required for effective knowledge internalization and augmentation, and often expose brittleness when faced with tasks involving multi-document synthesis, conflict resolution, or complex multi-step reasoning.

In this paper, we introduce NeuRAG, a framework that reimagines RAG as an end-to-end neural process for direct knowledge augmentation, as is shown in Figure 1(b). NeuRAG's core innovation lies in its neuralized knowledge augmentation mechanism, framing retrieval as dynamic reconfiguration of the model's parameter space rather than static context augmentation, which conceptualizes each document as a Hyper-Neuron, instantiated via a LoRA module as a lightweight and modular parameter block that encodes document-level knowledge. A collection of **Hyper-Neurons** forms a dynamic, overlay network—a **Knowledge Hyper-Layer**—on top of the base LLM. For a given hidden-state query, a learned neural router assesses relevance and computes activation weights for each Hyper-Neuron. These weighted Hyper-Neurons are then fused in parameter space and injected into the forward pass, temporarily and adaptively augmenting the model's very reasoning circuitry with the precise knowledge needed.

This proposed knowledge augmentation process, which we term Hyper-Neuron Fusion, collapses the traditional retrieve-then-read pipeline into a single, differentiable forward pass. It moves beyond retrieving for reasoning to achieving retrieval as augmentation, where external knowledge dynamically becomes part of the model's parametric self. NeuRAG thus provides a pathway from statically prompted LLMs towards dynamically composable models, whose functional parameters can be fluidly assembled based on context.

In summary, our contributions are threefold:

- We propose a novel paradigm for neural knowledge augmentation, replacing textual context concatenation with dynamic parameter-space integration via Hyper-Neurons.
- By conceptualizing documents as activatable, composable parameter modules (i.e., Hyper-Neurons) and introducing a Knowledge Hyper-Layer for their dynamic orchestration, we design a fully differentiable architecture where retrieval, weighting, and fusion of knowledge are jointly optimized for the same task.
- We conduct extensive experiments across multiple datasets and LLMs, demonstrating the effectiveness of NeuRAG and showing its potential as a next-generation RAG paradigm.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) augments LLMs by retrieving external documents and appending them to the input context (Lewis et al., 2020; Borgeaud et al., 2022; Guu et al., 2020; Izacard and Grave, 2021). Recent RAG works have explored different designs for incorporating external knowledge into generation. Dynamic RAG approaches (Trivedi et al., 2023; Asai et al., 2024; Jiang et al., 2023; Jeong et al., 2024; Su et al., 2024) iteratively inject newly retrieved documents as textual context during inference. In contrast, Parametric RAG (Su et al., 2025; Tan et al., 2025) encodes external documents as LoRA modules, enabling knowledge incorporation without expanding the input context. Despite their diverse designs, existing paradigms decouple retrieval from the generation process, which prevents the retrieval stage from leveraging the model's reasoning capabilities effectively. In contrast, our method neuralizes retrieval to incorporate external knowledge as dynamically activated and fused parameters within the decoding computation itself.

2.2 Retrieval of External Knowledge

Most existing RAG methods retrieve external knowledge through text- or embedding-based similarity matching, including classical lexical methods and dense neural retrievers (Robertson et al., 1994; Karpukhin et al., 2020; Chen et al., 2024). Recent works have further explored hidden-state-driven routing over a large pool of LoRA modules during decoding (Fleshman and Van Durme, 2025). While effective, these approaches treat knowledge retrieval as auxiliary mechanisms that are supported by externally constructed indices. In contrast, our approach models retrieval as an internal, hidden-state-driven operation, which serves as an integral part of the model's internal generation process.

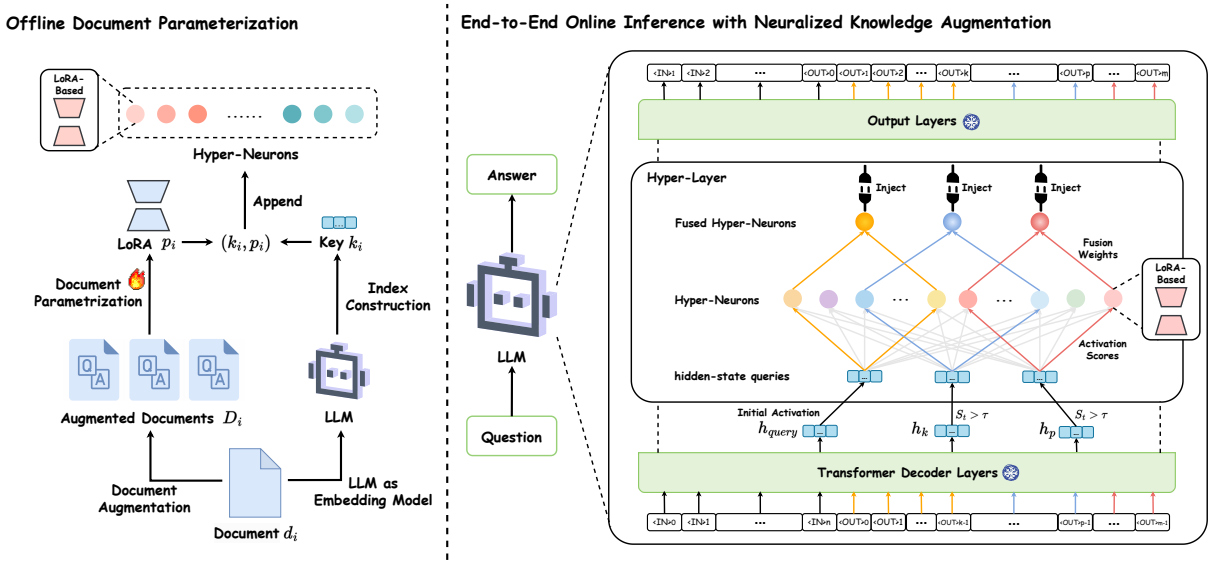

Figure 2: Overview of NeuRAG framework. Left: Offline document parameterization as Hyper-Neurons. Right: End-to-end online inference with neuralized knowledge augmentation. NeuRAG augments LLM generation process with Hyper-Layer, which is triggered by default when generating the first output token $< OUT >_0$, and triggered if $S_t > \tau$ for subsequent generation steps (when generating $< OUT >_k$ and $< OUT >_p$). Each time Hyper-Layer is triggered, the model's current hidden state will be fed into a neural router to compute activation scores for each Hyper-Neuron. Top-k Hyper-Neurons are then fused and injected (indicated by) into the model's FFN layers, taking effect until Hyper-Layer is triggered again. Color-coded vertical arrows () indicate partial generation process augmented by different Hyper-Neuron configurations.

3 Methodology

3.1 Overview of NeuRAG Framework

NeuRAG reformulates retrieval-augmented generation as an end-to-end neuralized process for direct knowledge augmentation. Rather than treating retrieval as an external, stage-wise operation separated from generation, NeuRAG incorporates external knowledge as dynamically activated parameter modules that participate directly in the model's decoding computation.

As shown in Figure 2, NeuRAG represents each external document as a lightweight, document-specific LoRA module, which is conceptualized as a Hyper-Neuron, and organizes these units into a Knowledge Hyper-Layer over the base LLM (§3.2). During decoding, the model continuously conditions on its evolving hidden states to assess relevance, activate and fuse selected Hyper-Neurons on demand, dynamically updates the model's effective parameters (§3.3). This design enables retrieval, weighting, and fusion of knowledge to be performed end-to-end as a unified parametric computation.

3.2 Offline Document Parameterization as Hyper-Neurons

Following PRAG (Su et al., 2025), NeuRAG adopts the same document parameterization approach, which converts each external document d_i into a document-level LoRA module p_i, which is further conceptualized as a document-specific Hyper-Neuron. Detailed implementationS are provided in Appendix A.2.

Hidden-State–Aligned Index Construction. To enable Hyper-Neuron activation conditioned on the model's internal decoding states, NeuRAG constructs a hidden-state–aligned index. Specifically, each document d_i is fed into the base LLM, and the hidden state of the final input token at the last layer L is extracted as the corresponding Hyper-Neuron's index key:

$$k_i = h_i^L, \tag{1}$$

where h_i^L denotes the last-layer hidden state associated with the final token of document d_i. The resulting Hyper-Neuron key-value set is defined as:

$$\mathcal{P} = \{(k_i, p_i)\}_{i=1}^{N}, \tag{2}$$

where each key k_i indexes its corresponding Hyper-Neuron p_i in the same hidden-state space used dur-

ing decoding-time activation. We stack all document keys into a matrix $K = [k_1; \ldots; k_N] \in \mathbb{R}^{N \times d}$, which serves as the frozen Hyper-Neuron index matrix.

