MONOROUTE: ONE SYMMETRIC LOW-RANK SEARCH SCALES ATTENTION MEMORY TO 1B TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce **MonoRoute** to combine the answer quality of attention-based retrieval with high question-answering throughput. Memory Sparse Attention (MSA) integrates document selection and multi-hop reasoning within a transformer, but repeatedly searches the corpus in 18 layers using 1024-dimensional routing keys. MonoRoute replaces these searches with a single routing layer per hop; later layers reuse its selection while attending to their own document key-value states. Projection symmetrization constructs a shared query and key **orthogonal projection**, enabling rank reduction to 128 dimensions without retraining and cutting routing-key storage eightfold. Together, these changes reduce leading-order similarity-scoring memory and compute by $144\times$, allowing us to construct a corpus index at only 4 bytes per token, while preserving iterative retrieval. Across five QA datasets, MonoRoute exceeds FlashRAG on all four reported answer-quality metrics and achieves 3.1-6.5$\times$ its throughput on HotpotQA and 2WikiMultiHopQA at batch size 16, including $3.1\times$ at one billion tokens. These results support attention-based retrieval as an accurate, high-throughput alternative to retrieval-augmented generation (RAG).

1 INTRODUCTION

Modern large language models increasingly succeed by conditioning on more *context*. A question may fit in a single sentence, yet answering it can require connecting evidence scattered across an entire document collection. One fact may reveal which fact to seek next, making retrieval part of the reasoning process itself. Retrieval-augmented generation (RAG) offers an established route: it retrieves passages from an external index and supplies them as text to a language model (Lewis et al., 2020). Memory Sparse Attention (MSA) brings memory selection into the transformer itself. It encodes documents with the same model that answers the question, storing layer-specific key–value (KV) states that attention consumes directly. Figure 1 illustrates how this approach connects RAG's external memory access with sparse attention's selective use of KV states.

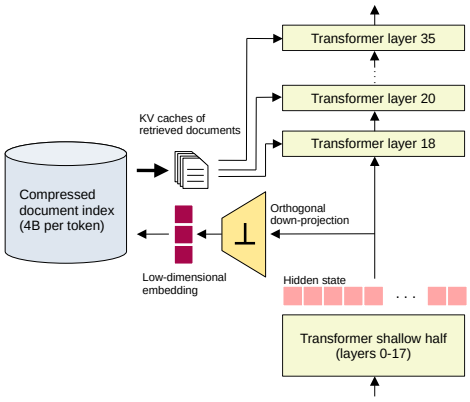

Figure 1: **Attention-based retrieval with MonoRoute** - selects documents based on a single attention layer; later layers reuse the selection. Routing queries and keys share a symmetrized low-rank orthogonal projection. The documents are indexed at only 4 bytes per token.

This integration of retrieval and reasoning is particularly useful for multi-hop questions, where the evidence needed at a later step depends on what has already been found. MSA's *Memory Interleave* appends retrieved documents to the context and uses that expanded context to guide the next retrieval step. In its original evaluation, MSA achieves higher LLM-judge answer-quality scores than the evaluated same-backbone RAG baselines on eight of nine QA datasets, including 2WikiMultiHopQA, HotpotQA, and MuSiQue (Chen et al., 2026b). These results motivate attention-based retrieval as a competitive alternative to RAG for reasoning over external knowledge. Our aim is to pair this answer quality with the throughput needed to serve many questions over the same memory at scale.

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.

MSA establishes the quality and scaling potential of this approach, but document retrieval (routing) eclipses attention in memory and compute costs, limiting throughput. Sparse access to memory does not imply cheap memory selection. At the beginning of each reasoning hop, each MSA routing layer compares its queries against the *routing keys* of all memory chunks to select documents for attention. Although attention consumes only the selected memory, the router repeats a corpus-wide similarity search in each of 18 layers on 1024-dimensional embedding vectors. This cost grows with both the number of searches and the width of their representations, and is paid each time the multi-hop retrieval expands the context. The routing-key bank also grows linearly with corpus size, making vector width a constraint on memory capacity as well as search cost.

We introduce **MonoRoute** (Figure 1), which exploits redundancy along both dimensions. At each reasoning hop, a single routing layer selects documents for all remaining routing layers, which still attend to their own KV states for those documents. The next hop can retrieve new evidence from the expanded context. We also reduce the routing vectors dimension from 1024 to 128. Together, these changes reduce the leading-order similarity-scoring work by $18\times(1024/128)=144\times$ for the same corpus and query length. End-to-end speedup also depends on attention, generation, and memory transfer.

Projection symmetrization makes the rank reduction possible in our construction. MSA learns distinct projections for the routing queries and keys. We derive a **shared orthogonal projection** for both from a symmetric positive-semidefinite approximation of the bilinear form induced by the routing projections on the space of hidden states, and then retain a rank-128 representation without retraining. This reduces the work per comparison by $8\times$ and cuts routing-key storage eightfold (Figure 4(e)). The smaller key bank directly lowers the memory cost of scaling MSA to larger document collections, complementing the reduction in searches from single-layer routing.

Our contributions are:

- **Single-layer retrieval over external document memory.** At each reasoning hop, we reuse one attention layer's document selection across all 18 routing layers, retaining layer-specific attention and the ability to retrieve new evidence at later hops.
- **Symmetrized low-rank retrieval.** We derive a shared orthogonal query and key projection from the symmetrized routing bilinear form. Reducing its width from 1024 to 128 without retraining cuts routing-key storage eightfold, freeing memory for larger document banks (Figure 4(e)).

With MSA's 64-token pooling and fp16 vectors, the routing index occupies nominally 4 bytes per source token, or 4 GB for one billion tokens. Our evaluation spans five single- and multi-hop QA datasets, including controlled KILT-Wikipedia corpora up to one billion tokens (Petroni et al., 2020). Section 4.2 compares answer quality and throughput with FlashRAG; Sections 4.3 and 4.4 isolate the contributions relative to MSA.

2 Related Work

Retrieval and latent memory. RAG retrieves passages that a generator reads as context (Lewis et al., 2020). Its defining interface is textual evidence; retrieval and generation need not be trained independently. Unlimiformer (Bertsch et al., 2023) retrieves encoder hidden states for cross-attention through a single nearest-neighbor index, with separate token selections for each decoder head and layer. MSA instead selects precomputed document KV states through learned transformer projections and uses Memory Interleave to condition later retrieval on accumulated evidence (Chen et al., 2026b). INTRA similarly retrieves evidence from reusable encoder states using decoder attention queries (Hoffer et al., 2026).

Knowledge bases in attention. KBLaM and AtlasKV also expose external knowledge directly to attention, but begin with structured facts. KBLaM maps knowledge triples to KV vectors through a sentence encoder and learned adapters, then integrates them with rectangular attention (Wang et al., 2025). AtlasKV combines knowledge-graph-derived KV representations with hierarchical pruning to support billion-triple knowledge graphs (Huang et al., 2026). MonoRoute instead operates on MSA's memory of unstructured documents. Appendix Table 5 summarizes these memory interfaces.

Sparse attention and shared routing. Sparse attention separates selecting relevant cache entries from attending to them. DeepSeek-V4, GLM-5, and MiniMax Sparse Attention use lightweight indexers to select entries or blocks (DeepSeek-AI, 2026; GLM-5 Team, 2026; Lai et al., 2026). Selection can nevertheless

retain a linear scan over the cache, as analyzed by IndexCache (Bai et al., 2026). Cross-layer reuse reduces how often this selection runs: IndexCache offers training-free and training-aware reuse of top-k selections across layers, while CLSA shares a single routing index across cross-decoder layers in a KV-sharing architecture (Sun et al., 2026). MonoRoute combines selection reuse with a shared symmetrized low-rank projection for MSA's external document memory. This construction retains layer-specific content KV states and renews document selection at each reasoning hop, without retraining the model.

