THINK LESS, RETRIEVE EARLIER: SELF-SPECULATIVE RETRIEVAL FOR EFFICIENT RETRIEVAL-AUGMENTED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-Augmented Generation (RAG) enhances LLM reasoning with external knowledge, but the sequential process of iterative retrieval and generation introduce substantial end-to-end latency. Existing acceleration methods primarily mitigate this overhead by limiting retrieval calls, reducing retrieval waiting, compressing retrieved contexts, or simplifying LLM reasoning. However, these methods typically reduce latency from either reasoning or retrieval perspectives, with little attention to both. We empirically observe that retrieval queries derived from intermediate-layer token reasoning are similar to those produced by full-layer LLM decoding, which is promising for invoking retrieval earlier while avoiding unnecessary full-layer computation. Inspired by this, we propose **FastRAG**, a new self-speculative retrieval framework that accelerates query generation through dynamic early-exit reasoning. **FastRAG** introduces an *exit trigger* to determine the optimal exit layer and a *token calibrator* to correct intermediate-layer token predictions, enabling earlier speculative query generation with less layer-wise computation. Next, we introduce SpecGRPO, which extends GRPO to optimize speculative-query reliability and early-exit efficiency. At inference, query entropy enables self-verification of speculative queries: low-entropy queries are directly accepted, while high-entropy queries fall back to full-layer LLM query generation, thereby reducing both retrieval waiting time and reasoning latency. Experiments on seven retrieval-augmented reasoning benchmarks demonstrate that **FastRAG** reduces the average end-to-end inference latency from 3.673 s to 1.382 s compared with Search-R1, while improving the average EM from 43.1% to 47.3%.

1 INTRODUCTION

Retrieval-augmented generation (RAG) (Lewis et al., 2020; Guu et al., 2020; Borgeaud et al., 2022; Singh et al., 2025) improves large language models (LLMs) by incorporating information retrieved from an external database. This paradigm has become particularly important for knowledge-intensive reasoning, where the LLM repeatedly formulates retrieval queries, searches relevant knowledge, and integrates the retrieved information into subsequent reasoning (Trivedi et al., 2023; Yao et al., 2022). However, in standard RAG (Fig. 1), retrieval begins only after the LLM generates a query, leading to sequential reasoning-retrieval and introducing inevitable retrieval waiting time (Zhang et al., 2024; Jiang et al., 2025).

A variety of approaches have been developed to accelerate RAG by reducing retrieval calls, input context, retrieval waiting, or LLM reasoning. Retrieval-call control methods (Asai et al., 2024; Jeong et al., 2024; Wu et al., 2026; Su et al., 2024) avoid unnecessary searches by determining whether additional external information is needed, but each retrieval still interrupts the reasoning process. Context compression methods (Xu et al., 2024; Xu et al., 2025; Cheng et al., 2024) remove redundant retrieved content before feeding it into the LLM, reducing the context-processing cost, but they do not eliminate the retrieval-induced waiting latency before subsequent reasoning. Retrieval-wait reduction methods (Jiang et al., 2025; Yang et al., 2026; Zhang et al., 2024) start retrieval earlier and overlap the retrieval process with ongoing LLM computation, thereby reducing retrieval-induced waiting, while the reasoning cost of generating retrieval queries is overlooked. Reasoning acceleration methods (Zhuang et al., 2024; Zheng & Worring, 2026) reduce reasoning latency by replacing costly

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

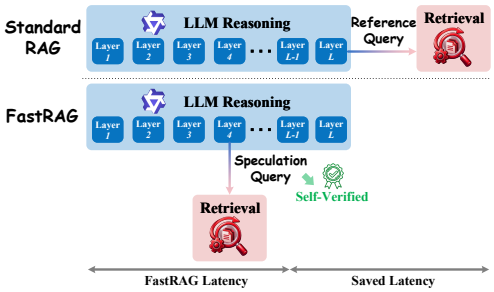


Figure 1: Comparison between Standard RAG and **FastRAG**, which uses intermediate-layer reasoning to shorten end-to-end latency.

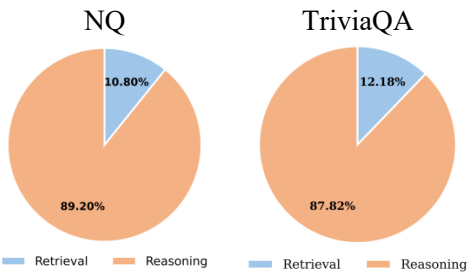


Figure 2: Latency distribution of retrieval and reasoning on NQ and TriviaQA with Qwen2.5-3B-Instruct.

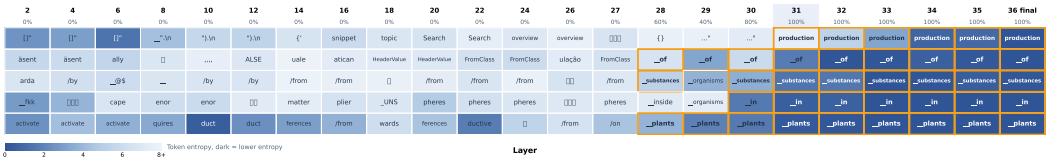
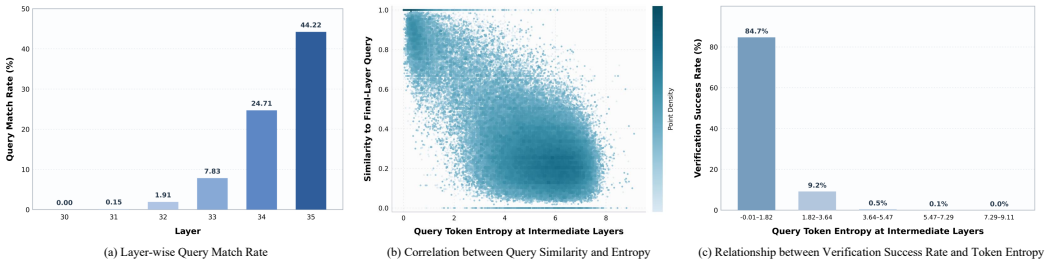


Figure 3: Layer-wise evolution of token predictions and entropy during retrieval-query generation. Deeper intermediate-layer tokens show lower entropy while becoming consistent with final-layer tokens, revealing redundant layer computation in full-layer decoding.



(a) Layer-wise Query Match Rate (b) Correlation between Query Similarity and Entropy (c) Relationship between Verification Success Rate and Token Entropy

Figure 4: **(a)** Layer-wise query match rate between intermediate-layer and final-layer queries. **(b)** Correlation between intermediate-layer query-token entropy and similarity to the final-layer query. **(c)** Verification success rate across different query-token entropy intervals.

LLM reasoning with lightweight models or latent representations, skipping unnecessary reasoning steps, or reusing KV states to reduce repeated prefill. However, prior works have mainly approached RAG acceleration from either the reasoning or retrieval perspective, with limited attention to jointly reducing reasoning and retrieval latency to improve end-to-end efficiency.

In this work, we investigate the latency bottleneck of RAG by examining LLM reasoning latency, retrieval waiting latency, and the layer-wise evolution of retrieval-query tokens. Our empirical analysis reveals three key findings. First, the reasoning process dominates end-to-end RAG latency (Fig. 2), indicating that retrieval-centric acceleration alone leaves substantial computational overhead unaddressed. Second, retrieval queries generated from intermediate-layer representations often remain consistent with those obtained through full-layer decoding (Fig. 3 & Fig. 4a), revealing considerable redundant layer computation before retrieval. Third, intermediate-layer query entropy is correlated with verification outcomes: low-entropy queries tend to exhibit higher similarity to final-layer queries and higher verification success rates (Fig. 4bc). These observations suggest that intermediate-layer reasoning can be used to generate reliable retrieval queries earlier, while query entropy provides an effective criterion for query self-verification. Together, these findings motivate jointly reducing redundant LLM computation and retrieval-induced waiting latency to lower end-to-end RAG latency.

Motivated by these observations, we propose **FastRAG** (Fig. 1), a self-speculative retrieval approach designed to jointly reduce reasoning and retrieval waiting latency. Specifically, we propose an *exit trigger* that dynamically determines the appropriate exit layer for each token, enabling speculative queries to be generated earlier from intermediate-layer reasoning with lower computational cost. Meanwhile, we introduce a *token calibrator* to correct intermediate-layer token predictions, improving

the reliability of speculative queries. Based on the token entropy of speculative queries, **FastRAG** directly accepts low-entropy queries, while high-entropy queries fall back to full-layer LLM query generation, avoiding unnecessary full-layer computation for confident queries. Finally, we develop Speculative Group Relative Policy Optimization (SpecGRPO) to jointly optimize query consistency, early exits, and intermediate-layer prediction quality. Our contributions are summarized as follows:

- We reveal that intermediate-layer reasoning can generate retrieval queries consistent with full-layer reasoning, providing a new perspective for RAG acceleration with the potential to jointly reduce both LLM reasoning latency and retrieval waiting latency.
- We propose **FastRAG**, a self-speculative retrieval framework equipped with an *exit trigger* and a *token calibrator*. By dynamically adapting token computation depth and refining intermediate-layer predictions, it enables early and reliable retrieval-query generation while reducing unnecessary LLM reasoning computation.
- We introduce SpecGRPO, which formulates speculative query generation as a unified reinforcement learning objective that jointly encourages accurate query matching, shallower token exits, and reliable intermediate-layer predictions.
- Experiments across seven benchmarks demonstrate that **FastRAG** reduces the average end-to-end inference latency from 3.673 s to 1.382 s (-62.4%) compared with Search-R1, while improving the average EM from 43.1% to 47.3% ($+4.2$ points).

2 PRELIMINARY

Notation. Consider retrieval round m of an LLM with L layers. Let $\mathbf{h}_{m,t}^{(\ell)} \in \mathbb{R}^d$ denote the hidden state at layer ℓ for predicting token t, where $\ell \in \{1, \ldots, L\}$. In standard RAG, each generated token is processed through all L layers, and the retrieval query $q_m^{\texttt{ref}}$ is produced through full-layer autoregressive decoding (Jin et al., 2025). Once the query is completed, retrieval is invoked and the retrieved result is then incorporated into the context for the next round reasoning.