3.3 End-to-End Online Inference with Neuralized Knowledge Augmentation

NeuRAG performs neuralized knowledge augmentation with the Hyper-Layer, which is dynamically triggered by a hidden-state-driven signal during the generation process. Hyper-Layer consists of a learned neural router and Hyper-Neurons. During its forward propagation, the router retrieves and activates relevant Hyper-Neurons, which are then fused and injected into the feed-forward network (FFN) layers of the base model, influencing subsequent token generation steps. Detailed algorithm is presented in Appendix A.4.

Hidden-State–Driven Trigger Signals. Let $h_t \in \mathbb{R}^d$ denote the last-layer decoder hidden state used to generate token x_t at decoding step t. NeuRAG derives two complementary signals based on h_t to assess the model's demand for updating parametric knowledge during decoding.

Hidden-State Drift Signal. The first signal measures semantic drift of the internal representation:

$$\Delta h_t = \frac{\|h_t - h_{t-1}\|_2}{\|h_{t-1}\|_2 + \varepsilon}. \tag{3}$$

This signal reflects how strongly the current representation deviates from the recent decoding history, indicating potential misalignment between the previously activated Hyper-Neurons and the evolving semantic requirements at step t.

Predictive Entropy Signal. The second signal captures predictive uncertainty, which we quantify using token-level entropy at decoding step t:

$$H_t = -\sum_{v \in \mathcal{V}} p_t(v) \log(p_t(v) + \varepsilon), \tag{4}$$

where $p_t(v)$ denotes the probability assigned to candidate token v in the vocabulary $\mathcal{V}$ at decoding step t. High entropy indicates increased reasoning uncertainty in the model's prediction over possible next tokens, suggesting that the current parametric configuration may be insufficient for continuing generation.

The two signals are combined into a unified score:

$$S_t = \lambda \, \widehat{\Delta h}_t + (1 - \lambda) \, \widehat{H}_t, \tag{5}$$

where $\widehat{\Delta h}_t$ and $\widehat{H}_t$ are obtained via sliding-window normalization (Appendix A.9). The unified score S_t serves as a trigger signal for determining when parametric knowledge should be reconfigured during decoding.

Neural Routing and Hyper-Neuron Fusion. At the first decoding step ($t = 0$), i.e., before the first output token is generated, Hyper-Layer is triggered by default to establish the initial parametric knowledge augmentation state. For subsequent steps ($t > 0$), a new parametric injection induced by newly activated Hyper-Neurons is applied only when the unified activation score satisfies $S_t > \tau$ to update the model's effective parameters on demand. The parametric injection at step t takes effect in subsequent decoding steps upon the next trigger of Hyper-Layer, ensuring uninterrupted generation.

When Hyper-Layer is triggered, a learned neural router computes activation scores over Hyper-Neurons in the Knowledge Hyper-Layer. Formally, the router projects the current hidden state h_t into the Hyper-Neuron index space:

$$\begin{cases} q_t = h_t W_r + b, \\ s_t = q_t K^\top, \end{cases} \tag{6}$$

where $W_r \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$ are the learnable parameters, and $K \in \mathbb{R}^{N \times d}$ is the frozen document-key matrix. Details of neural router training are provided in Appendix A.8.

Based on s_t, the Top-k most relevant Hyper-Neurons $\mathcal{E}_t$ are activated and softly fused to produce a single Hyper-Neuron update:

$$P_t = \sum_{i \in \mathcal{E}_t} w_{t,i} \, p_i, \tag{7}$$

with fusion weights computed via a restricted softmax:

$$w_{t,i} = \frac{\exp(s_{t,i})}{\sum_{j \in \mathcal{E}_t} \exp(s_{t,j})}, \quad i \in \mathcal{E}_t. \tag{8}$$

The fused Hyper-Neuron representation P_t is then injected into the FFN layers as a parametric knowledge augmentation:

$$W_t = W + P_t, \tag{9}$$

where W denotes the original FFN weight matrix, P_t represents the fused low-rank Hyper-Neuron update applied during decoding, and t indexes the current decoding step.

3.4 Dynamic Adaptation

NeuRAG decouples Hyper-Neuron instantiation from neural router training, enabling efficient adaptation to changes in the external knowledge base without modifying the base LLM. When documents are added to or removed from the knowledge repository, the Knowledge Hyper-Layer is updated accordingly by instantiating or removing the corresponding Hyper-Neurons, while the lightweight neural router can be retrained offline to realign the hidden-state–to–activation mapping efficiently.

4 Experimental Setup

4.1 Datasets and Metrics

We evaluate our method on five widely-used knowledge-intensive QA benchmarks that cover multi-hop reasoning, compositional reasoning, and reading comprehension. For multi-hop question answering, we include 2WikiMultihopQA (2WikiQA) (Ho et al., 2020) and HotpotQA (Yang et al., 2018), which require integrating evidence from multiple documents. To assess compositional and commonsense reasoning, we experiment with ComplexWebQuestions (CWQ) (Talmor and Berant, 2018) and StrategyQA (SQA) (Geva et al., 2021), where answers often require multi-step inference. We additionally use IIRC (Ferguson et al., 2020) to evaluate performance on incremental reading comprehension tasks, where models must reconcile information scattered across multiple passages.

For each dataset, following the setting of PRAG (Su et al., 2025), we use the first 300 questions of each subtask for evaluation. Separately, we collect another 100 questions from each subtask to train the neural router, with detailed information provided in Appendix A.8. Utilizing documents from the combined set of evaluation and neural router training questions, we construct the Hyper-Neuron set using the procedure described in §3.2.

For evaluation metric, we adopt F1 score to provide a balanced performance assessment by jointly capturing precision and recall and appropriately rewarding partially correct predictions.

4.2 Base Models

We implement our approach using three representative open-source LLMs, including Qwen2.5-1.5B-Instruct (Yang et al., 2025), LLaMA3.2-1B-Instruct (Meta, 2024a), and LLaMA3-8B-Instruct (Meta, 2024b). All experiments are conducted using PyTorch on NVIDIA RTX 4090 and RTX 3090 GPUs with 24 GB of memory.

4.3 Baselines

We compare NeuRAG with representative baselines spanning Standard RAG, Dynamic RAG, and Parametric RAG paradigms, including Vanilla LLM, Standard RAG, FLARE (Jiang et al., 2023), DRAGIN (Su et al., 2024), PRAG (Su et al., 2025), and DyPRAG (Tan et al., 2025). Implementation details of all baselines are provided in Appendix A.3.

5 Experimental Results

5.1 Main Results

We evaluate NeuRAG against representative RAG baselines across three base LLMs with settings provided in Appendix A.13 and prompt templates used are provided in Appendix A.12. The complete results are reported in Table 1. To provide a robustness-aware assessment of cross-dataset performance by reducing the influence of extreme cases, we additionally perform a rank-based evaluation, results are reported in Table 2. Specifically, methods are ranked independently on each dataset, with the top three receiving 3, 2, and 1 points respectively, and the final score is obtained by aggregating points across datasets. Our key observations are summarized below.

Overall Performance. NeuRAG consistently achieves both the best average and rank-based evaluation performance across datasets and LLMs, suggesting that decoding-aware Hyper-Neuron activation effectively supports evolving reasoning trajectories. Consistent improvements across Qwen2.5-1.5B-Instruct, LLaMA3.2-1B-Instruct, and LLaMA3-8B-Instruct further demonstrate strong cross-model generalization.

Comparison with Dynamic RAG. Dynamic RAG methods such as DRAGIN and FLARE iteratively retrieve and inject textual context during inference, which remains constrained by LLM's context capacity and stage-wise interaction between retrieval and generation. As a result, retrieval cannot be tightly coupled with the model's decoding dynamics. In contrast, NeuRAG performs retrieval via hidden-state-driven parametric activation within a single decoding process on demand, enabling continuous knowledge integration and augmentation.