Vector search over KV caches. RetrievalAttention already uses CPU-resident ANN indexes to retrieve attention keys, addressing the distribution mismatch between queries and keys (Liu et al., 2024). RetroInfer combines an attention-aware vector index with coordinated CPU–GPU cache management (Chen et al., 2026a); FlashMemory uses lookahead selection to retain query-relevant KV chunks on GPU (Wang et al., 2026). These systems accelerate access to a sequence's KV cache; our setting is an external document memory encoded before queries arrive.

3 METHODOLOGY

3.1 BACKGROUND: RECTANGULAR ATTENTION AND MEMORY SPARSE ATTENTION

Our method MonoRoute builds on MSA (Chen et al., 2026b), which is a trained framework designed for massive question-answering (QA) tasks, unifying the functionalities of large language models (LLM) and retrieval augmented generation (RAG) systems.

MSA and our method MonoRoute achieve memory-augumented generation through a decoder-only transformer architecture whose self-attention mechanism in the deeper half of the layers (referred to as *routing layers*) is modified by prefixing the input context with entries from an external memory bank in the form of KV cache states, as illustrated in Figure 2. Thus, in addition to self-attention, the context tokens also attend to these memory entries, integrating the external knowledge into their state.

Among related works, this fundamental design was used in KBLaM (Wang et al., 2025) as *rectangular attention* and is shared by its successor, AtlasKV (Huang et al., 2026). Implementing this design requires two phases: the *memory encoding phase*, during which the raw memory bank entries are mapped to vector representations in the form of KV cache tensors compatible with the rectangular attention, and the *deployment phase*, during which QA tasks can be processed at scale.

Memory encoding phase. The raw knowledge base is assumed to consist of plain-text documents. Each document is encoded by passing it through MSA's transformer layers and the KV cache states from the routing layers are stored[1] in the memory bank. Since the same transformer weights are used to process the documents and the user prompt (henceforth referred to as *question* due to the QA specialization), the document representations stored in the memory bank are naturally compatible with the corresponding routing layers' rectangular attention. This memory encoding phase must be completed before the system can be used to process user questions.

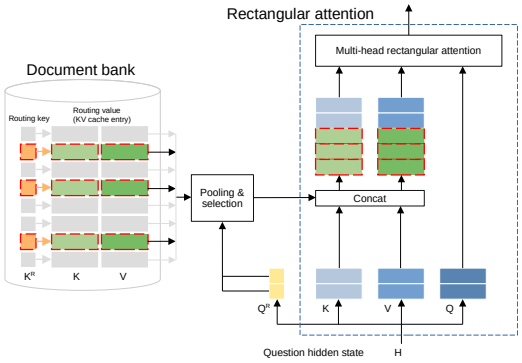

Figure 2: The architecture of an MSA rectangular attention module used in routing layers. Adapted from Chen et al. (2026b).

Deployment phase. In each routing layer ℓ, MSA selects a small set of documents relevant to the question by mapping the attention hidden state input H^ℓ to so-called *routing keys* K_ℓ^R (for the documents) and *routing queries* Q_ℓ^R (for the question). The routing keys are then compared against the routing query using cosine or inner-product similarity and pooling over the token and head dimensions, thus returning the top-scoring documents for the given question. The routing keys and queries are computed using trainable linear *routing projections* $W_{K^R}^\ell$ and

[1] MSA also compresses the KV representations of documents through mean-pooling.

$W^{\ell}_{Q^R}$:

$$K^R_\ell = H^\ell W^\ell_{K^R}, \qquad Q^R_\ell = H^\ell W^\ell_{Q^R}. \tag{1}$$

This sparse routing mechanism makes MSA's cost of generation proportional to the combined size of the retrieved documents rather than the total size of the entire knowledge base.

The authors of MSA have implemented their design on top of the Qwen3-4B-Instruct-2507 (Qwen Team, 2025) model. Thus, with zero-based layer numbering, the routing layers are 18–35 out of a total of 36 layers. The decoder and routing projection weights were fine-tuned using supervised contrastive learning (Khosla et al., 2020) with the objective of autoregressively generating the unique document IDs of relevant documents. Thus, the system effectively acts as a re-ranker of the documents prefixed in the routing layers.

MSA's scalability bottleneck. While the *generation cost* of MSA is proportional only to the size of the retrieved KV content, in order to find the top-scoring documents for a given question MSA performs a full scan of the document index in each routing layer. Thus, the *routing cost* is still linear in the size of the document bank as well as the number of routing layers. This bottleneck did not hinder the deployment of MSA at the 100M-token scale, since the authors managed to fit the routing keys in GPU VRAM, thus profiting from the high on-device bandwidth. However, this approach is wasteful of the hardware resources, since most of the GPU VRAM ends up being occupied by the routing key store, even as the probability that any particular document will be relevant for a question tends to zero with increasing document bank size.

3.2 Contributions of MonoRoute

In this work, we address MSA's routing bottleneck by replacing its multiple dependency-chained, linear scan-based routing steps with a single-layer low-rank retrieval step, which embeds the attention hidden states of both the questions and the documents using the same projection matrix, thus emulating the embedding symmetry between queries and documents that is typical of RAG systems.

Our proposed solution is twofold. First, we introduce *projection symmetrization* which allows us to reduce the dimensionality of the routing keys and queries and embed them in the same space. Second, we reuse the retrieval decisions of the single shallowest routing layer in all routing layers, thereby reducing the memory footprint and latency of document retrieval. Together with MSA's 64-token pooling and fp16 vectors, these optimizations allow us to run memory-augmented generation with a total document bank size of one billion tokens with a retrieval index whose size is only 4 bytes per token (128×2 bytes$/64$ tokens). We now discuss each of these contributions in turn.

3.2.1 Routing projection symmetrization

In each routing layer, MSA retrieves documents with the highest scoring fixed-length token chunks. For a chunk c, denote by $K^R_{c,h,\ell}$ the portion of its routing key for the layer ℓ in the attention head h. The score of c against the user question is computed according to

$$S_{c,\ell} := \max_{\text{token } t} \operatorname*{mean}_{\text{head } h} \langle Q^R_{t,h,\ell}, K^R_{c,h,\ell} \rangle = n_h^{-1} \max_{\text{token } t} \langle Q^R_{t,\cdot,\ell}, K^R_{c,\cdot,\ell} \rangle, \tag{2}$$

where $Q^R_{t,h,\ell}$ denotes the routing query vector at question token t, head h, and layer ℓ, and n_h denotes the number of heads. The equivalence over the mean of the per-head inner-products[2] and the inner-product over the grouped $Q^R_{t,\cdot,\ell}$ and $K^R_{c,\cdot,\ell}$ is due to the linearity of the mean over the heads.

Recall from (1) that the routing keys and queries are linear projections of the hidden state H^ℓ, and let d_R denote the total routing dimension across heads. By (2), retrieval in layer ℓ is a maximum-inner-product search over all chunk tokens in a d_R-dimensional space, so both the storage of routing keys for the entire corpus and the cost of scoring against them scale linearly with d_R. We therefore seek projections of rank $r \ll d_R$ that preserve the ranking of chunks.