Problem Formalization. Standard RAG introduces two major latency components: the LLM reasoning latency required to generate the retrieval query and the waiting latency caused by retrieval before the next reasoning round (Zhang et al., 2024). Given an LLM with L Transformer layers, sequence length n, hidden dimension d, and FFN intermediate dimension $d_{\texttt{ffn}}$, the computational cost of a forward pass can be approximated as $\text{FLOPs} = L(4nd^2 + 2n^2d + 2ndd_{\texttt{ffn}})$ (Narayanan et al., 2021). For fixed n, d, and $d_{\texttt{ffn}}$, this cost grows linearly with L, and full-layer decoding requires each generated token to pass through all layers. At round m, the resulting reasoning latency is

$$T_m^{\texttt{LLM}} = \sum_{t=1}^{T_m} \sum_{\ell=1}^{L} \delta_{m,t}^{(\ell)}, \tag{1}$$

where T_m is the number of generated tokens before retrieval and $\delta_{m,t}^{(\ell)}$ denotes the latency of processing token t at layer ℓ. As L increases, more layer-wise computation leads to higher LLM reasoning latency. Moreover, standard RAG executes retrieval and subsequent reasoning sequentially, meaning that the next reasoning round starts only after the retrieval results are returned. (Jiang et al., 2025). Accordingly, the waiting latency at retrieval round m is defined as $T_m^{\texttt{wait}} = t_m^{\texttt{next}} - t_m^{\texttt{query}}$, where $t_m^{\texttt{query}}$ denotes when retrieval starts and $t_m^{\texttt{next}}$ denotes when the next reasoning round begins. These two latency components directly affect end-to-end RAG efficiency.

3 METHODOLOGY

Motivated by the above observations, we propose **FastRAG** to reduce both query generation and retrieval waiting latency. As illustrated in Fig. 5, it consists of an *exit trigger* (§3.1) to identify the optimal exit layer and a *token calibrator* (§3.2) to refine intermediate-layer token predictions, enabling earlier speculative query generation (§3.3). Subsequently, we introduce a SpecGRPO optimization scheme (§3.4) that incorporates speculative-query consistency rewards, early-exit rewards, and token-calibrator KL regularization to jointly encourage accurate speculation, earlier exits, and reliable intermediate-layer token predictions.

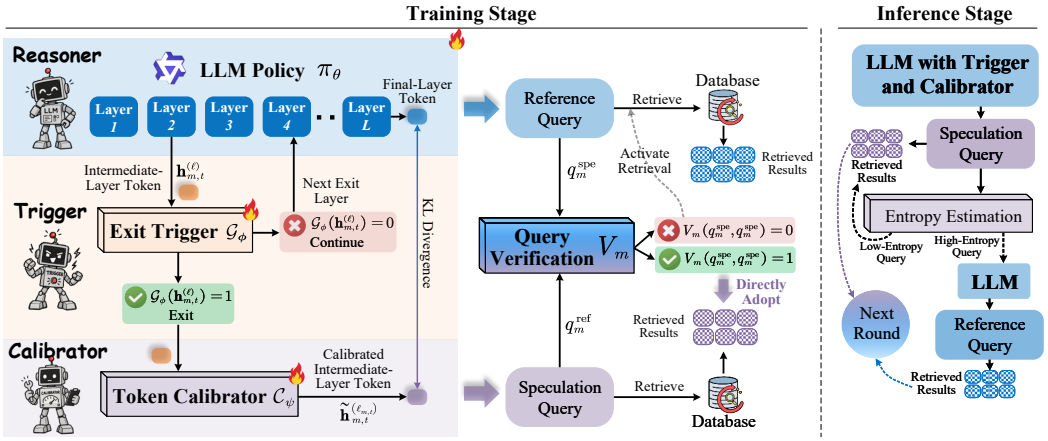


Figure 5: **Overview of FastRAG.** *Left:* Training stage, where the *exit trigger* selects the exit layer for each token, the *token calibrator* refines intermediate-layer token predictions, and speculative queries are verified against full-layer reference queries. *Right:* Inference stage, where entropy-guided query selection directly adopts low-entropy speculative queries, while high-entropy queries fall back to full-layer LLM query generation before proceeding to the next round, alleviating both LLM reasoning latency and retrieval-induced waiting latency.

3.1 Exit Trigger

Standard decoding processes each token through all LLM layers. Our observation suggests that using intermediate-layer tokens for reasoning can still generate accurate retrieval queries. To this end, we introduce an *exit trigger*, implemented as a binary adapter $\mathcal{G}_\phi$, to decide between EXIT and CONTINUE. Let $\mathbf{h}^{(\ell)}_{m,t}$ denote the hidden state at layer ℓ for predicting token t at round m. We define a set of candidate exit layers, $\mathcal{E} = \{\ell_{\min}, \ell_{\min}+\Delta, \ldots, \ell_{\max}\}$, where $\ell_{\max} < L$ and $\Delta = 2$. At each candidate layer $\ell \in \mathcal{E}$, the *exit trigger* takes the current hidden state and a learnable layer-position embedding $\mathbf{e}_\ell$ as input and predicts the exit probability

$$p^{(\ell)}_{m,t} = \sigma\Big(\mathcal{G}_\phi\left(\mathbf{h}^{(\ell)}_{m,t}, \mathbf{e}_\ell\right)\Big), \tag{2}$$

where $\sigma(\cdot)$ denotes the sigmoid function. The exit action is sampled as $a^{(\ell)}_{m,t} \sim \text{Bernoulli}\left(p^{(\ell)}_{m,t}\right)$, where $a^{(\ell)}_{m,t} = 1$ denotes EXIT and $a^{(\ell)}_{m,t} = 0$ denotes CONTINUE. If CONTINUE is selected, computation proceeds to the next candidate layer; if EXIT is selected, computation stops at the current layer and the token is decoded from its intermediate representation. The *exit trigger* is shared across all candidate layers, with a learnable layer-position embedding distinguishing different layers. Exit depth is determined independently for each token and varies across the generated sequence. The training recipe for the *exit trigger* is detailed in §3.4.

3.2 Token Calibrator

Intermediate hidden states are not explicitly optimized for prediction through the final LM head. Accordingly, we introduce a *token calibrator*, a lightweight adapter $\mathcal{C}_\psi$, to refine intermediate-layer representations for token prediction. For an intermediate exit layer $\ell_{m,t}$, we first calibrate its hidden representation $\widetilde{\mathbf{h}}^{(\ell_{m,t})}_{m,t}$ and then obtain the corresponding token distribution $\mathbf{p}^{(\ell_{m,t})}_{m,t}$ as

$$\widetilde{\mathbf{h}}^{(\ell_{m,t})}_{m,t} = \mathcal{C}_\psi\left(\mathbf{h}^{(\ell_{m,t})}_{m,t}, \mathbf{e}_\ell\right), \qquad \mathbf{p}^{(\ell_{m,t})}_{m,t} = \text{Softmax}\Big(\mathbf{W}_{\text{LM}}\,\text{Norm}\left(\widetilde{\mathbf{h}}^{(\ell_{m,t})}_{m,t}\right)\Big). \tag{3}$$

where $\mathbf{W}_{\text{LM}}$ denotes the shared LM head and $\text{Norm}(\cdot)$ denotes the final normalization layer. Once EXIT is triggered at layer $\ell_{m,t}$, the corresponding intermediate representation is refined by the *token calibrator* before decoding. The *token calibrator* is shared across all candidate exit layers and uses the layer embedding to distinguish layer-specific token calibration. By narrowing the prediction gap between intermediate and final layers, it improves the reliability of early-exit token predictions. The training recipe for the *token calibrator* is detailed in §3.4.

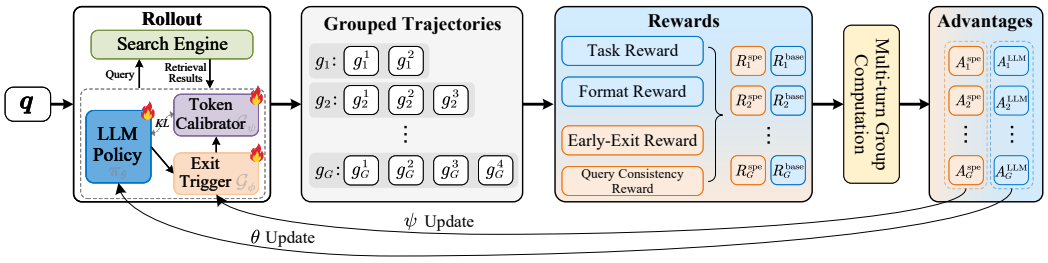


Figure 6: **Optimization for `FastRAG`**. SpecGRPO enables the LLM policy equipped with the *Exit Trigger* and *token calibrator* to interact with the search engine and generate grouped multi-turn trajectories. It combines task and format rewards with early-exit and query-consistency rewards to optimize the LLM policy and *exit trigger*, while the *token calibrator* is optimized by KL divergence between the calibrated intermediate-layer token distribution and the corresponding final-layer token distribution.

3.3 Self-Speculative Retrieval and Verification

Training Stage. For each training question, we sample a group of G rollout trajectories, $\mathcal{B} = \{g_1, g_2, \ldots, g_G\}$. At round m, the full-layer path generates a reference query $q_m^{\texttt{ref}}$, while the intermediate-layer path generates an earlier speculative query $q_m^{\texttt{spe}}$ from dynamically selected exit layers, each conditioned on its own reasoning prefix. The speculative query triggers retrieval in advance. Once $q_m^{\texttt{ref}}$ is available, $q_m^{\texttt{spe}}$ is verified against it. The speculative retrieval results are reused if verification succeeds; otherwise, retrieval is performed with $q_m^{\texttt{ref}}$. For rollout i and round m, the query verification is

$$V_{i,m} = \mathbb{I}\big[\mathrm{Sim}(q_{i,m}^{\texttt{spe}}, q_{i,m}^{\texttt{ref}}) \geq \tau\big], \tag{4}$$

where $\mathbb{I}[\cdot]$ is the indicator function, Sim is the query similarity function described in the Appendix A.3, and τ denotes the query similarity threshold.