Comparison with Parametric RAG. PRAG encodes documents as LoRA modules, enabling

Base Model	Method	2WikiMultihopQA					HotpotQA			IIRC	SQA	CWQ	AVG
		Compare	Bridge	Inference	Compose	Avg	Bridge	Compare	Avg				
LLaMA3.2-1B-Instruct	Vanilla	0.4289	0.2417	0.1691	0.0787	0.2296	0.1325	0.4026	0.2676	0.0598	0.5800	0.3494	0.2973
	Standard RAG	0.4123	0.2678	**0.2251**	**0.1021**	0.2518	**0.2138**	0.4246	**0.3192**	0.1034	0.5900	0.3739	0.3277
	FLARE	0.4013	0.2589	0.1960	0.0823	0.2346	0.163	0.3784	0.2707	0.0847	0.5867	0.3173	0.2988
	DRAGIN	0.4556	0.3357	0.1919	0.0901	0.2683	0.1431	0.4015	0.2723	0.0747	0.6067	**0.3900**	0.3224
	PRAG	0.5020	0.2434	0.1911	0.0824	0.2547	0.1365	0.4090	0.2728	0.1313	0.5600	0.3586	0.3155
	DyPRAG	0.5125	0.4815	0.1735	0.0754	0.3107	0.1405	**0.4390**	0.2898	0.1328	0.5800	0.3686	0.3364
	NeuRAG	**0.5153**	**0.5189**	0.1788	0.0786	**0.3229**	0.1432	0.4318	0.2875	**0.1384**	**0.6500**	0.3498	**0.3497**
Qwen2.5-1.5B-Instruct	Vanilla	**0.4574**	0.3906	0.1704	0.0727	0.2728	0.1218	0.3964	0.2591	0.1712	0.6067	0.2647	0.3149
	Standard RAG	0.3875	0.3884	0.1187	0.0568	0.2379	**0.1619**	0.3713	0.2666	**0.1896**	0.6100	0.2823	0.3173
	FLARE	0.1896	0.1282	0.0852	0.0437	0.1117	0.0750	0.1229	0.0990	0.1461	0.5300	0.1647	0.2103
	DRAGIN	0.2771	0.1826	0.1025	0.0680	0.1576	0.0801	0.1851	0.1326	0.1038	0.6000	0.1788	0.2346
	PRAG	0.4496	0.4396	0.1929	**0.1114**	0.2984	0.1327	0.4042	0.2685	0.1267	**0.6500**	0.3082	0.3303
	DyPRAG	0.4303	0.4720	0.1704	0.0855	0.2896	0.1372	0.4139	0.2756	0.1023	0.5800	**0.3194**	0.3134
	NeuRAG	0.4553	**0.4933**	**0.2113**	0.0951	**0.3138**	0.1265	**0.4553**	**0.2909**	0.1200	0.6300	0.3159	**0.3341**
LlaMA3-8B-Instruct	Vanilla	0.5490	0.5520	0.2459	0.1443	0.3728	0.1900	0.4563	0.3232	0.1953	0.7233	0.4244	0.4078
	Standard RAG	0.5843	0.4777	0.1920	0.1107	0.3412	0.1968	0.4210	0.3089	0.2164	0.7067	0.3545	0.3855
	FLARE	0.4293	0.3769	**0.3086**	0.1627	0.3194	0.2493	0.4324	0.3409	0.1932	0.6867	0.3084	0.3697
	DRAGIN	0.5185	0.4480	0.2664	0.1833	0.3541	0.2618	0.6116	0.4367	0.2049	0.7200	0.3101	0.4052
	PRAG	0.5778	0.5893	0.2761	**0.1917**	0.4087	**0.3368**	**0.6588**	**0.4978**	0.1971	0.7156	0.4354	0.4509
	DyPRAG	0.5739	0.5643	0.2533	0.1888	0.3951	0.2485	0.5859	0.4172	0.1617	0.6800	0.4187	0.4145
	NeuRAG	**0.5921**	**0.5980**	0.2640	0.1824	**0.4091**	0.2707	0.6473	0.4590	**0.2247**	**0.7700**	**0.4375**	**0.4601**

Table 1: Overall results of NeuRAG and competing RAG baselines on five knowledge-intensive QA benchmarks. All metrics are F1 scores. Avg denotes the average performance across different subtasks within each dataset, while AVG denotes the average performance across all datasets. Bold and underlined values indicate the best and second-best results, respectively. For 2WikiMultihopQA, HotpotQA, and CWQ, results of Vanilla, Standard RAG, PRAG, and DyPRAG are taken from Tan et al. 2025, while results of FLARE and DRAGIN are taken from Su et al. 2025.

Method	LLaMA3.2-1B	Qwen2.5-1.5B	LLaMA3-8B
Vanilla	0	2	3
Standard RAG	6	4	2
FLARE	0	1	0
DRAGIN	6	0	3
PRAG	1	7	7
DyPRAG	7	6	1
NeuRAG	**10**	**10**	**14**

Table 2: Results of rank-based evaluation. 2WikiQA and HotpotQA are evaluated using Avg. Bold values indicate the best performance.

parametric knowledge injection but remains a fixed parametric configuration throughout decoding, while DyPRAG allows test-time generation yet still lacks token-level adaptation. By activating and fusing Hyper-Neurons based on hidden-state feedback on demand, NeuRAG achieves fine-grained, decoding-aware parametric adaptation.

5.2 Effectiveness and Behavior Analysis

We present quantitative and qualitative analyses to examine the effectiveness, necessity, and behavioral characteristics of NeuRAG's end-to-end neuralized knowledge augmentation mechanism.

Ablation on Hyper-Neuron Activation Mechanisms. We first examine whether decoding-aware Hyper-Neuron activation is necessary for effective neural knowledge augmentation. Table 3 compares four variants: (i) a **static configuration**, where a fixed set of Hyper-Neurons is activated only once before decoding; (ii) a **random activation** variant, which preserves the activation mechanism but randomly selects Hyper-Neurons at each activation; (iii) a **similarity-based** variant, which replaces the learned neural router with direct dot-product matching between hidden states and Hyper-Neuron keys; and (iv) NeuRAG.

The static configuration leads to a substantial performance drop, indicating that a single-time, fixed parametric knowledge augmentation is insufficient for reasoning processes which requires knowledge evolving during decoding. Random activation yields marginal gains because random activated Hyper-Neurons can provide additional reasoning capabilities, but unstructured activation fails to consistently align with the model's reasoning state. Notably, the similarity-based baseline also underperforms NeuRAG, demonstrating that the LLM's native hidden-state semantic is not inherently aligned with effective Hyper-Neuron activation. Overall, these results demonstrate that

Method	Cpare	Bridge	Inf.	Cpose	Avg
Static config.	0.4935	0.4559	0.1510	0.0484	0.2872
Random	0.5141	0.4704	0.1379	0.0618	0.2961
Similarity	0.4850	0.5074	0.1550	0.0573	0.3012
NeuRAG	**0.5153**	**0.5189**	**0.1788**	**0.0786**	**0.3229**

Table 3: Effect of Hyper-Neuron reconfiguration strategies on 2WikiMultihopQA with LLaMA3.2-1B-Instruct. Bold values indicate the best performance.

Setting	Cpare	Bridge	Inf.	Cpose	Avg
Delay 5 tks	**0.5200**	0.4597	0.1337	0.0519	0.2913
After 50 tks	0.5141	0.4755	0.1580	0.0559	0.3009
NeuRAG	0.5153	**0.5189**	**0.1788**	**0.0786**	**0.3229**

Table 4: Effect of temporal constraints on Hyper-Neuron reconfiguration on 2WikiMultihopQA with LLaMA3.2-1B-Instruct.

effective performance gains arise from aligning the model's internal reasoning signals with structured, on-demand Hyper-Neuron activation, rather than from Hyper-Neuron injection alone.

Temporal Sensitivity of Hyper-Neuron Activation. To investigate the temporal sensitivity of neuralized parametric knowledge augmentation, we impose two temporal constraints on Hyper-Neuron activation and injection: (a) delaying each activation by five decoding steps (Delay 5 tks), and (b) disabling activation after the first 50 generated tokens (After 50 tks). Results are shown in Table 4.

Both constraints lead to clear performance degradation. Delaying activation impedes the timely acquisition of new knowledge during the reasoning process, while disabling activation after 50 tokens deprives the model of external information required in the later stages of complex reasoning process. These results indicate that effective neuralized knowledge augmentation must be triggered promptly in response to internal reasoning dynamics and remain available throughout the entire decoding trajectory.

Behavioral Efficiency of Hyper-Neuron Activation and Injection. We analyze the efficiency of NeuRAG's parametric knowledge augmentation by measuring activation frequency and temporal spacing (results are reported in Table 5). Across datasets and LLMs, Hyper-Layer is triggered only a few times per example (typically 1–4), with long intervals of 8–20 tokens. This behavior indicates

Model	2Wiki		Hotpot		CWQ	
	Cnt	Intv	Cnt	Intv	Cnt	Intv
LLaMA3.2-1B	2.91	13.84	1.64	9.18	0.29	0.38
Qwen2.5-1.5B	2.64	15.18	1.67	8.54	0.40	0.33
LLaMA3-8B	2.18	10.41	1.86	7.76	0.48	1.95

Model	StrategyQA		IIRC	
	Cnt	Intv	Cnt	Intv
LLaMA3.2-1B	1.26	8.10	1.28	9.94
Qwen2.5-1.5B	1.67	9.32	1.25	3.90
LLaMA3-8B	1.35	6.81	1.29	4.12

Table 5: Reconfiguration statistics across datasets (averaged per question). Cnt denotes the average number of reconfiguration events, and Intv denotes the average token interval between consecutive reconfigurations.