To motivate our construction, consider the inner product similarity between the routing key of a chunk c and the routing query of a question token t. We have

$$\langle Q^R_{t,\cdot,\ell}, K^R_{c,\cdot,\ell} \rangle = \sum_{\text{head } h} \langle H^\ell_t W^{\ell,h}_{Q^R}, H^\ell_c W^{\ell,h}_{K^R} \rangle = \sum_{\text{head } h} \langle H^\ell_t, H^\ell_c W^{\ell,h}_{K^R} (W^{\ell,h}_{Q^R})^\top \rangle = \langle H^\ell_t, H^\ell_c M \rangle, \tag{3}$$

[2] MSA uses the cosine similarity during training, but plain inner-product during inference.

where $M := \sum_{\text{head } h} W^{\ell,h}_{K^R} (W^{\ell,h}_{Q^R})^\top$. Denote by $d = 2560$ the dimension of M, i.e., the dimension of the hidden states. By considering the singular value decomposition of $M = UDV^\top$, we have found that setting $\tilde{W}_{K^R} = (U\sqrt{D})_{1:r}$ and $\tilde{W}_{Q^R} = (V\sqrt{D})_{1:r}$, the matrix M admits a very faithful low-rank approximation $M \approx \tilde{W}_{K^R}(\tilde{W}_{Q^R})^\top$ for ranks as low as $r = 128$. Thus, replacing the original key and query projection matrices with $\tilde{W}_{K^R}$ and $\tilde{W}_{Q^R}$ in the experiments already yielded the desired dimensionality reduction while preserving full accuracy. In this work, however, we pursue a different approach by showing that the bilinear form (3) is well approximated in practice by a low-rank **symmetric positive semidefinite** matrix $WW^\top$, allowing us to embed the **keys and queries in a common space** without sacrificing model accuracy. Compared to the SVD approach, ours is a stronger and more surprising result, as symmetry between the keys and queries is a rigid structural constraint which is a priori never present in attention-based systems. The symmetry was also targeted in order to **resolve the query–key distribution mismatch** known to degrade graph-based ANN indices (Liu et al., 2024), should such an index later be used instead of the flat search; in the present system, however, the operative benefit is the reduction in rank.

First, we define the symmetrized matrix $S := (M + M^\top)/2$. Second, we compute the eigenvalue decomposition $S = U\Lambda U^\top$, where U is an orthogonal matrix and Λ is a diagonal matrix with the eigenvalues of S sorted in decreasing order (from most positive to most negative). We will now exploit this decomposition to construct a projection matrix $W^{\ell,sym}$ of rank $r \le d$ for obtaining the *symmetrized* routing keys and queries according to

$$K^{sym}_{c,\ell} = H^\ell_c W^{\ell,sym}, \qquad Q^{sym}_{t,\ell} = H^\ell_t W^{\ell,sym}. \tag{4}$$

Note that the inner product similarity between the routing keys and queries compare as

$$\langle Q^R_{t,\cdot,\ell}, K^R_{c,\cdot,\ell}\rangle - \langle Q^{sym}_{t,\ell}, K^{sym}_{c,\ell}\rangle = \big\langle H^\ell_t, H^\ell_c (M - W^{\ell,sym}(W^{\ell,sym})^\top)\big\rangle, \tag{5}$$

and thus the deviation of the symmetrized key-query score from the original one depends on the approximation quality of the matrix M by the symmetric positive-semidefinite matrix $W^{\ell,sym}(W^{\ell,sym})^\top$.

In order to construct $W^{\ell,sym}$, we analyze the average magnitude of the hidden states along different eigenvectors of S over some representative set $\mathcal{D}$. Concretely, Figure 3 shows $\sqrt{|\Lambda_i| \mathbb{E}_{H\sim\mathcal{D}} |\langle H, U_i\rangle|^2}$ for $i \in \{1,\dots,d\}$, where we take $\mathcal{D}$ to be the chunks of the first 512 documents of the TriviaQA-10M dataset in the shallowest routing layer of MSA (transformer layer 18). We see that the majority of the hidden states energy is captured by the eigendirections corresponding to the largest eigenvalues in absolute value.

We thus analyze how well S and M can be approximated by retaining the top r eigenvalues of S largest in absolute value. Concretely, given a rank parameter $r \le d$, let $\mathcal{I}_r \subset \{1,\dots,d\}$ be an index set of size r selecting r eigenvalues of S largest in absolute value. Write $U_{\mathcal{I}_r} \in \mathbb{R}^{d\times r}$ for the matrix whose columns are $(U_i)_{i\in\mathcal{I}_r}$ and $\Lambda_{\mathcal{I}_r} = \mathrm{diag}\big((\Lambda_i)_{i\in\mathcal{I}_r}\big)$, and set

$$S_r := U_{\mathcal{I}_r} \Lambda_{\mathcal{I}_r} U^\top_{\mathcal{I}_r}, \tag{6}$$

the rank-r truncation of S.

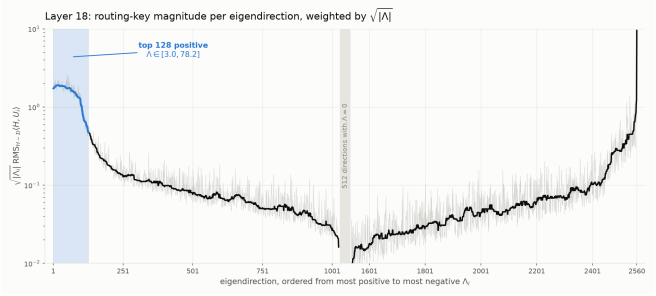

Figure 3: $\sqrt{|\Lambda_i| \mathbb{E}_{H\sim\mathcal{D}} |\langle H, U_i\rangle|^2}$ for each eigendirection U_i of S in layer 18 of MSA (shallowest routing layer). The values are sorted from most positive eigenvalues on the left to most negative eigenvalues on the right. Due to its construction, the matrix S has 1024 positive and 1024 negative eigenvalues, as well as 512 eigenvalues which are exactly zero.

We write $n^+_r = \big|\{i \in \mathcal{I}_r : \Lambda_i > 0\}\big|$ and $n^-_r = \big|\{i \in \mathcal{I}_r : \Lambda_i < 0\}\big|$. Recall from (5) that the difference in retrieval induced by a different routing projection W depends on how closely $WW^\top$ approximates M. However, the Frobenius error $\|M - WW^\top\|_F$ is not by itself the relevant notion of accuracy here, since document retrieval depends only on the *order* the scores induce. Accordingly, we will measure how closely S_r reproduces the score ordering of M through two metrics, the *Pearson correlation of scores* and the *median rank reproduction* across documents at a fixed question. Concretely, for a scoring matrix $X \in \mathbb{R}^{d\times d}$ let

$$\mathrm{sc}_X(q,c) = \max_{t\in q} \max_{\text{chunk } j\in c} \big\langle H^\ell_t, H^\ell_j X\big\rangle \tag{7}$$

be the score (up to the averaging constant) that document c receives for question q, where we will take X to be either the original M or the full symmetrization S. Now, for a set $\mathcal{Q}$ of questions and a set $\mathcal{C}$ of documents we define

$$\mathrm{corr}_X(S_r) = \underset{q\in\mathcal{Q}}{\mathrm{median}}\,\mathrm{Cov}_{c\in\mathcal{C}}\big(\mathrm{sc}_{S_r}(q,c),\mathrm{sc}_X(q,c)\big)\Big/\sqrt{\mathrm{Var}_c\big(\mathrm{sc}_{S_r}(q,c)\big)\mathrm{Var}_c\big(\mathrm{sc}_X(q,c)\big)}. \tag{8}$$

Next, writing $\mathcal{T}_k(X,q)$ for the k documents with the highest $\mathrm{sc}_X(q,\cdot)$ and $\mathrm{rk}_X(q,c)$ for the rank of document c in that ordering, we let

$$\mathrm{medrk}_X(S_r) = \underset{q\in\mathcal{Q},\,c\in\mathcal{T}_{16}(S_r,q)}{\mathrm{median}}\ \mathrm{rk}_X(q,c), \tag{9}$$

the median position, in X's ordering, of a document that S_r places in its top-16. A truncation that reproduces the ordering exactly attains a medrk of 8.5. Table 1 gives the fidelity of the matrices S_r according to these metrics for the shallowest routing layer of MSA (transformer layer 18 of 36) where we take $\mathcal{Q}$ to be the 128 TriviaQA-10M questions and the empirical distribution is over the first 512 documents of the same dataset. Rank 128 already reproduces the ordering of the full symmetrized form almost exactly, and thereafter both measures sit at the ceiling imposed by symmetrization itself: $\mathrm{corr}(S,M)=0.943$ and a median rank of 9 for S's own top-16 under M. The residual is due to the antisymmetric part of M, which no symmetric approximation can recover at any rank.