Inference Stage. To avoid unnecessary full-layer query generation, we use the entropy of speculative-query tokens to determine whether a full-layer fallback is required. Let $\mathcal{Q}_m$ denote the set of token positions in the speculative query $q_m^{\texttt{spe}}$. For each query token $t \in \mathcal{Q}_m$, we measure its uncertainty using the entropy of the token distribution:

$$H_{m,t} = -\sum_{v \in \mathcal{V}} \mathbf{p}_{m,t}^{(\ell_{m,t})}(v) \log \mathbf{p}_{m,t}^{(\ell_{m,t})}(v), \qquad U_m = \frac{1}{|\mathcal{Q}_m|} \sum_{t \in \mathcal{Q}_m} H_{m,t}. \tag{5}$$

where $\mathcal{V}$ denotes the vocabulary and U_m denotes the average entropy of the speculative query. If $U_m \leq \gamma$, where γ denotes the entropy threshold, $q_m^{\texttt{spe}}$ is directly accepted; otherwise, **`FastRAG`** falls back to full-layer LLM query generation and uses the resulting reference query $q_m^{\texttt{ref}}$ for retrieval.

3.4 SpecGRPO Optimization

As shown in Fig. 6, we optimize **`FastRAG`** with SpecGRPO, which extends GRPO (Shao et al., 2024) with speculative-query consistency rewards, early-exit rewards, and token-calibrator KL regularization. For rollout i, let $\mathcal{M}_i$ denote the set of retrieval rounds and $M_i = |\mathcal{M}_i|$.

Speculative-Query Consistency Rewards. To encourage accurate speculative retrieval, we reward the speculative query that matches its reference query and penalize mismatched queries. According to the verification result $V_{i,m}$ defined in Eq. 4, the speculative-query consistency reward is defined as

$$R_i^{\texttt{con}} = \frac{1}{M_i} \sum_{m \in \mathcal{M}_i} (2V_{i,m} - 1), \tag{6}$$

where $V_{i,m} = 1$ indicates successful query verification and $V_{i,m} = 0$ indicates a mismatch. Thus, each matched speculative query receives a reward of $+1$, while each mismatched query receives a penalty of -1. $R_i^{\texttt{con}}$ encourages the intermediate-layer path to generate speculative queries consistent with the reference queries.

Early-Exit Rewards. Accurate speculative queries should also be generated with less layer-wise computation. Let $T_{i,m}$ denote the number of tokens generated by the intermediate-layer path at round m of rollout i. We define $D_{i,m}$ as the average early-exit score at round m and $R_i^{\texttt{exit}}$ as the early-exit reward over all retrieval rounds of rollout i:

$$D_{i,m} = \frac{1}{T_{i,m}} \sum_{t=1}^{T_{i,m}} \frac{L - \ell_{i,m,t}}{L}, \qquad R_i^{\texttt{exit}} = \frac{1}{M_i} \sum_{m \in \mathcal{M}_i} V_{i,m} D_{i,m}, \tag{7}$$

where $\ell_{i,m,t}$ denotes the exit layer of token t, and L denotes the total number of LLM layers. $D_{i,m}$ reflects the average computation. A larger $D_{i,m}$ indicates that tokens exit at shallower layers on average. The query verification term $V_{i,m}$ ensures that the early-exit reward is granted only when the speculative query matches its reference query, discouraging excessively aggressive early exits that may lead to incorrect speculative queries.

Token-Calibrator KL Divergence. The *token calibrator* is optimized by aligning its intermediate-layer token distribution with the final-layer token distribution. For each token generated from an intermediate exit layer, we obtain the corresponding final-layer token distribution $\mathbf{p}_{i,m,t}^{(L)}$ and compute the KL divergence loss $\mathcal{L}_{\texttt{cal}}$ between $\mathbf{p}_{i,m,t}^{(L)}$ and the intermediate-layer token distribution $\mathbf{p}_{i,m,t}^{(\ell_{i,m,t})}$:

$$\mathcal{L}_{\texttt{cal}} = \frac{1}{|\mathcal{I}|} \sum_{(i,m,t) \in \mathcal{I}} D_{\mathrm{KL}} \left(\mathbf{p}_{i,m,t}^{(L)} \,\|\, \mathbf{p}_{i,m,t}^{(\ell_{i,m,t})} \right), \tag{8}$$

where $\mathcal{I}$ denotes the set of tokens generated from intermediate exit layers. The final-layer distribution $\mathbf{p}_{i,m,t}^{(L)}$ is treated as a fixed target, and gradients are not propagated through it, preventing the calibration objective from altering the final-layer distribution and ensuring the optimization focuses on improving the intermediate-layer prediction.

Optimization. The optimization of **FastRAG** consists of three components: the *exit trigger* is optimized by speculative-query consistency and early-exit rewards, the LLM by task and format rewards together with positive speculative feedback, and the *token calibrator* by KL divergence.

We combine the speculative-query consistency and early-exit rewards as $R_i^{\texttt{spe}} = \lambda_{\texttt{con}} R_i^{\texttt{con}} + \lambda_{\texttt{exit}} R_i^{\texttt{exit}}$. Following (Guo et al., 2025), the task and format rewards are defined as $R_i^{\texttt{task}} = \mathbb{I}[\widehat{a}_i = a^*]$ and $R_i^{\texttt{fmt}} = \mathbb{I}[\mathbf{y}_i \in \mathcal{F}]$, respectively, and combined as $R_i^{\texttt{base}} = \lambda_{\texttt{task}} R_i^{\texttt{task}} + \lambda_{\texttt{fmt}} R_i^{\texttt{fmt}}$. For a group of G rollouts sampled from the same question, the base and speculative advantages are normalized as

$$A_i^{\texttt{base}} = \frac{R_i^{\texttt{base}} - \mu_{\texttt{base}}}{\sigma_{\texttt{base}} + \epsilon}, \qquad A_i^{\texttt{spec}} = \frac{R_i^{\texttt{spe}} - \mu_{\texttt{spe}}}{\sigma_{\texttt{spe}} + \epsilon}. \tag{9}$$

We optimize the *exit trigger* using $A_i^{\texttt{spe}}$. For LLM optimization, we combine the base advantage with positive speculative feedback as $A_i^{\texttt{LLM}} = A_i^{\texttt{base}} + \delta_i \max(A_i^{\texttt{spe}}, 0)$, where $\delta_i = 1$ if all speculative queries in rollout i pass verification and $\delta_i = 0$ otherwise. This promotes successful speculation while preventing interference from incorrect speculative feedback. The *token calibrator* is optimized using $\mathcal{L}_{\texttt{cal}}$ defined in Eq. 8.

Following GRPO (Shao et al., 2024), we optimize the LLM policy and the *exit trigger* with separate policy objectives. The clipped surrogate term is

$$\mathcal{H}(r, A) = \min(rA, \mathrm{clip}(r, 1 - \epsilon_c, 1 + \epsilon_c)A), \tag{10}$$

where r denotes the importance ratio, A the corresponding advantage, and ϵ_c the clipping coefficient. We use trajectory-level advantages for token- and action-level optimization, where $A_i^{\texttt{LLM}}$ and $A_i^{\texttt{spe}}$ are shared across all generated tokens and exit decisions within rollout i, respectively. The importance ratios are

$$r_{i,m,t}^{\texttt{LLM}} = \frac{\pi_\theta(y_{i,m,t} \mid \mathbf{y}_{i,m,<t})}{\pi_{\theta_{\text{old}}}(y_{i,m,t} \mid \mathbf{y}_{i,m,<t})}, \qquad r_{i,m,t}^{\texttt{spe},(\ell)} = \frac{\pi_\phi\left(a_{i,m,t}^{(\ell)} \mid \mathbf{h}_{i,m,t}^{(\ell)}, \mathbf{e}_\ell\right)}{\pi_{\phi_{\text{old}}}\left(a_{i,m,t}^{(\ell)} \mid \mathbf{h}_{i,m,t}^{(\ell)}, \mathbf{e}_\ell\right)}. \tag{11}$$

where π_θ and $\pi_{\theta_{\text{old}}}$ denote the current and old LLM policies, and π_ϕ and $\pi_{\phi_{\text{old}}}$ the current and old *exit trigger*. $y_{i,m,t}$ is the generated token, $a_{i,m,t}^{(\ell)}$ the EXIT/CONTINUE action, and $\mathbf{h}_{i,m,t}^{(\ell)}$ and $\mathbf{e}_\ell$ the

hidden state and layer-position embedding. The optimization objectives are

$$\mathcal{J}_{\texttt{LLM}} = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{T}_i|}\sum_{(m,t)\in\mathcal{T}_i}\mathcal{H}\big(r_{i,m,t}^{\texttt{LLM}}, A_i^{\texttt{LLM}}\big)\right] - \beta\mathcal{L}_{\texttt{ref}},$$
$$\mathcal{J}_{\texttt{spe}} = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{D}_i|}\sum_{(m,t,\ell)\in\mathcal{D}_i}\mathcal{H}\Big(r_{i,m,t}^{\texttt{spe},(\ell)}, A_i^{\texttt{spe}}\Big)\right]. \tag{12}$$

where $\mathcal{T}_i$ denotes the generated token positions, $\mathcal{D}_i$ the sampled EXIT/CONTINUE decisions, β the strength of reference-policy regularization, and $\mathcal{L}_{\texttt{ref}}$ the KL divergence between the current and fixed reference policies. Finally,

$$\mathcal{L}_{\texttt{FastRAG}} = -\mathcal{J}_{\texttt{LLM}} - \lambda_{\texttt{spe}}\mathcal{J}_{\texttt{spe}} + \lambda_{\texttt{cal}}\mathcal{L}_{\texttt{cal}}, \tag{13}$$

where $\lambda_{\texttt{spe}}$ and $\lambda_{\texttt{cal}}$ control the contributions of the *exit trigger* and *token calibrator*, respectively. $\mathcal{J}_{\texttt{LLM}}$, $\mathcal{J}_{\texttt{spe}}$, and $\mathcal{L}_{\texttt{cal}}$ optimize the LLM parameters θ, *exit trigger* parameters ϕ, and *token calibrator* parameters ψ, respectively.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Implementation Details. We conduct experiments on Qwen2.5-3B/7B-Instruct (Yang et al., 2024). Following Search-R1 (Jin et al., 2025), we use the training sets of Natural Questions (NQ) (Kwiatkowski et al., 2019) and HotpotQA (Yang et al., 2018) for training. We use the 2018 Wikipedia dump (Karpukhin et al., 2020) as the retrieval corpus and E5 (Wang et al., 2022) as the retriever. The rollout group size for each prompt is set to 5. We train all models using the verl library (Sheng et al., 2025) with a rollout temperature of 1.0. For evaluation, we report the exact match (EM) score, the average latency per question (Lat.), efficiency defined as $\mathrm{Eff.} = \mathrm{EM}/\mathrm{Lat.}$, and the average tail latency per question at the P50, P95, and P99 percentiles. The latency is measured on four NVIDIA H100 GPUs with 80 GB of memory. More implementation details are provided in the Appendix A.1.