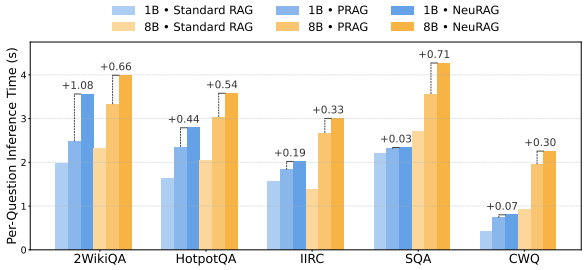

Figure 3: Comparison of the average per-question inference time for Standard RAG, PRAG, and NeuRAG on LLaMA3.2-1B-Instruct and LLaMA3-8B-Instruct across multiple datasets.

selective, on-demand parametric knowledge augmentation rather than frequent or unstable parameter switching. More complex reasoning tasks exhibit higher activation frequency, reflecting greater variability in knowledge demand. Additional visualizations on ComplexWebQuestions presented in Appendix A.10 further confirm that the activated Hyper-Neurons are diverse and task-specific.

We further evaluate inference efficiency (results are shown in Figure 3). Compared to PRAG, NeuRAG incurs less than one second of additional latency in most settings, demonstrating that decoding-aware Hyper-Neuron activation and injection can be achieved with modest computational overhead. A comprehensive breakdown of latency across different model scales and specific operational stages is provided in Appendix A.11.

Case Study: Token-Level Dynamics of Hyper-Neuron Activation. Figure 4 presents a token-level analysis of a representative example from 2WikiMultihopQA with a 128-token reasoning trajectory. Hyper-Layer is triggered only when the

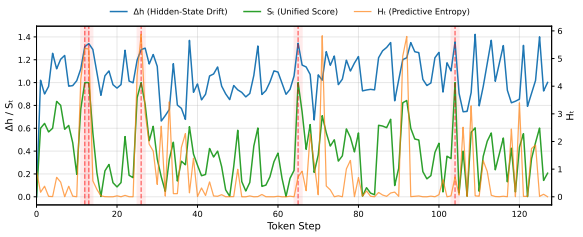

Figure 4: Token-level reasoning trajectory of the example. Vertical markers indicate the steps where retrieval is triggered, coinciding with peaks in the unified trigger score S_t. Raw Δh_t and H_t are displayed together.

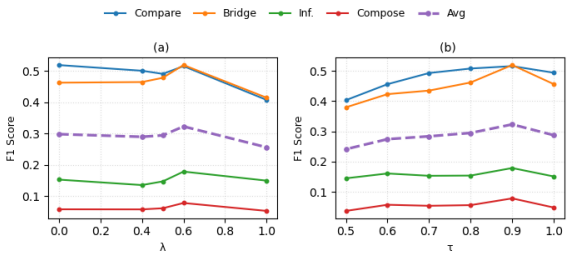

Figure 5: (a) Impact of the mixing factor λ of retrieval decisions and (b) Influence of the trigger threshold τ on 2WikimultihopQA with LLaMA3.2-1B-Instruct.

unified score S_t exceeds the threshold τ. A detailed step-by-step analysis of this example, provided in Appendix A.5, indicates that these moments correspond to shifts between distinct subproblems in the multi-hop reasoning process, and are associated with changes in knowledge demand.

5.3 Hyperparameter Analysis

Mixing Factor λ in S_t. The mixing factor λ balances normalized hidden-state drift signal and predictive entropy signal in the unified activation score. Using either signal alone ($\lambda = 0$ or $\lambda = 1$) degrades performance, indicating that the two signals capture complementary aspects of the model's internal state. As shown in the left panel of Figure 5, performance peaks around $\lambda \approx 0.6$, suggesting that Hyper-Neuron reconfiguration is most effective when both representation drift and predictive uncertainty jointly signal increased knowledge demand. Additional analysis in Appendix A.7 shows that hidden-state drift signal provides more frequent variation, while predictive entropy signal contributes sparser but complementary cues, consistent with the observed optimal choice of λ.

Trigger Threshold τ. We vary the trigger threshold τ from 0.5 to 1.0, with representative answer trajectories provided in Appendix A.6. As shown

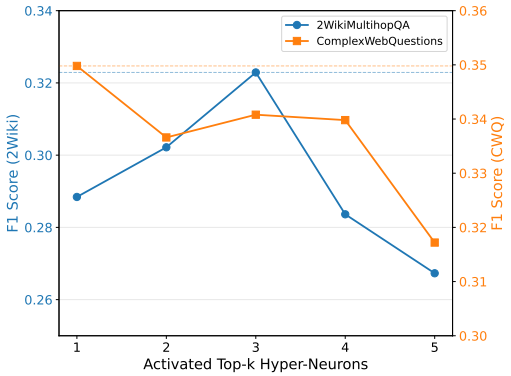

Figure 6: Effect of the number of activated Hyper-Neurons each time on 2WikimultihopQA and ComplexWebQuestions with LLaMA3.2-1B-Instruct.

in the right panel of Figure 5, disabling reconfiguration ($\tau = 1.0$) leads to performance drop, indicating that a fixed parametric configuration is insufficient for long evidence chains. Conversely, overly permissive thresholds cause frequent and unstable Hyper-Neuron switching. Moderate values $\tau \in [0.8, 1.0)$ provide the best trade-off between stability and adaptability, suggesting that Hyper-Neuron reconfiguration should be selective rather than frequent.

Number of Activated Hyper-Neurons (Top-k). Figure 6 reports performance for Top-$k \in [1, 5]$ on 2WikiMultihopQA and CWQ. Increasing k does not consistently improve performance. Because NeuRAG performs on-demand Hyper-Neuron reconfiguration during decoding, a small number of activated Hyper-Neurons is often sufficient, while fusing too many can dilute document-specific parametric contributions. Overall, performance varies smoothly across different values of k, indicating that NeuRAG is not sensitive to the exact choice of k and benefits more from precise, selective activation than from overly large sets of activated Hyper-Neurons.

6 Conclusions

In this work, we identify a common limitation of existing RAG methods that retrieval and reasoning remain structurally separated, preventing external knowledge retrieval from being deeply and seamlessly integrating with LLM's generation process. To address this, we propose NeuRAG, an end-to-end knowledge augmentation framework. NeuRAG equips base LLM with a knowledge Hyper-Layer, which contains knowledge-

encoded LoRA modules as Hyper-Neurons, enabling decoding-dynamics-awared knowledge retrieval and fusion through its carefully designed forward propagation mechanism. Extensive experiment results show the effectiveness and robustness of NeuRAG, demonstrating its potential as the next-generation RAG paradigm.

Limitations

Our work also has several limitations. Firstly, it depends on access to intermediate hidden states and dynamic parameter composition, which could not be applied to closed-source LLMs. Besides, operating over a large collection of document-specific LoRA modules may pose scalability challenges, particularly in memory footprint and retrieval management. Furthermore, the Hyper-Layer adopts a relatively simple FNN-like architecture. Future works could explore less intrusive retrieval signals, more efficient retrieval and caching strategies to enhance scalability and generalization, and more complex Hyper-Layer structure to support deeper knowledge integration.

Acknowledgments

This research is supported by the National Key Research and Development Program of China under the grant No. 2023YFF0725003, National Natural Science Foundation of China under the grant No. 62376129, Tianjin Science and Technology Plan Project under the grant No. 25ZGZNGX00100, Tianjin Science and Technology Plan Project under the grant No. 24JCYBJC01950, and Nankai University-Alpha ESS Smart Energy Storage Joint Research Center Project.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

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.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

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.

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. 2020. IIRC: A dataset of incomplete information reading comprehension questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1137–1147, Online.

William Fleshman and Benjamin Van Durme. 2025. Lora-augmented generation (lag) for knowledge-intensive language tasks. *arXiv preprint arXiv:2507.05346*.

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

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

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.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

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.

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.

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.

Meta. 2024a. Llama-3.2-1b-instruct. https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct. Accessed: 2024-09.