The matrix S_{128}, however, has 19 negative eigenvalues, and therefore cannot be factored as $W^{\ell,sym}(W^{\ell,sym})^\top$, i.e., we cannot use it to define (4). On the other hand, if we take $\mathcal{J}_r = \{1,\dots,r\}$ to be the index set of the r most-positive eigenvalues, we can define $W^{\ell,sym} = U_{\mathcal{J}_r}\sqrt{\Lambda_{\mathcal{J}_r}}$. As shown in Appendix A.3, the negative eigenvalues predominantly capture the dataset average, and thus removing them has a regularizing effect.

Table 1: Fidelity of the rank-r symmetrized approximation (6) with respect to the original matrix M and the full symmetrized matrix S. For each rank r we additionally report how many of the retained eigenvalues are positive (n_r^+) and negative (n_r^-).

r	n_r^+	n_r^-	corr_S	corr_M	medrk_S	medrk_M
16	16	0	0.475	0.439	43	52
32	32	0	0.662	0.610	18	23
64	63	1	0.905	0.865	10	11
128	109	19	0.998	0.943	8	9
256	174	82	1.000	0.944	8	9
2048	1024	1024	1.000	0.943	8	9

Table 2: Fidelity of the two symmetrized low-rank approximations in the shallowest routing layer of MSA (transformer layer 18 of 36), measured by the metrics in (8) and (9).

	$U_{\mathcal{J}_r}\Lambda_{\mathcal{J}_r}U_{\mathcal{J}_r}^\top$				$U_{\mathcal{J}_r}U_{\mathcal{J}_r}^\top$			
r	corr_S	corr_M	medrk_S	medrk_M	corr_S	corr_M	medrk_S	medrk_M
16	0.475	0.439	43	52	0.474	0.440	43	52
32	0.662	0.610	18	23	0.656	0.605	18	23
64	0.835	0.784	10	12	0.786	0.751	11	13
128	0.937	0.869	9	10	0.859	0.808	10	11
256	0.937	0.869	9	10	0.846	0.802	10	11

For our final symmetrized routing projection we make one more regularization step, namely to dispense with the $\sqrt{\Lambda_{\mathcal{J}_r}}$ factor and simply set $W^{\ell,sym} = U_{\mathcal{J}_r}$, which we used in our experiments as the rank-128 projection symmetrization. Table 2 reports the ranking fidelity metrics for $U_{\mathcal{J}_r}\Lambda_{\mathcal{J}_r}U_{\mathcal{J}_r}^\top$ and $U_{\mathcal{J}_r}U_{\mathcal{J}_r}^\top$ at several values of r. We observe that the approximation quality saturates at rank $r=128$ and that dropping the scaling factors $\sqrt{\Lambda_{\mathcal{J}_r}}$ costs little in terms of the score correlation and median rank reproduction. We further prefer the flat variant as, since $U_{\mathcal{J}_r}$ has orthonormal columns, the map $H \mapsto HU_{\mathcal{J}_r}$ is an isometry on the retained subspace, i.e., Euclidean distances between projected hidden states are exactly the distances between their components in that subspace. The weighting $U_{\mathcal{J}_r}\sqrt{\Lambda_{\mathcal{J}_r}}$ instead scales coordinate i by $\sqrt{\Lambda_i}$, which over the retained spectrum $\Lambda\in[3.0,78.2]$ stretches the leading direction by a factor of 5.1 relative to the trailing one and concentrates a correspondingly larger share of every pairwise distance in a few coordinates. The pure orthogonal projection $U_{\mathcal{J}_r}$ on the other hand preserves the hidden states geometry and would be required for a graph-based index or a quantized representation. We therefore adopt the orthogonal projection $W^{\ell,sym}=U_{\mathcal{J}_r}$.

3.2.2 Single layer retrieval

In Chen et al. (2026b)'s implementation of MSA, there are 18 routing layers, namely the last half of the layers 18–35. As demonstrated in Table 3, we empirically observed that the shallowest routing layer

already contains the highest-scoring documents of the remaining layers in an overwhelming number of cases. Therefore, in MonoRoute, we score and select the top-16 documents only in layer 18, whereas in all remaining routing layers $\ell > 18$ we simply prefix the rectangular attention with the layer-specific content KV cache states of the same selected documents, thereby short-circuiting the retrieval in those layers. Since the MSA decoder is fine-tuned to autoregressively generate the unique document IDs of the most relevant documents it sees in the rectangular attention regardless of the order in which they were prefixed, the system effectively acts as a re-ranker and hence it suffices to include the same high-quality documents in every router layer.

Reducing the number of active routing score calculations from 18 to 1 yields significant memory and time savings for the retrieval mechanism. Moreover, since the document KV cache states required for the rectangular attention in all layers $\ell \geq 18$ are known all at once as soon as the routing selection in layer 18 is made, we can keep those layers' KV caches on disk and fetch them in bulk to the GPU once the selection has been made.

Table 3: Coverage of each router layer's own top-j selection by the selection made at layer 18, on TriviaQA-10M. Entries are $\mathbb{E}_{\text{question}}\left|\text{top}_j(\ell) \cap \text{top}_{16}(18)\right|/j$, the fraction of layer ℓ's j highest-ranked documents that layer 18 also retrieved. At $j = 1$ the agreement is at least 0.91 for every layer.

	router layer ℓ								
	19	20	21	24	27	30	33	34	35
$j=1$	0.99	0.96	0.98	0.91	0.96	0.95	0.93	0.94	0.91
$j=2$	0.98	0.96	0.96	0.89	0.92	0.89	0.88	0.88	0.88
$j=4$	0.96	0.93	0.92	0.84	0.87	0.82	0.82	0.83	0.81
$j=8$	0.90	0.86	0.84	0.72	0.72	0.69	0.68	0.67	0.65
$j=16$	0.70	0.66	0.64	0.51	0.49	0.44	0.44	0.44	0.41

We have additionally studied a more aggressive generalization of this optimization where we only prefix the selected documents' KV states in layers $\ell = 18, \ldots, m$ and revert the final layers' $\ell = (m+1), \ldots, 35$ rectangular attention to the standard self attention (i.e., not prefixing any states). As seen in Table 7, preliminary experiments suggest no loss of accuracy as long as $m \geq 29$ and a sharp deterioration for lower values of m. We plan to include this stronger optimization in the further development of MonoRoute.

4 Experiments

4.1 Setup

Benchmarks. We compare MonoRoute with FlashRAG on five QA datasets: single-hop MS MARCO (Bajaj et al., 2016) and TriviaQA (Joshi et al., 2017), and multi-hop HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). HotpotQA uses nested KILT-Wikipedia corpora of 100M, 500M, and 1B tokens; we denote this setting by KILT-HQA. Both systems use the same corpus, questions, and answer scorers; Appendix A.5 gives dataset and chunking details.

Models and retrieval. MonoRoute uses MSA-4B with the shared rank-128 projection and exact top-16 document retrieval at layer 18. FlashRAG uses e5-base-v2 to retrieve three documents and Qwen3-4B-Instruct-2507 for generation (Qwen Team, 2025). MSA-4B uses this backbone with additional memory-specific fine-tuning. On multi-hop tasks, MonoRoute re-routes after each document injection; FlashRAG uses IRCoT with three documents per round and at most ten rounds. Appendix A.7 gives checkpoint, hop-policy, and memory-placement details. This is an end-to-end system comparison; the MSA ablations separately test our routing changes.

Hardware and batching. Both systems run separately on the same node at per-GPU batch sizes $b \in \{1, 16\}$, with greedy decoding and at most 128 new tokens per generation step. At 1B tokens, routing keys occupy 4.89 GB, separately from the 1.41 TB of content KV states. Appendix A.7 specifies hardware and memory placement.