Baselines and Benchmarks. We evaluate **FastRAG** on seven QA benchmarks, including three general QA datasets (NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023)) and four multi-hop QA datasets (HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023)). We compare against prompt-based search reasoning (Search-o1 (Li et al., 2025)), RL-based search methods (ZeroSearch (Sun et al., 2025), Search-R1 (Jin et al., 2025), and R1-Searcher (Song et al., 2025)), and RAG acceleration methods targeting retrieval-call reduction (Self-RAG (Asai et al., 2024), Adaptive-RAG (Jeong et al., 2024), and HiPRAG (Wu et al., 2026)), retrieval latency (PipeRAG (Jiang et al., 2025), SearchAgent-X, and RaLMSpec (Zhang et al., 2024, Yang et al., 2026)), context compression (RECON (Xu et al., 2025) and xRAG (Cheng et al., 2024)), and reasoning latency (EfficientRAG (Zhuang et al., 2024)). Additional benchmark details are provided in the Appendix A.2.

4.2 MAIN RESULTS

Accuracy. Table 1 shows the EM performance across seven general and multi-hop QA benchmarks. **FastRAG** achieves consistent accuracy improvements across both model scales. On Qwen2.5-3B-Instruct, **FastRAG** obtains an average EM of 35.4%, outperforming Search-R1 3B Base by 5.1 points (30.3% $\rightarrow$ 35.4%) and the strongest competing RAG acceleration baseline, RECON, by 2.2 points (33.2% $\rightarrow$ 35.4%). The gains are particularly pronounced on multi-hop benchmarks, reaching 36.6% on HotpotQA, 36.3% on 2Wiki, and 29.2% on Bamboogle. Scaling to Qwen2.5-7B further improves the average EM to 47.3%, exceeding Search-R1 7B Base by 4.2 points (43.1% $\rightarrow$ 47.3%) and the strongest competing baseline, RECON, by 3.2 points (44.1% $\rightarrow$ 47.3%). These improvements suggest that the proposed speculative retrieval mechanism preserves strong task performance across different reasoning complexities.

Table 1: Main results on general and multi-hop QA benchmarks with Qwen2.5-Instruct (3B/7B). EM scores (%) are reported. The best EM is highlighted in **bold**, while the second-best EM is underlined. Retrieval-call reduction methods are shown with red background, retrieval-latency reduction methods with green background, context compression methods with orange background, and reasoning-latency reduction method with blue background.

Methods	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average
3B Base Model								
Search-o1 3B	16.6	31.0	8.2	14.8	22.4	5.2	22.4	17.2
ZeroSearch 3B	41.4	57.4	44.8	27.4	30.0	9.8	11.1	31.7
Search-R1 3B Base	40.6	58.7	43.5	28.4	27.3	4.9	8.8	30.3
Search-R1 3B Instruct	34.1	54.5	37.8	32.4	31.9	10.3	26.4	32.5
Self-RAG	36.5	49.2	36.8	33.3	34.0	10.8	27.1	32.5
Adaptive-RAG	33.6	45.7	34.3	31.3	33.4	10.8	24.4	30.5
HiPRAG	38.5	49.3	38.0	33.4	33.4	12.0	24.2	32.7
PipeRAG	37.4	49.0	34.8	33.2	31.9	**13.3**	25.3	32.1
SearchAgent-X	36.8	47.3	35.6	32.5	32.2	12.2	27.4	32.0
RaLMSpec	29.5	39.4	29.9	27.4	27.2	10.6	22.2	26.6
RECON	37.0	49.1	37.5	33.5	35.3	11.5	28.4	33.2
xRAG	33.0	45.2	33.8	30.8	30.6	10.6	24.6	29.8
EfficientRAG	34.0	46.5	37.8	31.8	31.5	10.8	23.9	30.9
FastRAG	**39.3**	**53.6**	**40.1**	**36.6**	**36.3**	12.7	**29.2**	**35.4**
7B Base Model								
Search-o1 7B	19.4	40.6	11.4	17.0	27.0	8.6	30.4	22.1
ZeroSearch 7B	43.6	65.2	48.8	34.6	35.2	18.4	27.8	39.1
R1-Searcher 7B	40.4	52.2	41.0	44.2	51.3	15.8	36.8	40.2
Search-R1 7B Base	48.0	63.8	45.7	43.3	38.2	19.6	43.2	43.1
Search-R1 7B Instruct	39.3	61.0	39.7	37.0	41.4	14.6	36.8	38.5
Self-RAG	46.0	58.0	43.4	46.0	42.5	18.2	43.5	42.5
Adaptive-RAG	42.7	54.5	40.0	43.5	41.3	15.4	40.5	39.7
HiPRAG	47.0	63.0	44.8	48.2	44.7	18.0	42.3	44.0
PipeRAG	44.6	56.8	42.0	45.4	41.7	18.9	39.7	41.3
SearchAgent-X	41.5	53.1	39.3	42.0	40.2	15.0	40.4	38.8
RaLMSpec	39.7	51.0	37.1	40.0	38.5	13.4	39.5	37.0
RECON	48.7	60.5	44.5	50.4	43.0	15.0	**46.8**	44.1
xRAG	43.4	55.2	41.0	44.2	40.3	15.8	41.8	40.2
EfficientRAG	46.8	59.0	45.0	47.0	43.5	17.5	43.0	43.1
FastRAG	**51.8**	**63.5**	**49.6**	**51.0**	**48.7**	**20.4**	46.1	**47.3**

Latency. Table 2 compares the end-to-end latency and tail latency on NQ with Qwen2.5-7B-Instruct. **FastRAG** substantially reduces inference latency while simultaneously improving task accuracy. Compared with Search-R1, **FastRAG** reduces the average latency from 3.314 s to 1.247 s, corresponding to a 62.4% reduction, while improving EM from 48.0% to 51.8%. The latency reduction is also consistent across the entire latency distribution: P50, P95, and P99 decrease from 2.647/7.083/11.214 s to 0.973/2.847/4.263 s, respectively, yielding approximately 63.2%/59.8%/62.0% reductions at all three percentiles. Compared with the fastest competing baseline, SearchAgent-X, **FastRAG** further reduces latency by 21.1% (1.580 s $\rightarrow$ 1.247 s). It also achieves the highest efficiency of 41.54, approximately 1.47$\times$ that of EfficientRAG (28.35), demonstrating that **FastRAG** substantially reduces the reasoning latency that dominates end-to-end inference through intermediate-layer reasoning, while maintaining competitive task performance.

Table 2: Latency comparison of different RAG acceleration methods on the NQ benchmark.

Method	Lat.(s)	Eff.	P50(s)	P95(s)	P99(s)	EM(%)
Search-R1 7B	3.314	14.48	2.647	7.083	11.214	48.0
Self-RAG	2.807	16.39	2.243	5.971	9.386	46.0
Adaptive-RAG	2.561	16.67	2.057	5.487	8.431	42.7
HiPRAG	3.046	15.43	2.439	6.517	10.183	47.0
PipeRAG	2.019	22.09	1.473	4.592	7.284	44.6
SearchAgent-X	1.580	26.27	1.517	3.643	6.417	41.5
RaLMSpec	1.856	21.39	1.624	4.287	6.836	39.7
RECON	2.286	21.30	1.893	5.114	5.763	48.7
xRAG	2.024	21.44	1.714	4.681	7.392	43.4
EfficientRAG	1.651	28.35	1.553	3.941	5.938	46.8
FastRAG	**1.247**	**41.54**	**0.973**	**2.847**	**4.263**	**51.8**

4.3 Ablation Study

Table 3 evaluates the contribution of each component in **FastRAG**. Removing the *token calibrator*, $R^{\texttt{con}}$, and $R^{\texttt{exit}}$ reduces the average EM from 47.3% to 46.1%, 45.4%, and 46.3%, respectively,

Table 3: Ablation study of each key components in **FastRAG** with Qwen2.5-7B.

Model	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average
Search-R1 7B	48.0	63.8	45.7	43.3	38.2	19.6	43.2	43.1
FastRAG 7B	51.8	**63.5**	**49.6**	**51.0**	**48.7**	20.4	**46.1**	**47.3**
w/o token calibrator	**52.0**	62.3	48.4	49.8	47.2	19.5	43.5	46.1
w/o $R^{\texttt{con}}$	50.0	61.8	48.9	49.4	47.9	18.8	41.0	45.4
w/o $R^{\texttt{exit}}$	50.5	62.8	48.5	50.3	47.4	**20.6**	44.0	46.3

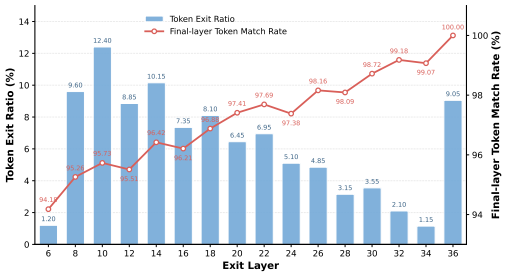


Figure 7: Exit-layer analysis. The bars show the token proportion exiting at intermediate layer. The line reports final-layer token match rate.