Meta. 2024b. Meta-llama-3-8b-instruct. https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct. Accessed: 2024-04.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST).

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. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

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*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.

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.

A Appendix

A.1 Notations

The notations used in this paper are summarized in Table 6.

Notation	Definition
Model & Parameters	
θ	Frozen pretrained LLM parameters
p_i	LoRA-based Hyper-Neuron for document d_i
P_t	Fused Hyper-Neuron update applied at decoding step t
α	Scaling factor of LoRA
r	LoRA rank
Document & LoRA Modules	
d_i	The i-th document in corpus
$\mathcal{D}_i$	Augmented dataset of rewrites and QA pairs for d_i
x	Training sequence $(d_i^k \oplus q_i^j \oplus a_i^j)$
h_i	Hidden state key derived from encoding document d_i
$\mathcal{P}$	Hidden state–indexed Hyper-Neuron set $\{(k_i, p_i)\}$
Neural Routing and Hyper-Neurons	
W_r, b	Neural router projection parameters
k_i	document key of Hyper-Neuron i
K	Frozen document-key matrix indexing Hyper-Neurons
h_{query}	Final hidden state of the encoded input query
s_t	Neural router activation scores at decoding step t
$\mathcal{E}_t$	Activated Top-k Hyper-Neuron set at step t
$w_{t,i}$	Fusion weight of Hyper-Neuron i at step t
Top-k	Number of activated Hyper-Neurons at each activation point
Dynamic Signals & Trigger	
h_t	Hidden state at decoding step t
Δh_t	Hidden-state drift signal
H_t	Predictive entropy signal
S_t	Unified trigger score
$\mathcal{V}$	Vocabulary
τ	Activation threshold
λ	Mixing factor for $(\widehat{\Delta h}_t,\ \widehat{H}_t)$

Table 6: Notations used in this paper.

A.2 Details of Document Parameterization as Hyper-Neurons

Offline Document Augmentation. To construct document-level Hyper-Neurons, we follow the document parameterization procedure of PRAG (Su et al., 2025). Relevant documents are first retrieved offline from Wikipedia during data preparation. Each document d_i is then augmented to improve robustness and coverage while preserving its factual content.

Specifically, the base LLM is prompted to rewrite d_i into n paraphrased variants $\{d_i^1, d_i^2, \ldots, d_i^n\}$, and to generate m question–answer pairs $\{(q_i^1, a_i^1), (q_i^2, a_i^2), \ldots, (q_i^m, a_i^m)\}$ that are answerable using d_i. The resulting augmented dataset for document d_i is defined as

$$\mathcal{D}_i = \{(d_i^k, q_i^j, a_i^j) \mid k \in [1, n],\ j \in [1, m]\}. \tag{10}$$

Document-Level Parameterization. Given the augmented dataset $\mathcal{D}_i$, we optimize a low-rank parameter update that encodes the knowledge of document d_i into the LLM in a parameter-efficient manner. Each training instance $(d_i^k, q_i^j, a_i^j) \in \mathcal{D}_i$ is concatenated into a single autoregressive training sequence:

$$x = [d_i^k \oplus q_i^j \oplus a_i^j], \tag{11}$$

where $\oplus$ denotes sequence concatenation.

The document-specific LoRA parameters $\Delta\theta_i$ are optimized by minimizing the standard next-token prediction loss:

$$\min_{\Delta\theta_i} \sum_{(d_i^k, q_i^j, a_i^j) \in \mathcal{D}_i} \sum_{t=1}^{T} -\log P_{\theta+\Delta\theta_i}(x_t \mid x_{<t}), \tag{12}$$

where θ denotes the frozen pretrained parameters of the base LLM. The update $\Delta\theta_i$ consists of low-rank LoRA matrices $\{A, B\}$ inserted into the FFN layers.

Concretely, each LoRA module modifies the original FFN weight matrix W via the standard low-rank update:

$$W' = W + \Delta W = W + AB^\top, \tag{13}$$

where $A, B \in \mathbb{R}^{d \times r}$ and $r \ll d$.

After training, the resulting update $\Delta\theta_i$ serves as a parametric representation of document d_i, which we denote as p_i. In NeuRAG, each p_i is treated as a Hyper-Neuron and constitutes a unit in the Knowledge Hyper-Layer. At inference time, these Hyper-Neurons can be selectively activated and fused, enabling knowledge injection without appending document text to the input context.

A.3 Details of Baselines

- **Vanilla** denotes the baseline where the base model answers questions using only its native parametric knowledge, without incorporating any retrieved external information.

- **Standard RAG** follows the classical retrieval-augmented generation pipeline: relevant documents are retrieved before generation, and both the query and the retrieved texts are fed into the model to produce the final answer.

- **FLARE**(Jiang et al., 2023) is a multi-round retrieval-augmented method that triggers retrieval whenever the model detects an uncertain token. Once retrieval is activated, the system forms a query using the most recently generated sentence that excludes the uncertain token, ensuring that the retriever focuses only on the stable portion of the model's output.

- **DRAGIN**(Su et al., 2024) is also a multi-round retrieval-augmented method that triggers retrieval when the model identifies an uncertain token that is semantically meaningful and likely to influence subsequent generation. Once activated, the retriever dynamically constructs the query based on the model's internal states and contextual information.

- **PRAG**(Su et al., 2025) first trains a separate LoRA module for each document in the corpus. At inference time, it retrieves the most relevant documents following the standard RAG procedure, then loads the corresponding LoRA modules onto the base model before answer generation.

- **DyPRAG**(Tan et al., 2025) trains a parametric translator that maps the hidden states of retrieved documents (obtained via an LLM encoder) into LoRAs. During inference, once a document is retrieved, the translator converts its hidden representation into a LoRA adapter, which is then loaded onto the base model to produce the answer.

A.4 NeuRAG Inference Algorithm

This appendix provides implementation-level details of NeuRAG decoding-time Hyper-Neuron activation and fusion. The overall inference procedure is summarized in Algorithm 1.

Algorithm 1 NeuRAG Decoding-Time Hyper-Neuron Activation and Fusion

Require: Input query q; Hyper-Neuron Set $\mathcal{P} = \{(k_i, p_i)\}_{i=1}^{N}$; neural router parameters (W_r, b); Top-k; activation threshold τ; mixing factor λ.

Ensure: Generated answer y

Initial Hyper-Neuron Activation:
Encode the input query and extract the last-layer hidden state h_{query}
Compute activation logits over Hyper-Neurons using the neural router
Select the initial Top-k Hyper-Neurons $\mathcal{E}_0$
Fuse the selected Hyper-Neurons to obtain the initial parametric update P_0
Inject P_0 into the model before generating the first output token
while generation not finished **do**
  Generate the next token and obtain the current hidden state h_t
  Compute trigger signals:
  $\Delta h_t = \frac{\|h_t - h_{t-1}\|_2}{\|h_{t-1}\|_2 + \varepsilon}$
  $H_t = -\sum_{v \in \mathcal{V}} p_t(v) \log(p_t(v) + \varepsilon)$
  Normalize signals
  $S_t = \lambda \widehat{\Delta h}_t + (1 - \lambda) \widehat{H}_t$
  if $S_t > \tau$ **then**
    Compute activation logits over the Knowledge Hyper-Layer from h_t
    Select a new Top-k Hyper-Neuron set $\mathcal{E}_t$
    Fuse the selected Hyper-Neurons to obtain an updated parametric update P_t
    Inject P_t into the model
    // The injection takes effect from the next decoding step
  end if
  $t \leftarrow t + 1$
end while
return y

A.5 Detailed Analysis of a Case Study Example

This appendix provides a structured, token-level analysis of the decoding-time Hyper-Neuron activation and injection behavior illustrated in Figure 4. We report the exact decoding positions where Hyper-Neuron activation is triggered, the local token context, the corresponding internal signals, and the resulting transitions between activated Hyper-Neuron sets.

Dataset: 2WikiMultihopQA

Question: *"Which film has the older director, The Gamecock or Monster A Go-Go?"*
Gold Answer: *"Monster A Go-Go"*

The model produces a 128-token reasoning trajectory that contains five Hyper-Neuron activation events. These events occur at points where the model undergoes clear transitions between distinct reasoning phases, such as shifting the target entity, initiating attribute comparison, or synthesizing intermediate evidence.

Summary of Hyper-Neuron Activation Events

The five activation points correspond to the following semantic transitions along the decoding trajectory:

- **Steps 12–13:** Transition from reasoning about the director of *The Gamecock* to reasoning about the director of *Monster A Go-Go*.
- **Step 26:** Transition from entity description to attribute-level comparison.
- **Step 65:** Transition from evidence narration to answer formulation.
- **Step 104:** Revisiting temporal information for final verification.