Metrics. We report *containment exact-match* (ContainsEM), *exact-match* (EM), token-level *F1*, and mean *LLM-judge* score. ContainsEM credits an answer containing a gold answer. The first three use SQuAD-style answer normalization and take the maximum score over a question's gold answers. For LLM-Judge, DeepSeek-V4.1-Flash assigns the system's answer a 0–5 rubric score given the question and gold answer, following the judge prompt in Appendix A of the MSA paper (Chen et al., 2026b). Except for MuSiQue, throughput is completed questions divided by client-side answer-loop wall time for both systems, including

query preparation and client overhead but excluding model/store/index loading and scoring. MuSiQue's timing boundary remains unverified (Appendix A.7).

4.2 Task accuracy and throughput

In Table 4, MonoRoute achieves higher scores on all four quality metrics across all seven configurations, including 72.4% versus 22.7% exact match on 2WikiMultiHopQA. Scaling KILT-HQA from 100M to 1B tokens reduces its judge score by 17%, compared with 45% for FlashRAG, leaving scores of 2.536 versus 1.090 at 1B. At $b = 16$, MonoRoute's throughput is 3.1–6.5× FlashRAG's on KILT-HQA and 2WikiMultiHopQA, and effectively tied on MS MARCO (1.02×). FlashRAG remains faster on TriviaQA at $b = 16$ and on both single-hop datasets at $b = 1$. The separately supplied MuSiQue timings favor MonoRoute by 4.9× at $b = 16$, subject to the timing caveat in Section 4.1.

Table 4: **Answer quality and throughput.** Cells report *FlashRAG / MonoRoute*; higher values are bold. Quality uses $b = 16$; throughput uses client-side answer-loop wall time, except for MuSiQue (timing boundary unverified; Section 4.1). Here b denotes per-GPU batch size.

	Quality				Throughput (quest. per sec.)	
Benchmark	ContainsEM	EM	token-F1	LLM-Judge	$b=1$	$b=16$
Single-hop						
MSMARCO 100M	0.133 / **0.414**	0.023 / **0.281**	0.291 / **0.571**	3.688 / **4.039**	**0.82** / 0.55	3.60 / **3.67**
TriviaQA 10M	0.859 / **0.875**	0.539 / **0.820**	0.653 / **0.840**	4.414 / **4.609**	**1.68** / 0.54	**1.74** / 1.42
Multi-hop						
KILT-HQA 100M	0.344 / **0.441**	0.177 / **0.372**	0.273 / **0.474**	1.975 / **3.056**	0.71 / **1.03**	0.82 / **3.97**
KILT-HQA 500M	0.306 / **0.383**	0.152 / **0.316**	0.238 / **0.404**	1.699 / **2.688**	0.65 / **0.83**	0.81 / **2.92**
KILT-HQA 1B	0.214 / **0.352**	0.097 / **0.288**	0.167 / **0.375**	1.090 / **2.536**	0.61 / **0.67**	0.74 / **2.28**
2WikiMultiHopQA	0.469 / **0.797**	0.227 / **0.724**	0.363 / **0.771**	2.548 / **4.253**	0.46 / **1.21**	1.10 / **7.15**
MuSiQue	0.131 / **0.310**	0.051 / **0.238**	0.110 / **0.333**	0.917 / **2.224**	0.58 / **0.89**	1.15 / **5.58**

4.3 Ablation of MonoRoute on single-hop QA

Figure 4 compares MSA with three successive modifications on 128 MS MARCO questions over approximately 100M tokens: *LT18* retrieves only at layer 18; *LT18-sym* adds the shared 1024-dimensional projection with equal eigendirection weights; *LT18-sym-r128* retains the 128 leading positive eigendirections. Appendix A.6 gives the protocol and additional measurements.

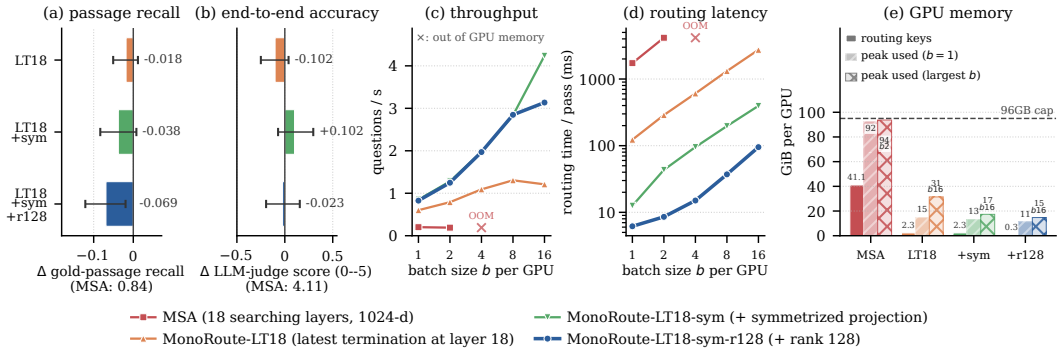

Figure 4: **Component ablation on MS MARCO 100M. (a,b)** Mean paired changes in selected-document recall and LLM-judge score (0–5) from MSA at $b = 1$ over 128 questions; error bars give 95% confidence intervals from bootstrap resampling of questions. **(c)** Throughput versus per-GPU batch size b ($2b$ concurrent questions); crosses mark out-of-memory runs. **(d)** Routing time per pass on the slower GPU. **(e)** Resident routing keys and peak GPU memory at $b = 1$ and the largest completed batch.

At $b = 1$, LT18 improves throughput 3.0× over MSA, with a 0.102-point judge-score decrease. LT18-sym scores 0.102 points above MSA and reduces routing time from 123 to 12.6 ms. The symmetrized

implementation also reduces query heads from 32 to eight, shrinking scoring tensors fourfold. It reaches 4.24 questions/s at $b=16$, 21× MSA's best tested throughput.

Rank reduction changes the judge score by -0.023 points relative to MSA and cuts resident routing keys from 2.3 to 0.29 GiB per GPU (41.1 GiB for MSA). At $b=1$, routing takes 6.2 ms per pass, a 280× reduction relative to MSA. Question throughput improves 4.1×, with peak GPU memory of 11 versus 92 GiB. At $b=16$, however, its throughput is lower than LT18-sym's (3.14 versus 4.24 questions/s): longer selected documents increase processing time and straggler effects (Appendix A.6).

4.4 Ablation of MonoRoute on multi-hop QA

Figure 5 evaluates the same four variants on 1,000 HotpotQA questions over 100M tokens of KILT-Wikipedia. Every variant re-routes after each document injection. Quality is measured at $b=1$; Appendix A.6 details the protocol and effects of batching.

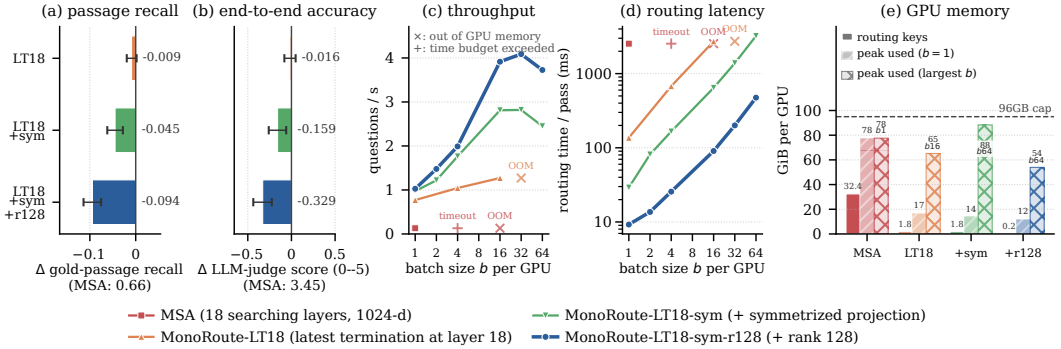

Figure 5: **Component ablation on KILT-HotpotQA 100M.** Panels, batch conventions, and confidence intervals follow Figure 4; quality uses 1,000 paired questions at $b=1$. Crosses mark out-of-memory runs; the plus sign marks a run exceeding the 150-minute time limit.