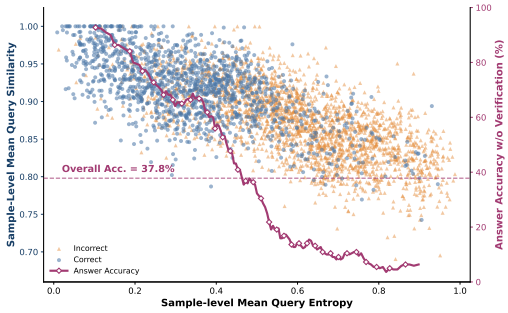


Figure 8: Relationship between sample-level mean query entropy, query similarity, and answer accuracy without full-layer fallback.

confirming that all components contribute to the final performance. The largest degradation occurs when removing $R^{\texttt{con}}$, highlighting the importance of maintaining consistency between speculative and reference queries. The *token calibrator* improves the reliability of intermediate-layer predictions, while $R^{\texttt{exit}}$ encourages effective early exit. Overall, these results demonstrate that intermediate-layer calibration, speculative-query consistency, and early-exit provide complementary benefits to **FastRAG**. More ablation studies are provided in the Appendix B, including sensitivity of the verification threshold γ (Appendix B.1), the query-similarity threshold τ (Appendix B.2), the *exit trigger* and *token calibrator* coefficients λ_{spe} and λ_{cal} (Appendix B.3), the reward coefficients λ_{task}, λ_{fmt}, λ_{con}, and λ_{exit} (Appendix B.4), the verification strategy (Appendix B.5).

4.4 Further Analyses and Case Studies

Exit Layer Analysis. Fig. 7 shows that **FastRAG** dynamically distributes token exits across intermediate layers on Qwen2.5-3B-Instruct. A substantial fraction of tokens exit at early-to-middle layers, with the highest ratio of 12.4% at layer 10. Meanwhile, the final-layer token match rate increases from 94.18% at Layer 6 to over 99% in deeper layers, demonstrating that **FastRAG** effectively reduces redundant layer computation while maintaining high token-level consistency.

Speculative Query Consistency Analysis. To verify the reliability of speculative queries, we analyze the relationship among speculative-query entropy, similarity to reference queries, and answer accuracy in Fig. 8. Lower-entropy queries exhibit higher similarity and are more likely to produce correct answers, while increasing entropy leads to declines in similarity and accuracy. These results indicate that speculative-query entropy can effectively indicate fallback necessity. More analyses, including efficiency experiments (Appendix C), training dynamics (Appendix D.1), intermediate-layer query token prediction analysis (Appendix D.2), and case studies are provided in the Appendix.

5 Conclusion

In this work, we find that intermediate-layer reasoning can produce retrieval queriefull-depth generation, enabling earlier retrieval with less computation. Based on this observation, we propose **FastRAG**, a self-speculative retrieval framework that uses an *exit trigger* and a *token calibrator* to dynamically generate speculative queries from intermediate layers. We further introduce SpecGRPO to jointly optimize query consistency, early exiting, and intermediate-layer prediction quality. Experiments across seven QA benchmarks show that **FastRAG** substantially reduces reasoning and retrieval waiting latency while preserving competitive answer accuracy.

AI USE STATEMENT

In preparing this work, we employed large language models (LLMs) to assist with polishing the writing, conducting literature reviews, improving the clarity of presentation, and refining the organization of the manuscript. All generated content was carefully reviewed and verified by the authors.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of **`FastRAG`** and SpecGRPO in §3, with experimental settings reported in §4.1 and Appendix A. All key configurations, including Qwen2.5-Instruct backbones, training data, optimization hyperparameters, rollout settings, reward settings, retrieval setup, hardware, benchmarks, and evaluation protocols, are reported. Code and configuration files will be released to support reproducibility for public research use.

ETHICS STATEMENT

In this work, we investigate speculative retrieval and intermediate-layer reasoning for improving the efficiency of retrieval-augmented generation. Our experiments are conducted using publicly available LLMs, training data, retrieval corpora, and QA benchmarks. We do not collect new human-subject data or private information. Our research focuses on reducing reasoning and retrieval latency in RAG systems and does not involve safety-critical applications.

REFERENCES

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

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

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. *Advances in Neural Information Processing Systems*, 37:109487–109516, 2024.

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

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

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

Taeho Hwang, Sukmin Cho, Soyeong Jeong, Hoyun Song, SeungYoon Han, and Jong C Park. Exit: Context-aware extractive compression for enhancing retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4895–4924, 2025.

Gautier Izacard and Edouard Grave. 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*, pp. 874–880, 2021.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 7036–7050, 2024.

Wenqi Jiang, Shuai Zhang, Boran Han, Jie Wang, Bernie Wang, and Tim Kraska. Piperag: Fast retrieval-augmented generation via adaptive pipeline parallelism. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 589–600, 2025.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

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

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9802–9822, 2023.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1–15, 2021.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V Vasilakos. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013, 2024.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 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)*, pp. 10014–10037, 2023.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. Hiprag: hierarchical process rewards for efficient agentic retrieval augmented generation. In *International Conference on Learning Representations*, volume 2026, pp. 82317–82347, 2026.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

Zhichao Xu, Minheng Wang, Yawei Wang, Wenqian Ye, Yuntao Du, Yunpu Ma, and Yijun Tian. Recon: reasoning with condensation for efficient retrieval-augmented generation. *arXiv preprint arXiv:2510.10448*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Tiannuo Yang, Zebin Yao, Bowen Jin, Lixiao Cui, Yusen Li, Gang Wang, Xiaoguang Liu, and Willie Neiswanger. Demystifying and enhancing the efficiency of large language model based search agents. In *International Conference on Learning Representations*, volume 2026, pp. 41432–41453, 2026.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Liu Weichuan, Lei Hou, and Juanzi Li. Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27022–27043, 2025.

Qianhao Yuan, Jie Lou, Zichao Li, Jiawei Chen, Yaojie Lu, Hongyu Lin, Le Sun, Debing Zhang, and Xianpei Han. Memsearcher: Iterative memory integration for search agent via end-to-end reinforcement learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14965–14977, 2026.

Wuyang Zhang and Shichao Pei. Predictive prefetching for retrieval-augmented generation. *arXiv preprint arXiv:2605.17989*, 2026.

Zhihao Zhang, Alan Zhu, Lijie Yang, Yihua Xu, Lanting Li, Phitchaya Mangpo Phothilimthana, and Zhihao Jia. Accelerating iterative retrieval-augmented language model serving with speculation. In *Forty-first International Conference on Machine Learning*, 2024.

Yijia Zheng and Marcel Worring. Latentrag: Latent reasoning and retrieval for efficient agentic rag. *arXiv preprint arXiv:2605.06285*, 2026.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Efficientrag: Efficient retriever for multi-hop question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3392–3411, 2024.

APPENDIX

A Additional Experimental Details

A.1 Training Configuration

We optimize **FastRAG** with a learning rate of 1×10^{-6} and a global batch size of 256. Training is conducted for 1 epoch. The maximum prompt length and response length are set to 695 and 6,024 tokens, respectively. We adopt full-parameter optimization with gradient checkpointing and train the models on 8 NVIDIA H100 GPUs. We set the reward coefficients to $\lambda_{\text{task}} = 0.4$, $\lambda_{\text{fmt}} = 0.2$, $\lambda_{\text{con}} = 0.3$, and $\lambda_{\text{exit}} = 0.1$. The *exit trigger* and *token calibrator* coefficients are set to $\lambda_{\text{spe}} = 0.3$ and $\lambda_{\text{cal}} = 0.2$, respectively. The reference-policy KL coefficient is $\beta = 0.01$, and the clipping coefficient is $\epsilon_c = 0.2$. We use a query-similarity threshold of $\tau = 0.78$ and an entropy threshold of $\gamma = 0.55$ for selective verification.

A.2 Benchmarks

We provide additional details of the evaluation benchmarks used in the main experiments.

Natural Questions (NQ) (Kwiatkowski et al., 2019) is a large-scale open-domain QA benchmark built from real user queries submitted to Google Search, with answers grounded in Wikipedia. It provides a realistic setting for evaluating knowledge retrieval and factual question answering.

TriviaQA (Joshi et al., 2017) contains trivia questions paired with evidence collected from Wikipedia and the Web. Compared with NQ, its questions are typically more knowledge-oriented and require retrieving specific factual evidence from a large corpus.

PopQA (Mallen et al., 2023) is an entity-centric open-domain QA benchmark constructed from factual relations in a knowledge base. It includes entities with substantially different popularity levels. It examines whether external retrieval can compensate for knowledge that is weakly represented in the model's parametric memory.

HotpotQA (Yang et al., 2018) is a widely used multi-hop QA benchmark where answering a question often requires combining evidence distributed across multiple Wikipedia articles. It includes both bridge and comparison questions and therefore evaluates the ability to perform iterative retrieval and evidence integration.

2Wiki (Ho et al., 2020) focuses on multi-hop reasoning over Wikipedia documents. The questions are associated with explicit reasoning paths connecting multiple supporting facts, enabling systematic evaluation of retrieval across intermediate reasoning steps.

MuSiQue (Trivedi et al., 2022) is designed to provide a challenging evaluation of multi-hop reasoning by composing independently answerable single-hop questions while reducing exploitable shortcuts. Solving these questions generally requires the model to identify intermediate entities, retrieve the corresponding evidence, and progressively combine information across multiple reasoning steps. This makes MuSiQue particularly suitable for evaluating iterative retrieval systems.

Bamboogle (Press et al., 2023) consists of manually constructed questions that cannot typically be answered through a single straightforward search. Instead, the model must use information obtained from one retrieval step to formulate subsequent searches and eventually connect multiple pieces of evidence. Although relatively small, Bamboogle provides a challenging setting for evaluating multi-step search and retrieval-augmented reasoning.