Prediction: *"Monster A Go-Go"*

Across all activation points, we observe sharp increases in both hidden-state drift and predictive entropy, resulting in unified activation scores S_t close to 1.0. This indicates that the activation triggers are driven by substantial changes in the model's internal representations rather than incidental noise or local fluctuations.

This detailed trace demonstrates that NeuRAG injects fused Hyper-Neuron updates precisely at points where the model undergoes major reasoning transitions. These token-level dynamics complement the main text by illustrating how internal hidden-state signals guide decoding-time Hyper-Neuron activation and enable adaptive parametric knowledge augmentation during multi-hop reasoning.

A.6 Decoding-Time Hyper-Neuron Activation under Different Thresholds

This appendix presents an extended case study illustrating how the Hyper-Layer trigger threshold τ influences decoding-time parametric injection behavior and the final prediction. We analyze the same example as in Appendix A.5 and visualize the token-level evolution of the hidden-state drift signal Δh_t, predictive entropy signal H_t, and unified activation score S_t over a 128-token decoding trajectory under four different threshold settings.

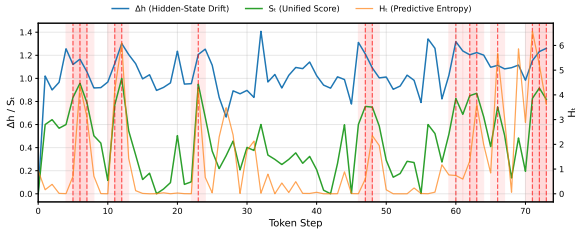

Figure 7: Hyper-Layer trigger signals and reconfiguration points for the case study with threshold $\tau = 0.7$.

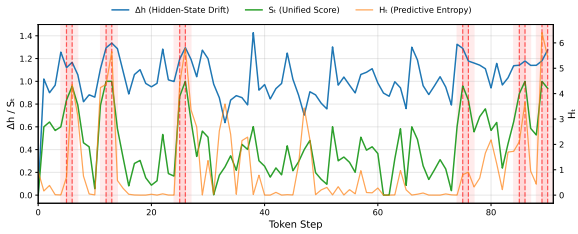

Figure 8: Hyper-Layer trigger signals and reconfiguration points for the case study with threshold $\tau = 0.8$.

Figures 7–10 illustrate how the decoding trajectory varies as τ increases from 0.7 to 1.0. Lower thresholds allow Hyper-Layer to be triggered more frequently, enabling rapid adaptation but also introducing instability due to excessive parametric injections. In contrast, higher thresholds favor stability by suppressing activation, but may delay or entirely miss necessary parametric updates when the model's internal reasoning state shifts.

To highlight the practical impact of threshold selection, Table 8 reports the final predictions produced under each setting. Notably, only the intermediate threshold $\tau = 0.9$ yields the correct answer *"Monster A Go-Go"*. Lower thresholds lead to incorrect predictions due to overly aggressive Hyper-Neuron activation, while disabling activation altogether ($\tau = 1.0$) prevents the model from adaptively injecting new parametric knowledge when required. This case study underscores that effective decoding-time Hyper-Neuron activation should be selective and decisive, rather than frequent or entirely suppressed.

Question: *Which film has the older director, The Gamecock or Monster A Go-Go?*					
Gold Answer: *Monster A Go-Go*					
Step	**Local Token Snippet (±3 tokens)**	S_t	Δh_t	H_t	**Old Top-3 → New Top-3**
12	`... directed by Richard` **`L`** `and . The ...`	0.9999	1.37	0.02	[32,560,889] → [127,103,64]
13	`... by Richard L` **`and`** `. The film ...`	0.9991	1.29	0.01	[127,103,64] → [787,573,789]
26	`... directed by Richard` **`L`** `. Land died ...`	0.9999	1.41	0.05	[787,573,789] → [853,72,879]
65	`... Go - Go .` **`Answer`** `: ...`	0.9999	1.32	0.04	[853,72,879] → [2,1,0]
104	`... David Dhawan` **`died`** `on December ...`	0.9999	1.28	0.03	[2,1,0] → [2,1,0]
Prediction: *Monster A Go-Go*					

Table 7: Comprehensive decoding-time Hyper-Neuron activation trace for the 2WikiMultihopQA case example. Each activation event reports the extended local token context (±3 tokens), the internal activation signals, and the corresponding transition between activated Top-k Hyper-Neuron sets. The token that triggers activation is highlighted with a gray background.

Question	*Which film has the older director, The Gamecock or Monster A Go-Go?*
Gold Answer	*Monster A Go-Go*
Prediction @ $\tau = 0.7$	*The Gamecock*
Prediction @ $\tau = 0.8$	*The Gamecock*
Prediction @ $\tau = 0.9$	*Monster A Go-Go*
Prediction @ $\tau = 1.0$	*The Gamecock*

Table 8: Model predictions for the case study under different Hyper-Neuron reconfiguration thresholds τ. Correct prediction is highlighted in green; incorrect predictions are shown in red.

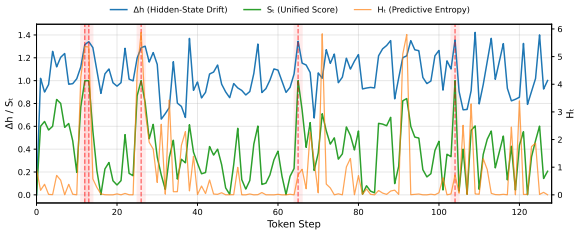

Figure 9: Hyper-Layer trigger signals and reconfiguration points for the case study with threshold $\tau = 0.9$.

A.7 Step-wise Behavior of Hidden-State Drift and Predictive Entropy Signals

Figure 11 illustrates a representative decoding trajectory with the step-wise evolution of hidden-state drift Δh_t, predictive entropy H_t, and the unified trigger score S_t. Red vertical lines mark decoding steps where S_t exceeds the threshold and triggers Hyper-Neuron reconfiguration. Three consistent patterns emerge.

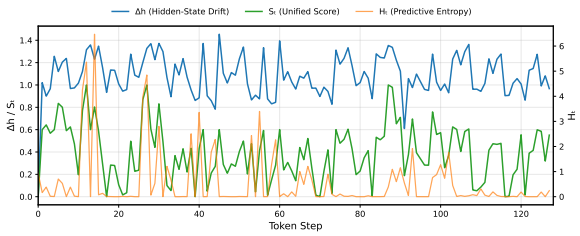

Figure 10: Hyper-Layer trigger signals and reconfiguration points for the case study with threshold $\tau = 1.0$.

Continuous but noisy hidden-state drift. The hidden-state drift signal exhibits frequent small-to-medium fluctuations throughout the entire decoding process. These variations often reflect local contextual adjustments or incremental representation updates, producing many peaks that do not correspond to substantive changes in reasoning intent. As shown in the top row, drift remains highly dynamic even in segments where no parametric reconfiguration occurs, indicating that drift alone is insufficient for reliably identifying when the model's active Hyper-Neurons should be updated.

Sparse but decisive predictive entropy spikes. In contrast, token-level predictive entropy remains near zero for extended spans and produces only a small number of sharp spikes, as shown in the second row. These spikes typically arise at points of increased uncertainty, such as entity boundaries or reasoning junctions. However, their sparsity makes entropy alone an unreliable indicator of sustained changes in knowledge demand, as many important semantic transitions do not necessarily coincide

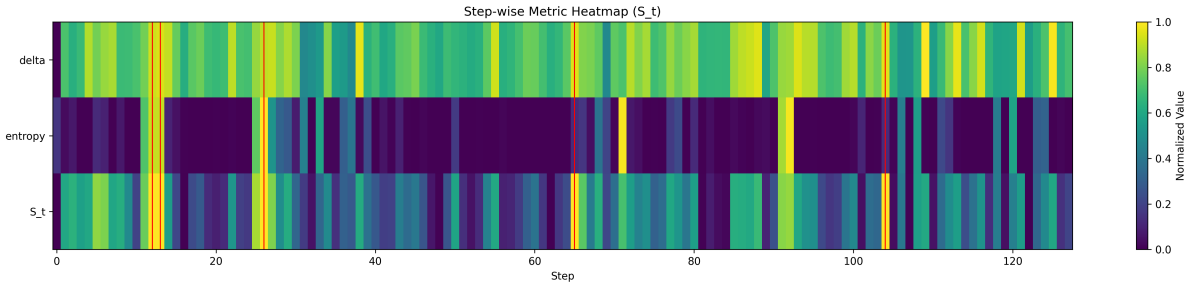

Figure 11: Step-wise visualization of decoding dynamics for a representative trajectory. The top two rows show the raw hidden-state drift Δh_t and token-level predictive entropy H_t, highlighting their distinct temporal characteristics. The bottom row presents the unified activation score S_t, computed via sliding-window normalization and mixing factor ($\lambda \approx 0.6$). Red vertical lines indicate Hyper-Neuron reconfiguration events where S_t exceeds the threshold. Notably, S_t suppresses isolated fluctuations and rises only when both signals jointly indicate a sustained shift in the model's internal reasoning state.

with high prediction uncertainty.