At $b=1$, LT18 improves throughput 5.9× with a 0.016-point judge-score decrease relative to MSA. LT18-sym and LT18-sym-r128 decrease the score by 0.159 and 0.329 points, respectively, and selected-document recall by 0.045 and 0.094. These results support reusing one layer's selection within each hop while retrieving fresh evidence at later hops. Projection compression introduces a larger accuracy trade-off than on MS MARCO: the rank-128 variant scores 3.12 versus MSA's 3.45.

The rank-128 variant reaches 1.03 questions/s at $b=1$ (7.9× MSA), reduces routing time 276× to 9.2 ms per pass, and uses 0.23 GiB of routing keys per GPU versus MSA's 32.4 GiB; peak GPU memory falls from 78 to 12 GiB. Its best throughput is 4.09 questions/s at $b=32$, 31× MSA's best tested throughput and 45% above LT18-sym's 2.82 at the same batch size.

The memory savings also permit larger batches. MSA runs out of memory at $b=16$ and LT18 at $b=32$; both symmetrized variants fit at $b=64$, although their throughput peaks at $b=32$. At $b=64$, rank reduction lowers peak GPU memory from 88 to 54 GiB, saving a further 34 GiB per GPU even at this larger batch.

5 Conclusion

We advocate attention-based retrieval as a competitive alternative to RAG, coupling evidence selection and multi-hop reasoning within the same model. MonoRoute advances this direction by eliminating repeated searches in MSA's highly redundant routing design: a **single** attention layer retrieves documents at each hop, replacing 18 layer-wise searches, while all routing layers retain attention to their own document KV states. Projection symmetrization unifies the routing-query and routing-key projections in one shared **orthogonal projection** and enables rank reduction from 1024 to 128, cutting routing-key storage eightfold without retraining. This compression lowers the memory cost of scaling MSA's document bank. Together, these contributions reduce leading-order similarity-scoring work by 144× while preserving fresh evidence selection at successive hops. The multi-hop ablation supports single-layer routing with little quality loss, while exposing an accuracy trade-off from projection compression. These results strengthen attention-based retrieval as a practical approach to reasoning over external knowledge with lower retrieval costs.

AI USE STATEMENT

In this work, we used generative AI tools to *implement methods*. We have not used generative AI tools for *generating synthetic datasets, design or provide feedback on research methodology or experiments, clean and reformat dataset, support qualitative and thematic data analysis, interpret results* and *help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, assist with translation* are not applicable to this work. Additionally, we used generative AI tools to *create or modify scientific figures and to edit the paper to improve readability*. We have reviewed all AI-assisted work: we checked LLM-generated software implementations through manual code review and manual execution thereof; data in the plots was verified against concrete numbers measured in experiments. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

MonoRoute studies efficient attention-based retrieval using existing benchmark datasets and pretrained models. We did not recruit human participants or collect new personal data. Reducing retrieval costs may broaden access to memory-augmented models, but can also lower the cost of harmful uses. Faster retrieval does not itself prevent biased or unsupported outputs, and access to stored memories can expose sensitive source content. Our accuracy and throughput measurements do not establish safety in deployment. Applications should respect dataset and model licenses, enforce access controls over both source documents and derived KV memories, and evaluate factuality and harmful outputs in their intended setting.

REPRODUCIBILITY STATEMENT

We will provide the code for KILT dataset setting and of our experiments as supplementary material through the submission system. Our experiments use the publicly available `EverMind-AI/MSA-4B` checkpoint. Appendix A.5 specifies the corpora and question sets; Appendices A.6 and A.7 document ablation protocols, serving settings, and memory placement. Section 4.1 specifies the answer metrics and LLM-judge model and prompt source.

REFERENCES

Yushi Bai, Qian Dong, Ting Jiang, Xin Lv, Zhengxiao Du, Aohan Zeng, Jie Tang, and Juanzi Li. IndexCache: Accelerating sparse attention via cross-layer index reuse, 2026. URL https://arxiv.org/abs/2603.12201.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. Unlimiformer: Long-range transformers with unlimited length input. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 35522–35543. Curran Associates, Inc., 2023. doi: 10.52202/075280-1543. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6f9806a5adc72b5b834b27e4c7c0df9b-Paper-Conference.pdf.

Yaoqi Chen, Jinkai Zhang, Baotong Lu, Qianxi Zhang, Chengruidong Zhang, Jing Liu, Jingjia Luo, Di Liu, Huiqiang Jiang, Qi Chen, Bailu Ding, Xiao Yan, Jiawei Jiang, Chen Chen, Mingxing Zhang, Cheng Li, Yuqing Yang, Fan Yang, and Mao Yang. Retroinfer: A vector storage engine for scalable long-context llm inference. *Proceedings of the VLDB Endowment*, 19(5):1016–1031, January 2026a. ISSN 2150-8097. doi: 10.14778/3796195.3796212. URL http://dx.doi.org/10.14778/3796195.3796212.

Yu Chen, Runkai Chen, Sheng Yi, Xinda Zhao, Xiaohong Li, Jianjin Zhang, Jun Sun, Chuanrui Hu, Yunyun Han, Lidong Bing, Yafeng Deng, and Tianqiao Chen. MSA: Memory sparse attention for

efficient end-to-end memory model scaling to 100M tokens, 2026b. URL https://arxiv.org/abs/2603.23516.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

GLM-5 Team. GLM-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Elad Hoffer, Yochai Blau, Edan Kinderman, Ron Banner, Daniel Soudry, and Boris Ginsburg. Retrieval from within: An intrinsic capability of attention-based models, 2026. URL https://arxiv.org/abs/2605.05806.

Haoyu Huang, Hong Ting Tsang, Jiaxin Bai, Xi Peng, Gong Zhang, and Yangqiu Song. Atlaskv: Augmenting llms with billion-scale knowledge graphs in 20gb vram, 2026. URL https://arxiv.org/abs/2510.17934.

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 (ACL)*, pp. 1601–1611, 2017.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18661–18673, 2020.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, Jinkai Hu, Jiayao Li, Rui Gao, Zekun Li, Songquan Zhu, Jingkai Zhou, and Pengyu Zhao. MiniMax sparse attention, 2026. URL https://arxiv.org/abs/2606.13392.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rockt"aschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. Retrievalattention: Accelerating long-context llm inference via vector retrieval, 2024. URL https://arxiv.org/abs/2409.10516.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. KILT: a benchmark for knowledge intensive language tasks, 2020. URL https://arxiv.org/abs/2009.02252.

Qwen Team. Qwen3 technical report. Technical report, Alibaba Group, May 2025. URL https://arxiv.org/abs/2505.09388.

Yutao Sun, Yanqi Zhang, Li Dong, Jianyong Wang, and Furu Wei. You only index once: Cross-layer sparse attention with shared routing, 2026. URL https://arxiv.org/abs/2606.06467.

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

Xi Wang, Taketomo Isazawa, Liana Mikaelyan, and James Hensman. Kblam: Knowledge base augmented language model, 2025. URL https://arxiv.org/abs/2410.10450.

Yan Wang, Qifan Zhang, Jiachen Yu, Tian Liang, Dongyang Ma, Xiang Hu, Zibo Lin, Chunyang Li, Zhichao Wang, Miao Peng, Nuo Chen, Jia Li, Yujiu Yang, Haitao Mi, and Dong Yu. Flashmemory-deepseek-v4: Lightning index ultra-long context via lookahead sparse attention, 2026. URL https://arxiv.org/abs/2606.09079.

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

A Appendix

A.1 External-memory interfaces

Table 5: External-memory interfaces of the methods reviewed in Section 2. Knowledge triples and source-document tokens measure different units of capacity.