A.3 Query Similarity Calculation

We implement $\mathrm{Sim}(\cdot,\cdot)$ in Eq. (4) as the cosine similarity between the dense representations of the speculative and reference queries. Before encoding, we remove leading and trailing whitespace and collapse consecutive whitespace characters, while preserving the original letter case. Both $q_{i,m}^{\texttt{spe}}$ and $q_{i,m}^{\texttt{ref}}$ are then encoded independently using the same E5 query encoder employed by the retriever. We prepend the E5 query instruction `query:` to both queries, tokenize them with a maximum length of 128, and apply attention-mask-aware mean pooling over the encoder outputs. Let $\mathbf{e}(q)$ denote the resulting pooled representation of query q. We obtain its ℓ_2-normalized representation as

$$\widehat{\mathbf{e}}(q) = \frac{\mathbf{e}(q)}{\|\mathbf{e}(q)\|_2}. \tag{14}$$

The query similarity used for verification is

$$\mathrm{Sim}\left(q_{i,m}^{\texttt{spe}}, q_{i,m}^{\texttt{ref}}\right) = \widehat{\mathbf{e}}\left(q_{i,m}^{\texttt{spe}}\right)^{\top} \widehat{\mathbf{e}}\left(q_{i,m}^{\texttt{ref}}\right). \tag{15}$$

This score is equivalent to cosine similarity and lies in $[-1,1]$. Because the two queries may be generated from different reasoning prefixes, verification is performed on their complete query-level representations and does not require token-wise alignment. For a Boolean condition $\mathcal{C}$, the indicator function is defined as

$$\mathbb{I}[\mathcal{C}] = \begin{cases} 1, & \text{if } \mathcal{C} \text{ is true,} \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Accordingly, the verification variable in Eq. (4) is computed as

$$V_{i,m} = \begin{cases} 1, & \mathrm{Sim}\left(q_{i,m}^{\texttt{spe}}, q_{i,m}^{\texttt{ref}}\right) \geq \tau, \\ 0, & \mathrm{Sim}\left(q_{i,m}^{\texttt{spe}}, q_{i,m}^{\texttt{ref}}\right) < \tau. \end{cases} \tag{17}$$

We use $\tau = 0.78$ in all experiments. When $V_{i,m} = 1$, the retrieval results initiated by $q_{i,m}^{\texttt{spe}}$ are retained. When $V_{i,m} = 0$, these speculative results are discarded and retrieval is performed using $q_{i,m}^{\texttt{ref}}$. An empty or malformed query, or a failure during similarity computation, is treated as an unsuccessful verification and follows the same reference-query fallback. During training, this procedure is applied to every speculative-reference query pair to compute $V_{i,m}$. During inference, query-similarity verification is not required, where high-entropy speculative queries instead fall back directly to full-layer reference-query generation.

B Additional Ablation Studies

B.1 Sensitivity to the Verification Threshold γ

Table 4 studies the effect of the verification threshold γ. The performance remains relatively stable across different thresholds, with the average EM ranging from 46.6% to 47.3%. The best average performance of 47.3% is achieved at $\gamma = 0.55$. A smaller γ causes more speculative queries to fall back to full-layer query generation, but does not consistently improve accuracy. In contrast, a larger γ directly accepts more speculative queries, which may admit less reliable high-entropy queries and slightly degrade overall performance. Overall, $\gamma = 0.55$ yields the strongest average performance across the seven benchmarks and is selected as the default setting.

Table 4: Sensitivity analysis of the verification threshold γ on Qwen2.5-7B-Instruct.

γ	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average
0.45	**52.1**	63.7	49.2	50.4	48.6	20.0	45.7	47.1
0.50	51.9	**63.8**	49.0	50.3	48.0	20.3	45.0	46.9
0.55	51.8	63.5	49.6	51.0	**48.7**	**20.4**	**46.1**	**47.3**
0.60	51.0	62.5	**49.9**	51.2	47.8	19.8	44.0	46.6
0.65	50.7	61.8	49.7	**51.3**	47.9	19.7	45.8	46.7

Table 5: Sensitivity analysis of the query-similarity threshold τ on Qwen2.5-7B-Instruct.

τ	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average
0.74	51.4	**63.8**	49.2	50.7	48.2	20.3	44.0	46.8
0.76	51.4	63.4	49.2	**51.3**	48.3	20.2	**47.3**	**47.3**
0.78	**51.8**	63.5	**49.6**	51.0	**48.7**	**20.4**	46.1	**47.3**
0.80	51.5	63.0	49.5	50.5	48.3	20.2	46.7	47.1
0.82	51.7	63.1	49.3	50.6	48.6	20.0	45.0	46.9

B.2 Sensitivity to the Query-Similarity Threshold τ

The query-similarity threshold τ controls the strictness of matching speculative queries with their reference queries during verification. As shown in Table 5, the performance is relatively stable across the tested range, with the average EM varying between 46.8% and 47.3%. Moderate values perform best, where both $\tau = 0.76$ and $\tau = 0.78$ achieve an average EM of 47.3%. A smaller threshold may accept loosely matched queries, while a larger threshold can reject semantically consistent queries and introduce unnecessary fallback retrieval. We adopt $\tau = 0.78$ for our experiments.

B.3 Sensitivity to the Exit Trigger Coefficient λ_{spe} and the token calibrator Coefficient λ_{cal}.

The coefficients λ_{spe} and λ_{cal} control the contributions of the *exit trigger* and *token calibrator* during optimization, respectively. As shown in Table 6, varying λ_{spe} from 0.1 to 0.5 with $\lambda_{\text{cal}} = 0.2$ yields average EM scores between 46.8% and 47.3%, with the best performance achieved at $\lambda_{\text{spe}} = 0.3$. Similarly, fixing $\lambda_{\text{spe}} = 0.3$, $\lambda_{\text{cal}} = 0.2$ achieves the highest average EM of 47.3%, while larger values generally lead to performance degradation. These results indicate that moderate weights provide a suitable balance between early-exit optimization and intermediate-layer token calibration. We therefore adopt $\lambda_{\text{spe}} = 0.3$ and $\lambda_{\text{cal}} = 0.2$ in our experiments.

B.4 Sensitivity to Reward Coefficients

We further evaluate the sensitivity of **FastRAG** to the reward coefficients λ_{task}, λ_{fmt}, λ_{con}, and λ_{exit}, which balance task correctness, output formatting, speculative-query consistency, and early-exit behavior during SpecGRPO optimization. As shown in Table 7, performance varies slightly across coefficient changes, while overly emphasizing one objective generally degrades the average EM. For λ_{task} and λ_{con}, both $(0.3, 0.4)$ and $(0.4, 0.3)$ achieve the highest average EM of 47.3%, whereas increasing λ_{task} further reduces performance. A similar trend is observed for λ_{fmt} and λ_{con}, where $(0.2, 0.3)$ performs best. For λ_{con} and λ_{exit}, the combination $(0.3, 0.1)$ achieves 47.3%, while either weaker or stronger early-exit weighting leads to lower performance. Jointly varying λ_{task} and λ_{exit} also favors $(0.4, 0.1)$. Overall, these results indicate that a balanced weighting among task accuracy, query consistency, formatting, and early-exit efficiency is important for stable optimization. We therefore use $\lambda_{\text{task}} = 0.4$, $\lambda_{\text{fmt}} = 0.2$, $\lambda_{\text{con}} = 0.3$, and $\lambda_{\text{exit}} = 0.1$ throughout training.

B.5 ABLATION Study of Verification method

The verification strategy directly affects the trade-off between answer accuracy and inference latency. As shown in Table 8, removing verification achieves the lowest latency of 1.214 s but reduces the average EM to 45.1%, indicating that directly accepting all speculative queries can propagate unreliable retrieval results. Standard verification improves the average EM by 2.5 points (45.1%→47.6%), but increases latency by 1.323 s (1.214 s→2.537 s) because each speculative query requires full-layer

Table 6: Sensitivity analysis of the *exit trigger* coefficient λ_{spe} and the token calibrator coefficient λ_{cal} on Qwen2.5-7B-Instruct.

(a) Sensitivity to λ_{spe} with $\lambda_{\text{cal}} = 0.2$.

$\lambda_{\text{spe}} : \lambda_{\text{cal}}$	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average
0.1 : 0.2	51.3	**63.8**	49.1	50.4	48.6	20.6	43.8	46.8
0.2 : 0.2	51.5	63.3	49.2	**51.3**	48.2	20.2	46.0	47.1
0.3 : 0.2	**51.8**	63.5	**49.6**	51.0	**48.7**	20.4	46.1	**47.3**
0.4 : 0.2	51.7	61.7	49.3	50.6	47.3	**20.7**	46.3	46.8
0.5 : 0.2	51.4	62.7	49.5	50.5	48.2	19.4	**46.6**	46.9

(b) Sensitivity to λ_{cal} with $\lambda_{\text{spe}} = 0.3$.

$\lambda_{\text{spe}} : \lambda_{\text{cal}}$	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average
0.3 : 0.1	**52.1**	63.7	49.4	50.8	48.5	20.3	45.6	47.2
0.3 : 0.2	51.8	63.5	49.6	51.0	48.7	**20.4**	**46.1**	**47.3**
0.3 : 0.3	51.4	**63.8**	49.3	51.2	**48.9**	19.8	42.5	46.7
0.3 : 0.4	51.0	62.6	**49.9**	50.3	47.9	19.9	46.0	46.8
0.3 : 0.5	50.8	62.4	49.8	**51.4**	47.8	19.7	42.9	46.4

Table 7: Sensitivity analysis of the reward coefficients on Qwen2.5-7B-Instruct.

(a) Sensitivity between λ_{task} and λ_{con}.

$\lambda_{\text{task}} : \lambda_{\text{fmt}} : \lambda_{\text{con}} : \lambda_{\text{exit}}$	Average
0.20 : 0.20 : 0.50 : 0.10	47.0
0.30 : 0.20 : 0.40 : 0.10	**47.3**
0.40 : 0.20 : 0.30 : 0.10	**47.3**
0.50 : 0.20 : 0.20 : 0.10	46.7
0.60 : 0.20 : 0.10 : 0.10	46.9

(b) Sensitivity between λ_{fmt} and λ_{con}.

$\lambda_{\text{task}} : \lambda_{\text{fmt}} : \lambda_{\text{con}} : \lambda_{\text{exit}}$	Average
0.40 : 0.10 : 0.40 : 0.10	47.1
0.40 : 0.15 : 0.35 : 0.10	47.1
0.40 : 0.20 : 0.30 : 0.10	**47.3**
0.40 : 0.25 : 0.25 : 0.10	46.8
0.40 : 0.30 : 0.20 : 0.10	46.5

(c) Sensitivity between λ_{con} and λ_{exit}.