Joint evidence for parametric reconfiguration. The unified activation score S_t (bottom row) rises only when hidden-state drift and predictive entropy increase in a coordinated manner within their respective local contexts. Many isolated drift fluctuations and entropy spikes are deliberately ignored unless they jointly provide aligned evidence of a meaningful transition in the model's internal reasoning state. This behavior is further stabilized by sliding-window normalization, which suppresses absolute-scale noise and emphasizes relative local surges. As a result, high-base-level fluctuations (e.g., around Step 90) do not trigger redundant reconfiguration events, as they lack sufficient contrast against the local baseline.

This joint-signal mechanism also explains the empirical optimum at $\lambda \approx 0.6$. If λ were too small, making entropy dominant, the extreme sparsity of entropy spikes would cause the model to miss gradual yet important semantic transitions. Conversely, an overly drift-dominant weighting would lead to excessive reconfiguration driven by routine representation fluctuations. A slightly drift-dominant mixture balances sensitivity and selectivity, enabling NeuRAG to reconfigure its active Hyper-Neurons only when the decoding dynamics consistently indicate a genuine shift in parametric knowledge requirements.

A.8 Neural Router Training Objective

The neural router serves as a parametric alignment module that maps the LLM's internal hidden states to structured activation patterns over Hyper-Neurons in the Knowledge Hyper-Layer. Rather than performing retrieval in the traditional sense, its role is to translate model-internal reasoning states into relevance scores over LoRA-based parametric knowledge modules, enabling decoding-time Hyper-Neuron activation and fusion.

Method	Cpare	Bridge	Inf.	Cpose	Avg
w/o $\mathcal{L}_{\text{mse}}$	0.4925	0.5064	0.1473	0.0578	0.3010
w/o $\mathcal{L}_{\text{kl}}$	0.4729	0.4951	0.1587	0.0415	0.2921
NeuRAG	**0.5153**	**0.5189**	**0.1788**	**0.0786**	**0.3229**

Table 9: Ablation study of alignment loss. The backbone model is the LLaMA3.2-1B.Bold values indicate the best performance.

Training Data and Settings. The neural router is trained offline using a set of training queries that is strictly disjoint from the evaluation queries used in all reported experiments. Specifically, while evaluation is conducted on the first 300 questions of each dataset following PRAG (Su et al., 2025) and DyPRAG (Tan et al., 2025), router training uses a held-out subset of queries drawn from the same subtasks and datasets. These queries are encoded by the base LLM, and the hidden state of the final query token is used as the supervision signal for router training. This design ensures that router training does not leak evaluation information, while remaining aligned with the representations encountered during inference.

Importantly, although the router is trained using query-level representations, it is applied during decoding to token-level hidden states. This is feasible because both training and inference operate within the same hidden-state space of the base LLM, allowing the router to generalize from query-level

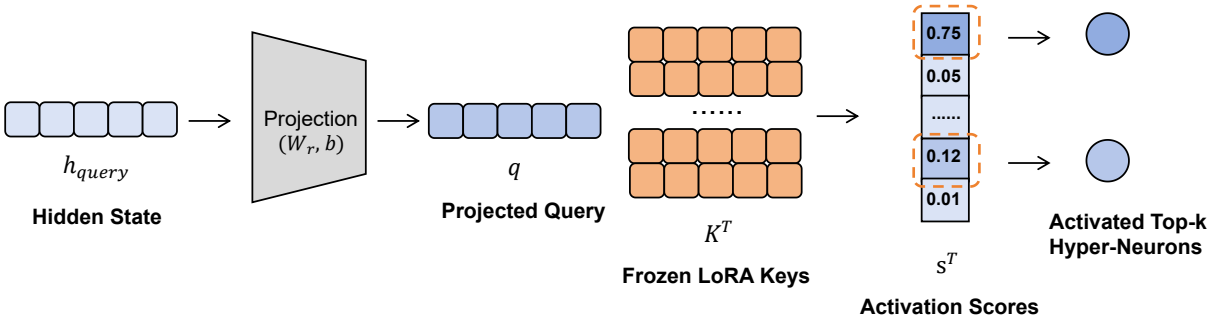

Figure 12: Architecture of the Neural Router.

semantics to intermediate reasoning states encountered during generation.

Training Objective. To obtain a stable and semantically meaningful routing behavior, we train the neural router using a hybrid alignment objective that combines two complementary supervision signals: (i) coarse lexical grounding to anchor activation locality, and (ii) semantic alignment to preserve consistency with the LLM's intrinsic representation geometry. Both signals are derived from generation-related representations of the base LLM and do not require task-specific labels.

Formally, let $h_{\text{query}} \in \mathbb{R}^d$ denote the last-layer hidden state of the final token after encoding the input query, and let $K \in \mathbb{R}^{N \times d}$ denote the frozen key matrix corresponding to N Hyper-Neurons. The neural router applies a learnable linear projection $W_r \in \mathbb{R}^{d \times d}$ with bias $b \in \mathbb{R}^d$ to compute activation logits:

$$s = (h_{\text{query}} W_r + b) K^\top. \tag{14}$$

Lexical Anchoring via MSE. To prevent degenerate or overly diffuse activation patterns, we introduce a coarse lexical anchoring signal derived from sparse retrieval during the document parameterization stage. Following the same retrieval protocol as PRAG (Su et al., 2025), we identify the top-3 documents ranked by BM25 for each training query and assign them a target activation value of 1.0, with all remaining candidates set to 0, forming a binary supervision vector $y \in \{0, 1\}^N$. We interpret the sigmoid-transformed logits $\sigma(s)$ as activation intensities and minimize the mean squared error:

$$\mathcal{L}_{\text{mse}} = \frac{1}{N} \|\sigma(s) - y\|^2. \tag{15}$$

This term does not enforce exact retrieval correctness, but provides a coarse lexical prior that stabilizes the Hyper-Neuron activation space.

Semantic Alignment via KL Divergence. To align Hyper-Neuron activations with the LLM's internal semantic structure, we derive a soft teacher distribution directly from the base model's hidden-state geometry. Specifically, we compute similarity scores between h_{query} and the Hyper-Neuron keys K using frozen LLM representations, and convert them into a teacher distribution p_T via softmax. The neural router's output distribution $p_S = \text{softmax}(s)$ is then aligned with this semantic prior by minimizing the Kullback–Leibler divergence:

$$\mathcal{L}_{\text{kl}} = \text{KL}(p_T \,\|\, p_S). \tag{16}$$

This objective preserves the relative semantic geometry induced by the base LLM, ensuring consistency between routing behavior and the model's native reasoning space.

Joint Training Objective. The final training objective combines both components:

$$\mathcal{L} = \lambda_{\text{mse}} \mathcal{L}_{\text{mse}} + \lambda_{\text{kl}} \mathcal{L}_{\text{kl}}. \tag{17}$$

The two terms play complementary roles: $\mathcal{L}_{\text{mse}}$ anchors activation locality through coarse lexical grounding, while $\mathcal{L}_{\text{kl}}$ enforces semantic consistency with the LLM's internal representations. Together, they yield a neural router that produces stable, interpretable, and semantically aligned Hyper-Neuron activation patterns suitable for decoding-time parametric knowledge augmentation.

Empirical Validation. We empirically validate the necessity of this joint alignment strategy through ablation studies on 2WikiMultihopQA (Table 9). Removing either objective consistently degrades performance, confirming that both lexical anchoring and semantic alignment are required for stable and effective Hyper-Neuron activation.