Method	Evidence supplied to the model	Memory access
RAG	Retrieved text passages	External retriever
KBLaM	Encoded knowledge triples	Dense rectangular attention
AtlasKV	Encoded knowledge-graph facts	Hierarchical KV pruning
MSA	Document KV states	18 routing layers, 1024D
MonoRoute	Document KV states	1 routing layer, 128D

A.2 MonoRoute Architecture

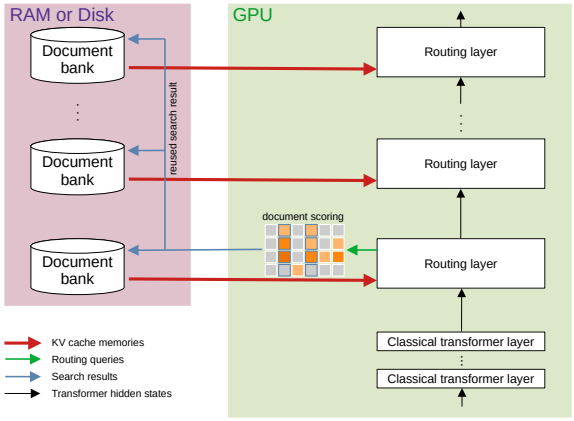

Figure 6: The architecture of MonoRoute. Document retrieval scoring is performed only in the first routing layer.

A.3 Negative Eigenvalues

We show that the negative eigenvectors of S are qualitatively different from the positive ones, in that they very weakly discriminate between various hidden states and largely capture their mean, and thus discarding them in the construction of $W^{\ell,sym}$ can be seen as a regularization step. To this end, let $U_{\mathcal{N}} \in \mathbb{R}^{d\times|\mathcal{N}|}$ collect the eigenvectors of a subspace $\mathcal{N}$. Let $\mathcal{D}$ be a set of hidden states, each written as a row vector so that its projection onto $\mathcal{N}$ is $HU_{\mathcal{N}} \in \mathbb{R}^{|\mathcal{N}|}$. We now define

$$\mathrm{shared}(\mathcal{D}) = \frac{\left\|\mathbb{E}_{H\sim\mathcal{D}}[HU_{\mathcal{N}}]\right\|^2}{\mathbb{E}_{H\sim\mathcal{D}}[\|HU_{\mathcal{N}}\|^2]} \in [0,1], \qquad \overline{\cos}(\mathcal{D}) \quad = \mathop{\mathbb{E}}_{\substack{H,H'\sim\mathcal{D}\\ H\neq H'}} \left[\frac{\langle HU_{\mathcal{N}}, H'U_{\mathcal{N}}\rangle}{\|HU_{\mathcal{N}}\|\,\|H'U_{\mathcal{N}}\|}\right]. \tag{10}$$

Since the empirical mean satisfies $\mathbb{E}_H\|HU_{\mathcal{N}}\|^2 = \left\|\mathbb{E}_H[HU_{\mathcal{N}}]\right\|^2 + \mathbb{E}_H\left\|HU_{\mathcal{N}} - \mathbb{E}_{H'}[H'U_{\mathcal{N}}]\right\|^2$, $\mathrm{shared}(\mathcal{D})$ is the fraction of the projections' energy carried by the component common to every $H \in \mathcal{D}$, and $1-\mathrm{shared}(\mathcal{D})$ is the fraction that varies across the dataset. A value of 1 means all projections coincide, so the subspace contributes a per-query constant to every document's score and cannot reorder them. Table 6 reports shared and $\overline{\cos}$ for eigenspaces corresponding to the 20 most negative and 20 most positive directions on $\mathcal{D}_{\mathrm{chunk}}$, the hidden states of the chunks of the first 512 documents of TriviaQA-10M, as well as $\mathcal{D}_{\mathrm{quest}}$, the question-token hidden states of the same dataset. We observe that in the 20 most negative directions both datasets are effectively very low-dimensional, whereas in the 20 most positive directions the data is well-separated.

Table 6: Hidden-state discrimination measures of the chunk and question projections in the 20 most negative and 20 most positive directions.

Subspace $\mathcal{N}$	$\mathcal{D}_{\text{chunk}}$		$\mathcal{D}_{\text{quest}}$	
	$\text{shared}(\mathcal{D})$	$\overline{\cos}(\mathcal{D})$	$\text{shared}(\mathcal{D})$	$\overline{\cos}(\mathcal{D})$
20 most negative λ	0.915	0.931	0.814	0.828
20 most positive λ	0.089	0.088	0.049	0.069

A.4 Capped single layer retrieval

Table 7: Capping retrieved-KV injection above a given router layer tested in a small experiment on the single-hop datasets MS MARCO 100M and TriviaQA 10M.

cap layer m		26	27	28	29	30	31	35
MS MARCO 100M	token-F1	0.0928	0.1273	0.1120	0.5767	0.5698	0.5722	0.5790
	citation F1	0.0000	0.0293	0.0085	0.8065	0.7986	0.7983	0.7866
TriviaQA 10M	token-F1	0.3858	0.6131	0.6999	0.8222	0.8303	0.8452	0.8298
	citation F1	0.0551	0.2706	0.4090	0.5801	0.5806	0.5771	0.5354

A.5 Benchmark details

MonoRoute and FlashRAG use the same corpus, questions, and answer scorers for each benchmark. Table 8 summarizes corpus and question statistics, Table 9 records the methods' respective memory granularities, and Table 10 reports retrieval and generation behaviour.

Single-hop QA. MS MARCO (Bajaj et al., 2016) provides passage-level QA, while TriviaQA (Joshi et al., 2017) provides open-domain factoid QA. We evaluate 128 questions from each, using corpora of approximately 100M and 10M tokens, respectively.

Multi-hop QA. HotpotQA (Yang et al., 2018) uses 1,000 questions filtered to require at least two distinct gold Wikipedia pages. The corresponding KILT-Wikipedia corpora (Petroni et al., 2020) contain 200-word passages and are nested at 100M, 500M, and 1B tokens, with larger corpora adding distractors. This setting tests retrieval and reasoning as the document bank grows.

For 2WikiMultiHopQA (Ho et al., 2020), we use the 1,000-question benchmark released with MSA (Chen et al., 2026b), covering comparison, inference, compositional, and bridge-comparison questions. Its 722K-token corpus is the deduplicated union of the questions' Wikipedia paragraphs; 765 questions have two gold documents and 235 have four. The memory model uses exactly two retrieval rounds for 99.5% of these questions.

MuSiQue (Trivedi et al., 2022) provides a held-out multi-hop stress test with compositional 2–4-hop questions. We evaluate 999 questions over an 11,656-document corpus.

Table 8: **Benchmarks: corpora and questions.** Token counts use the Qwen3 tokenizer; benchmark names indicate build targets. Tok./doc and Gold docs/q are mean tokens per document and gold provenance documents per question.

Benchmark	Task	Tokens	Documents	Tok./doc	Questions	Gold docs/q
MSMARCO 100M	single-hop	105 708 601	961 686	110	128	1.30
TriviaQA 10M	single-hop	10 486 478	12 589	833	128	6.12
KILT-HQA 100M	multi-hop	104 525 989	340 476	307	1 000	2.00
KILT-HQA 500M	multi-hop	525 912 906	1 722 966	305	1 000	2.00
KILT-HQA 1B	multi-hop	1 048 855 396	3 446 270	304	1 000	2.00
2WikiMultiHopQA 722K	multi-hop	722 701	6 119	118	1 000	2.47

Table 9: **Benchmarks: memory granularity.** Number of 64-token chunks MonoRoute pools each corpus into (one routing key per chunk) and number of 256-token chunks FlashRAG embeds, with the resulting chunks per document.

Benchmark	MonoRoute chunks	per doc	FlashRAG chunks	per doc
MSMARCO 100M	2 394 213	2.5	962 313	1.0
TriviaQA 10M	172 653	13.7	47 477	3.8
KILT-HQA 100M	1 888 983	5.5	604 182	1.8
KILT-HQA 500M	9 550 036	5.5	3 044 092	1.8
KILT-HQA 1B	19 088 643	5.5	6 077 817	1.8
2WikiMultiHopQA 722K	14 219	2.3	6 869	1.1

Table 10: **Retrieval and generation at batch 16.** Paired entries report *FlashRAG / MonoRoute*. Rounds/q counts IRCoT iterations or Memory-Interleave hops. Selected-document recall compares gold documents with FlashRAG's retrieved documents across rounds or MonoRoute's generated document IDs. Generated tokens exclude injected text; time/q is client-side answer-loop wall time divided by completed questions, excluding loading and scoring. Speed-up is MonoRoute throughput divided by FlashRAG throughput. The last column reports ContainsEM at $b=1$.