$\lambda_{\text{task}} : \lambda_{\text{fmt}} : \lambda_{\text{con}} : \lambda_{\text{exit}}$	Average
0.40 : 0.20 : 0.35 : 0.05	46.4
0.40 : 0.20 : 0.325 : 0.075	46.8
0.40 : 0.20 : 0.30 : 0.10	**47.3**
0.40 : 0.20 : 0.275 : 0.125	47.0
0.40 : 0.20 : 0.25 : 0.15	46.8

(d) Sensitivity between λ_{task} and λ_{exit}.

$\lambda_{\text{task}} : \lambda_{\text{fmt}} : \lambda_{\text{con}} : \lambda_{\text{exit}}$	Average
0.45 : 0.20 : 0.30 : 0.05	46.9
0.425 : 0.20 : 0.30 : 0.075	46.9
0.40 : 0.20 : 0.30 : 0.10	**47.3**
0.375 : 0.20 : 0.30 : 0.125	47.0
0.35 : 0.20 : 0.30 : 0.15	46.5

verification. In contrast, entropy-guided verification improves the average EM by 2.2 points over no verification (45.1%→47.3%) with only 0.168 s additional latency (1.214 s→1.382 s). Compared with standard verification, it maintains comparable accuracy (47.6%→47.3%, −0.3 points) while reducing latency by 1.155 s (2.537 s→1.382 s), corresponding to a 45.5% reduction. Moreover, standard verification achieves the best performance on four of the seven benchmarks, while entropy-guided verification leads on the remaining three, demonstrating that query entropy provides an effective criterion for selectively avoiding unnecessary full-layer verification.

C Additional Efficiency Experiments

To further evaluate the efficiency of **FastRAG** across different benchmarks, Table 9 reports the average latency and performance over NQ, TriviaQA, PopQA, HotpotQA, 2Wiki, MuSiQue, and Bamboogle based on Qwen2.5-7B-Instruct. Compared with Search-R1, **FastRAG** reduces the average end-to-end latency from 3.673 s to 1.382 s (−62.4%) while improving the average EM from 43.1% to 47.3% (+4.2 points), and increases the efficiency score from 11.73 to 34.23 (2.92×). The latency improvement is also consistent across the distribution, with P50/P95/P99 decreasing from 2.947/8.283/13.117 s to 1.013/2.993/4.587 s. Compared with EfficientRAG, the fastest competing baseline, **FastRAG** further reduces latency from 1.830 s to 1.382 s (−24.5%) and improves efficiency from 23.55 to 34.23 (+45.4%). It also achieves lower P50, P95, and P99 latency than SearchAgent-X, RaLMSpec, and xRAG, while its average EM of 47.3% exceeds the strongest competing baseline, RECON, by 3.2 points (44.1%→47.3%). These results demonstrate that the efficiency gains of

Table 8: Ablation study of different verification strategies on seven QA benchmarks.

Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average	Lat.(s)
FastRAG w/o Verification	51.9	60.2	49.7	47.8	45.3	18.0	42.8	45.1	**1.214**
FastRAG w/ Standard Verification	**52.3**	63.0	**50.1**	50.7	**49.2**	20.0	**47.9**	**47.6**	2.537
FastRAG w/ Entropy-Guided Verification	51.8	**63.5**	49.6	**51.0**	48.7	**20.4**	46.1	47.3	1.382

Table 9: Average performance and latency comparison of different RAG acceleration methods across NQ, TriviaQA, PopQA, HotpotQA, 2Wiki, MuSiQue, and Bamboogle.

Method	Lat.(s)	Eff.	P50(s)	P95(s)	P99(s)	EM(%)
Search-R1 7B	3.673	11.73	2.947	8.283	13.117	43.1
Self-RAG	3.111	13.66	2.493	7.087	11.423	42.5
Adaptive-RAG	2.838	13.99	2.276	6.487	10.317	39.7
HiPRAG	3.376	13.03	2.693	7.617	12.031	44.0
PipeRAG	2.238	18.45	1.617	5.083	8.027	41.3
SearchAgent-X	1.911	20.30	1.427	4.143	6.687	38.8
RaLMSpec	2.057	17.99	1.547	4.337	7.023	37.0
RECON	2.534	17.40	1.843	5.687	8.917	44.1
xRAG	2.243	17.92	1.677	4.983	7.893	40.2
EfficientRAG	1.830	23.55	1.487	3.997	6.193	43.1
FastRAG	**1.382**	**34.23**	**1.013**	**2.993**	**4.587**	**47.3**

FastRAG generalize across benchmarks and latency percentiles while maintaining strong task performance.

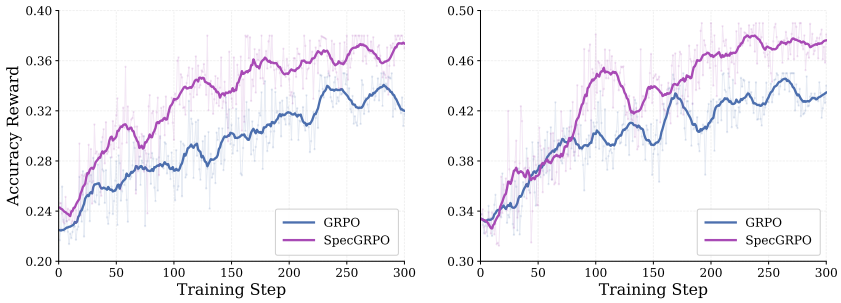
D Additional Analyses

D.1 Training Dynamics

We analyze the training dynamics of GRPO and SpecGRPO on Qwen2.5-3B/7B-Instruct in Figs. 9 and 10. As shown in Fig. 9, the accuracy reward shows an overall increasing trend during training for both model scales, while SpecGRPO converges to a higher reward than standard GRPO, indicating that the additional objectives do not compromise task optimization and instead improve the final policy. Fig. 10 further shows that both the speculative-query consistency reward and early-exit reward progressively increase throughout training. The consistency reward reaches a higher level as training proceeds, indicating increasingly reliable speculative queries, while the improvement in the early-exit reward reflects the model's growing ability to reduce unnecessary layer computation. Similar trends are observed for both 3B and 7B models, demonstrating that SpecGRPO jointly learns accurate retrieval-query speculation and effective early-exit behavior while maintaining stable task-reward optimization.

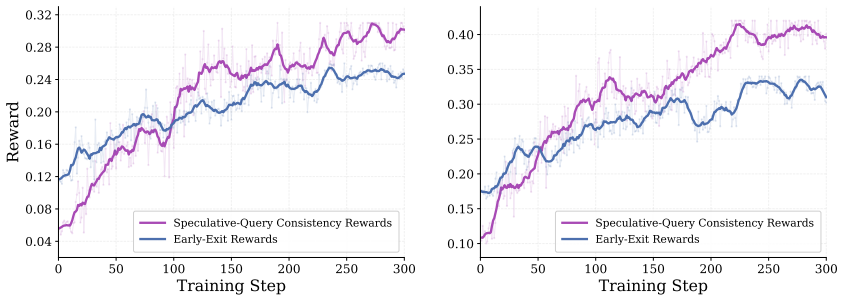
D.2 Intermediate-Layer Query Token Prediction Analysis

To further understand how retrieval-query tokens evolve across LLM layers in our trained **FastRAG**, Figs. 11 and 12 visualize the layer-wise token predictions and their entropy for two cases. In both examples, predictions at shallow layers are unstable and exhibit low consistency with the final-layer tokens, while deeper layers progressively converge toward the final query with lower token entropy. In Fig. 11, the token match rate increases from 8% at Layer 6 to 75% at Layers 15–16, and several deeper intermediate layers already reach over 90% agreement, with Layers 33 and 35 achieving complete agreement with the final-layer prediction. A similar pattern appears in Fig. 12, where the match rate rises from 12% at Layer 8 to 50% at Layers 13–16 and remains around 62–75% across many later layers before reaching 88% at Layer 35. Meanwhile, the entropy visualization becomes generally lower as predictions stabilize, indicating increasing confidence in intermediate-layer query tokens. These cases show that many retrieval-query tokens become reliable well before the final layer, while occasional intermediate-layer mismatches persist, motivating dynamic early exiting with entropy-guided fallback in **FastRAG**.



(a) Qwen2.5-3B-Instruct (b) Qwen2.5-7B-Instruct

Figure 9: Training dynamics of accuracy reward for GRPO and SpecGRPO on Qwen2.5-3B and 7B.



(a) Qwen2.5-3B-Instruct (b) Qwen2.5-7B-Instruct

Figure 10: Training dynamics of the Speculative-Query Consistency Reward and Early-Exit Reward on Qwen2.5-3B and 7B.

E RELATED WORK

Retrieval-Augmented Generation. Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Guu et al., 2020; Borgeaud et al., 2022; Singh et al., 2025) provides LLMs with external knowledge to extend its generation capabilities. Early works train retrievers together with LLMs and study how to generate answers from multiple passages (Lewis et al., 2020; Guu et al., 2020; Izacard & Grave, 2021; Izacard et al., 2023). Later methods introduce retrieval during generation. For example, IRCoT (Trivedi et al., 2023) uses intermediate reasoning results to guide subsequent searches, while FLARE (Jiang et al., 2023) and DRAGIN (Su et al., 2024) use token uncertainty to determine when additional retrieval is needed. Self-RAG (Asai et al., 2024) learns to retrieve and assess retrieval results through reflection tokens. Adaptive-RAG (Jeong et al., 2024) chooses among no retrieval, single-step retrieval, and iterative retrieval according to question complexity. Recent approaches increasingly optimize query generation and search strategies end-to-end from task feedback. For example, Search-R1 (Jin et al., 2025) trains LLMs to interleave reasoning with multi-round search through reinforcement learning, while MemSearcher (Yuan et al., 2026) extends this paradigm with compact memory management across retrieval rounds. However, these methods typically require waiting for retrieval results before subsequent reasoning, resulting in retrieval-induced waiting latency (Zhang et al., 2024; Jiang et al., 2025).