A finer-grained analysis reveals their distinct functional roles:

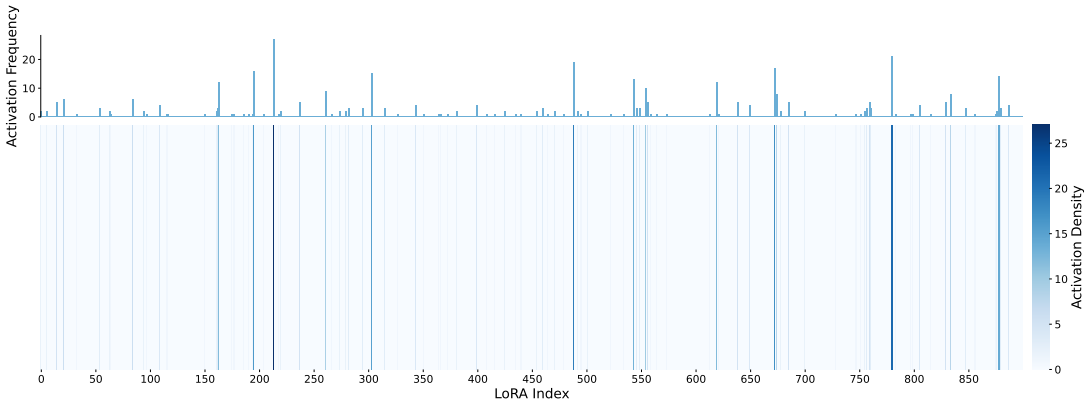

Figure 13: Hyper-Neuron activation statistics on ComplexWebQuestions using LLaMA3.2-1B-Instruct. The histogram (top) shows broad utilization across the Hyper-Neuron space, while the heatmap (bottom) reveals sparse, query-specific activation bands, indicating both diversity and specialization.

- **Lexical Anchoring for Structured Knowledge Transitions.** In the Bridge category, which requires reliable transitions across intermediate entities, removing $\mathcal{L}_{\text{mse}}$ leads to the largest performance drop. This indicates that coarse lexical grounding is critical for stabilizing Hyper-Neuron activation when decoding involves entity-level shifts and structured knowledge transitions.

- **Semantic Alignment for Multi-Step Reasoning Composition.** In the Inference and Compose categories, which demand multi-step reasoning and evidence synthesis, removing $\mathcal{L}_{\text{kl}}$ causes severe degradation. This demonstrates that semantic alignment is essential for preserving the LLM's internal reasoning geometry during decoding-time parametric reconfiguration.

Overall, NeuRAG achieves the strongest performance by jointly balancing these two alignment signals. This enables the neural router to function as a stable parametric alignment mechanism that maps evolving hidden-state dynamics to coherent and effective Hyper-Neuron activation during end-to-end decoding.

A.9 Sliding-Window Normalization

For completeness, we present the exact form of the sliding-window normalization used to obtain $\widehat{\Delta h}_t$ and $\widehat{H}_t$ in the unified retrieval score. Let $\mathcal{W}_\Delta(t)$ and $\mathcal{W}_H(t)$ denote the most recent window size(we set it to 10 steps) observations of Δh and H, respectively. The normalized quantities described in §3.3 are given by

$$\begin{cases} \widehat{\Delta h}_t = \frac{\Delta h_t - \min \mathcal{W}_\Delta(t)}{\max \mathcal{W}_\Delta(t) - \min \mathcal{W}_\Delta(t) + \varepsilon}, \\ \widehat{H}_t = \frac{H_t - \min \mathcal{W}_H(t)}{\max \mathcal{W}_H(t) - \min \mathcal{W}_H(t) + \varepsilon}. \end{cases} \tag{18}$$

This min–max normalization ensures that both signals are placed on a comparable scale and remain stable under local fluctuations during generation.

A.10 Analysis of Hyper-Neuron Activation

To examine whether neural router learns a structured and semantics-aware Hyper-Neuron activation policy rather than degenerating into trivial static or random activation, we analyze the activation patterns produced by the neural router on the CWQ dataset using LLaMA3.2-1B-Instruct, as shown in Figure 13.

Activation Balance and Coverage. Based on the top-3 relevant Hyper-Neurons identified for each of the 300 evaluation questions (totaling 900 Hyper-Neurons), the histogram in the top panel shows that activations are broadly distributed across the entire index range (0–899), with no dominant or collapsed modes. This balanced coverage indicates that the neural router does not overspecialize on a small subset of parametric modules, but instead effectively utilizes the available Hyper-Neuron capacity to support diverse information needs across queries.

Structured Specialization Patterns. The heatmap in the bottom panel reveals a sparse, vertically banded structure, indicating that for semantically related queries, the neural router consistently activates similar subsets of Hyper-Neurons. These stable activation patterns suggest that NeuRAG has learned to map fine-grained reasoning states to cor-

Table 10: Latency Breakdown for LLaMA-3.2-1B (Unit: Seconds)

Stage	Standard RAG	NeuRAG
Retrieval	0.000012	0.000013
LoRA Mount	—	0.181606
LLM Generate	0.435609	0.351311
E2E Total	**0.438261**	**0.665589**

Table 11: Latency Breakdown for LLaMA-3-8B (Unit: Seconds)

Stage	Standard RAG	NeuRAG
Retrieval	0.000012	0.000013
LoRA Mount	—	0.395754
LLM Generate	0.801864	1.259607
E2E Total	**0.805155**	**1.822367**

responding parametric capabilities, rather than performing ad hoc or noisy reconfiguration. Together, these observations confirm that the neural router supports selective, structured, and semantics-aware Hyper-Neuron activation, enabling effective parametric adaptation during decoding.

A.11 Detailed Inference Efficiency Analysis

To further validate the inference efficiency of NeuRAG, we conduct detailed latency breakdown experiments on the ComplexWebQuestions (CWQ) dataset.

Experimental Setup The experiments are executed on NVIDIA RTX 3090 GPUs. We evaluate two representative model scales: LLaMA-3.2-1B and LLaMA-3-8B. The latency is measured across four distinct stages:

- **Retrieval**: The duration from receiving a query to obtaining retrieved documents.
- **LoRA Mount**: The time required to load the corresponding Hyper-Neuron and complete the parameter merging or adapter setting (e.g., `set_adapter`), reaching a state ready for generation.
- **LLM Generate**: The time elapsed during the `model.generate()` execution until the final token is decoded.
- **E2E Total**: The total end-to-end wall-clock time from initial input to the final text response.

Latency Breakdown Results Table 10 and Table 11 present the measured latency (in seconds) for LLaMA-3.2-1B and LLaMA-3-8B, respectively.

A.12 Prompts used in this work

- **Chain-of-Thought** The following prompt utilizes few-shot demonstrations to guide the model in reasoning step-by-step, combining retrieved context with parametric knowledge.

Prompt Template with CoT

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. Here are some examples about how to answer the questions.

Question: q_1
Answer: a_1

Question: q_2
Answer: a_2

...
Question: q_i
Answer: a_i

Here are some reference.
Passage 1: {passage_1}
Passage 2: {passage_2}
...
Passage i: {passage_i}

Let's think step by step. Answer the questions in the same format as above.
Question: {question}
Answer: {answer}

- **Direct Context Integration** The following prompt instructs the model to answer the question directly by synthesizing the provided reference passages with its internal knowledge.

Prompt Template without CoT

You should answer the question by referring to the knowledge provided below and integrating your own knowledge.

Passage 1: {passage_1}
Passage 2: {passage_2}
...
Passage i: {passage_i}

Question: {question}
The answer is {answer}

A.13 Settings

Table 12 summarizes the activation settings used for all models and datasets, including the mixing factor λ, the activation threshold τ, the number of activated Hyper-Neurons (Top-k) each time. These hyperparameters jointly control the timing, sensitivity, and capacity of Hyper-Neuron reconfiguration during inference, and are kept consistent across different LLMs to ensure fair comparison.

Activation Settings					
LLaMA3.2-1B-Instruct					
	2WikiMultihopQA	HotpotQA	IIRC	StrategyQA	ComplexWebQuestions
Mixing Factor λ	0.6	0.6	0.6	0.6	0.6
Trigger Threshold τ	0.9	0.95	0.95	0.95	0.8
Top-k	3	1	1	1	1
Qwen2.5-1.5B-Instruct					
	2WikiMultihopQA	HotpotQA	IIRC	StrategyQA	ComplexWebQuestions
Mixing Factor λ	0.6	0.6	0.6	0.6	0.6
Trigger Threshold τ	0.9	0.95	0.95	0.95	0.8
Top-k	3	1	1	1	1
LLaMA3-8B-Instruct					
	2WikiMultihopQA	HotpotQA	IIRC	StrategyQA	ComplexWebQuestions
Mixing Factor λ	0.6	0.6	0.6	0.6	0.6
Trigger Threshold τ	0.9	0.95	0.95	0.95	0.8
Top-k	1	1	1	1	1

Table 12: Activation settings for all models and datasets.