RAG Acceleration. Recent studies have investigated several directions for improving the efficiency of RAG. Among them, retrieval-call control methods (Asai et al., 2024; Jeong et al., 2024; Wu et al., 2026; Su et al., 2024; Yao et al., 2025) avoid unnecessary retrieval by determining whether external knowledge is needed based on the current reasoning state. For example, Adaptive-RAG (Jeong et al., 2024) predicts the complexity of the input query with a lightweight classifier and dynamically selects among no retrieval, single-step retrieval, and iterative retrieval. SEAKR (Yao et al., 2025) estimates self-aware uncertainty from the LLM's internal states and triggers retrieval only when the model exhibits high uncertainty. Context compression methods (Xu et al., 2024; Xu et al., 2025; Cheng et al., 2024; Hwang et al., 2025) reduce LLM input overhead by filtering redundant tokens, thereby lowering prefill and decoding costs. For example, RECON (Xu et al., 2025) compresses retrieved information into concise representations before adding it to the multi-turn context. xRAG (Cheng et al., 2024) maps the dense embedding of each retrieved document into a single soft token,

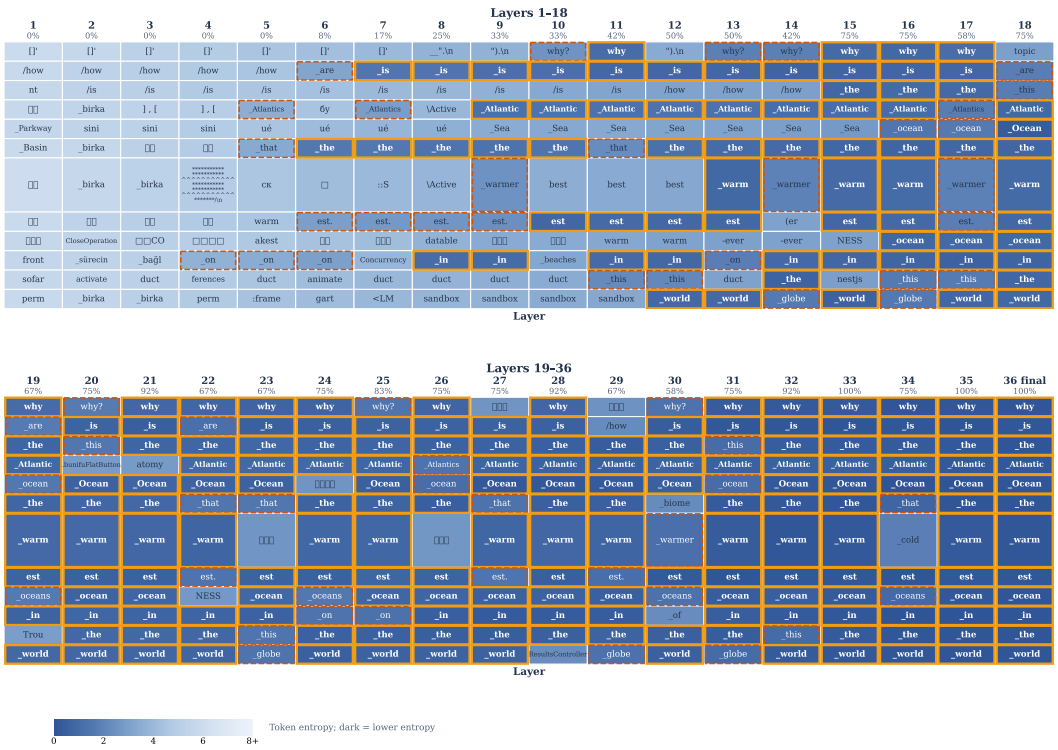


Figure 11: Case 1: Layer-wise token prediction dynamics and entropy across LLM layers.



Figure 12: Case 2: Layer-wise token prediction dynamics and entropy across LLM layers.

substantially reducing context length. EXIT (Hwang et al., 2025) performs context-aware sentence selection over retrieved documents and retains only sentences relevant to the query. Retrieval-wait reduction methods (Jiang et al., 2025; Yang et al., 2026; Zhang et al., 2024; Zhang & Pei, 2026) reduce retrieval-induced waiting latency by initiating retrieval in advance and running it in parallel with ongoing LLM generation. For instance, Predictive Prefetching (Zhang & Pei, 2026) anticipates future retrieval needs and generates retrieval queries, enabling retrieval to proceed asynchronously during generation. Reasoning acceleration methods (Zhuang et al., 2024; Zheng & Worring, 2026) reduce inference latency by avoiding repeated LLM calls, replacing explicit autoregressive reasoning with latent computation, or reusing intermediate representations. For example, EfficientRAG (Zhuang et al., 2024) uses lightweight modules to generate queries across multiple retrieval rounds, reserving

the LLM only for final answer generation. LatentRAG (Zheng & Worring, 2026) replaces explicitly generated natural-language reasoning and retrieval queries with latent tokens produced in a single forward pass, which are then directly mapped to retrieval representations. Overall, these methods improve RAG efficiency from different perspectives, but mainly target either LLM computation or retrieval overhead, with limited attention to the end-to-end latency caused by both reasoning and retrieval. In contrast, **FastRAG** ingeniously leverages the potential of intermediate-layer tokens to avoid redundant layer computation, which enables earlier retrieval-query generation and retrieval execution, ultimately reducing end-to-end latency.

F LIMITATIONS

Although effective, **FastRAG** has several limitations. First, our evaluation focuses on Qwen2.5-3B/7B-Instruct with a fixed retriever and corpus, and its generalization to larger models and different retrievers remains to be studied. Second, **FastRAG** relies on intermediate-layer query predictability and uses token entropy to estimate the reliability of speculative queries. For difficult queries, intermediate-layer predictions may diverge from final-layer queries, potentially increasing verification costs. Finally, SpecGRPO requires full-layer reference queries during training for query-consistency supervision and token calibration, introducing additional training overhead.

Case 2

user prompt

```
<question>
swan lake the sleeping beauty and the nutcracker are three famous ballets by
</question>
```

assistant output

```
<think>
The question is asking about the composer of the ballets Swan Lake,
The Sleeping Beauty, and The Nutcracker.
</think>
<tool_call>
{"name": "wikipedia_search",
 "arguments": {"query": "Swan Lake The Sleeping Beauty The Nutcracker
composer",
              "top_n": 1}}
</tool_call>
```

user prompt

```
<tool_response>
"Swan Lake"
... Tchaikovsky studied the music of "specialists" ...
</tool_response>
```

assistant output

```
<think>
The result does not directly provide the composer's name.
I will perform another search to find the specific composer.
</think>
<tool_call>
{"name": "wikipedia_search",
 "arguments": {"query": "composer of Swan Lake The Sleeping Beauty The
Nutcracker",
              "top_n": 1}}
</tool_call>
```

user prompt

```
<tool_response>
"Swan Lake"
... Tchaikovsky studied the music of "specialists" ...
</tool_response>
```

assistant output

```
<think>
Based on the context, the composer is Pyotr Ilyich Tchaikovsky.
</think>

\boxed{Pyotr Ilyich Tchaikovsky}
```

System Prompt

In this environment you have access to a set of tools you can use to assist with the user query. You may perform multiple rounds of function calls. In each round, you can call one or more functions.

Here are available functions in JSONSchema format:

```
{
    "type": "function",
    "function": {
        "name": "wikipedia_search",
        "description": "Search Wikipedia for a given query.",
        "parameters": {
            "type": "object",
            "properties": {
                "query": {
                    "type": "string",
                    "description": "Query to search for."
                },
                "top_n": {
                    "type": "integer",
                    "description": "Number of results to return. The default value is 5.",
                    "default": 5
                }
            },
            "required": ["query"]
        }
    }
}
```

In your response, you need to first think about the reasoning process in the mind and then conduct function calling to get the information or perform the actions if needed. The reasoning process and function calling are enclosed within <think> </think> and <tool_call> </tool_call> tags. The results of the function calls will be given back to you after execution, and you can continue to call functions until you get the final answer for the user's question.

Finally, if you have got the answer, enclose it within \boxed{} with latex format and do not continue to call functions.

For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:

```
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

Case 1

user prompt

<question> who is the current director of the us mint </question>

assistant output

<think>
To answer this question, I need to search for the current director of the US Mint. I will use the wikipedia_search function to find this information.
</think>
<tool_call>
{"name": "wikipedia_search",
 "arguments": {"query": "current director of the US Mint"}}
</tool_call>

user prompt

<tool_response>
{"name": "wikipedia_search",
 "arguments": {"query": "current director of the US Mint"}}

"Director of the United States Mint"
The Director of the United States Mint is a presidential appointment needing Senate confirmation. David J. Ryder became director in April 2018. He previously served as director from 1992 to 1993. ...

"David J. Ryder"
... David J. Ryder is the current Director of the United States Mint. ...
</tool_response>

assistant output

<think>
The search results indicate that the current director of the US Mint is David J. Ryder. He was confirmed by a voice vote on March 21, 2018, and has been serving in this position since then.
</think>
<tool_call>
{"name": "wikipedia_search",
 "arguments": {"query": "David J. Ryder US Mint"}}
</tool_call>

Case 1

user prompt

```
<tool_response>
{"name": "wikipedia_search",
 "arguments": {"query": "David J. Ryder US Mint"}}

"David J. Ryder"
... Ryder was again nominated as Director of the United States Mint by
President Donald Trump and confirmed by a voice vote on March 21, 2018.
David J. Ryder ... is the current Director of the United States Mint. ...

"Director of the United States Mint"
... David J. Ryder became director in April 2018. ...
</tool_response>
```

assistant output

```
<think>
The search results clearly state that David J. Ryder is the current director
of the US Mint. He was confirmed in March 2018 and has been serving in this
position since then.
</think>

<tool_call>
{"name": "wikipedia_search",
 "arguments": {"query": "current director of the US Mint"}}
</tool_call>
```

user prompt

```
<tool_response>
{"name": "wikipedia_search",
 "arguments": {"query": "current director of the US Mint"}}

"Director of the United States Mint"
The Director of the United States Mint is a presidential appointment needing
Senate confirmation. David J. Ryder became director in April 2018. ...

"David J. Ryder"
... David J. Ryder is the current Director of the United States Mint. ...
</tool_response>
```

assistant output

```
<think>
The search results confirm that David J. Ryder is the current director of
the US Mint. He was confirmed in March 2018 and has been serving in this
position since then.
</think>

The current director of the US Mint is \boxed{David J. Ryder}.
```