Benchmark	Rounds/q	Sel. recall	Gen. tok./q	Time/q (s)	Speed-up	Contains-EM b=1
MSMARCO 100M	1.00 / 1.00	0.58 / 0.79	58 / 76	0.28 / 0.27	1.02×	0.133 / 0.430
TriviaQA 10M	1.00 / 1.00	0.50 / 0.60	24 / 78	0.57 / 0.71	0.81×	0.852 / 0.867
KILT-HQA 100M	3.22 / 1.46	0.86 / 0.56	29 / 54	1.22 / 0.25	4.83×	0.289 / 0.453
KILT-HQA 500M	3.09 / 1.47	0.79 / 0.46	30 / 55	1.24 / 0.34	3.62×	0.213 / 0.390
KILT-HQA 1B	3.35 / 1.48	0.72 / 0.40	31 / 56	1.35 / 0.44	3.09×	0.152 / 0.361
2WikiMultiHopQA 722K	2.87 / 2.00	0.88 / 0.86	26 / 66	0.91 / 0.14	6.50×	0.466 / 0.799

A.6 Ablation protocol and additional measurements

The component ablations use two GPUs and measure quality at $b=1$, whereas Table 4 measures quality at $b=16$. All MSA-relative speedups use measurements from the corresponding ablation. Selected-document recall is the fraction of gold documents identified by the model's generated document IDs, distinct from recall among the router's top-16 candidates.

Scope of the ablations. The variants retain the same MSA checkpoint and layer-specific content KV states. Single-layer routing retains only layer 18's routing-key bank, reducing resident routing keys approximately $18\times$; rank reduction supplies a further $8\times$ reduction. The smaller reductions in peak GPU memory reflect other allocations, including model weights, content states, and scoring tensors. The cross-layer coverage in Table 3 motivates selection reuse: layer 18 covers 91–99% of later layers' first-ranked documents, although coverage of the full top-16 falls to 41% at layer 35. The HotpotQA ablation tests the effect of this reuse on multi-hop answers; neither coverage nor small mean score changes establish statistical equivalence.

The projection ablation evaluates the shared projection together with equal eigendirection weights and the reduction from 32 query heads to eight. Its speedup cannot be attributed to symmetry alone. The signed approximation S_{128} in Table 1 has score correlation 0.943 with M, but is a diagnostic rather than the deployed projection. The deployed positive-subspace projection has correlation 0.808 (Table 2); its answer quality is measured directly in Figures 4 and 5. These experiments do not compare against a nonsymmetric rank-128 factorization of M, so they establish the effectiveness of this construction rather than its optimality among low-rank alternatives.

Single-hop QA: MS MARCO 100M. All four ablation variants use the same content KVs, exact scans in 16,384-chunk blocks, top-16 selection, greedy decoding, and MSA's Memory Interleave. Throughput averages two runs after warm-up using client-side answer-loop wall time, excluding memory construction, loading, and scoring. At $b=1$, MSA and LT18 achieve 0.201 and 0.601 questions/s, respectively; LT18's passage recall decreases by 0.018 relative to MSA. LT18-sym's best throughput is $21\times$ MSA's best tested throughput. For LT18-sym-r128, routing takes 6.2 ms per pass at $b=1$, a $280\times$ reduction relative to MSA, and throughput is approximately 0.83 questions/s ($4.1\times$ MSA). Its absolute passage recall is 0.769. MSA

runs out of memory above $b=2$. At $b=16$, LT18's routing pass takes 2.7 s, and its throughput peaks at approximately 1.3 questions/s over the tested batches. LT18 and LT18-sym each retain 2.3 GiB of routing keys per GPU, but LT18-sym's smaller scoring tensors reduce peak memory at $b=16$ from 31 to 17 GiB. At that batch, the rank-128 variant reaches 3.14 questions/s versus LT18-sym's 4.24; its selected passages are longer and incur more padding.

Multi-hop QA: KILT-HotpotQA 100M. The same four variants, content KVs, exact scans in 16,384-chunk blocks, top-16 selection and greedy decoding are run on 1000 HotpotQA questions, each with two gold pages, over a 100M-token memory of 200-word Wikipedia documents from KILT. Memory Interleave runs with the speculative hop turned *off* for every variant: after each document injection the enriched context is prefilled again and routed afresh, so every hop is a full routing pass (1.5 injections per question on average, 2 for 48–59% of questions across variants). Each configuration is run once with a 150-minute limit. Batch sizes 1, 4, 16 and 32 are tested in ascending order until an out-of-memory failure; $b=64$ and $b=2$ are also tested for variants that complete this sequence. Quality is measured at $b=1$ because batched serving delays answer generation until every question has finished retrieving and then makes an additional generation call. At $b=1$, MSA and LT18 achieve 0.130 and 0.770 questions/s; LT18's passage recall decreases by 0.009 relative to MSA. MSA is slower at $b=4$ (0.098 questions/s over the 800 questions answered within the budget) and runs out of memory at $b=16$. LT18-sym's best throughput (2.82 questions/s at $b=32$) is 21.7× MSA's best tested throughput. For LT18-sym-r128, routing takes 9.2 ms per pass at $b=1$, a 276× reduction relative to MSA, and throughput is 1.03 questions/s at $b=1$ (7.9× MSA) and 4.09 at $b=32$ (31×). Its absolute passage recall is 0.564 (0.658 for MSA) and its mean judge score 3.12 (3.45 for MSA).

First-hop top-16 router recall changes from 0.686 to 0.677 with symmetrization and to 0.595 after rank reduction. The latter two variants score 0.957 and 0.923 on MS MARCO. These candidate-recall measurements are distinct from the selected-document recall in panel (a).

Multi-hop routing and batching. Routing shares divide the profiler's summed routing time by worker generation time (routing, prefill, and decoding) on the slower GPU, including warm-up. At $b=1$, routing accounts for 88% of MSA's worker generation time, with approximately 2.5 s per pass; LT18-sym reduces this to approximately 29 ms. Later-hop queries include the accumulated context, reaching 2.7k tokens. For LT18 at $b=32$, a question that injects five passages produces an 82 GiB scoring tensor. The smaller tensors of LT18-sym and LT18-sym-r128 fit at $b=64$, with peak GPU memory of 88 and 54 GiB, respectively. Both achieve their highest throughput at $b=32$. At $b=64$, LT18-sym's routing takes 3.2 s per pass (44% of worker generation time); this share is 10% for LT18-sym-r128, whose throughput decreases by approximately 9%. Batched prefill and decoding pad sequences to the longest context in the batch. The additional generation call in batched serving changes approximately a quarter of answer strings and lowers the judge score by up to 0.3 points at $b=64$, without changing the selected passages.

A.7 Serving details

These serving configurations correspond to Table 4; the component-ablation protocols are specified in Appendix A.6. The FlashRAG comparison includes differences in fine-tuning, document budgets, chunk granularity, hop policies, and CPU versus GPU search. It establishes the measured quality and throughput of these complete systems; the MSA ablations test our routing changes under a common checkpoint and retrieval policy. Matched-budget RAG and alternative search-parallelism configurations are not evaluated.

Timing. Except for MuSiQue, both systems in Table 4 are timed over the complete question loop in the benchmark process, starting after model, memory-store, and index loading and ending before scoring. This includes query preparation and the full retrieval–generation loop. For MonoRoute, it covers chat templating and tokenization, the routing-query forward pass, exact key scans, content-KV fetching and assembly, prefill, decoding, document-text injection, request/result transfer to GPU workers, and client-side parsing and control between hops. For FlashRAG, it covers query embedding, FAISS search, prompt construction, generation, and IRCoT's retrieval and parsing across rounds. Ablation throughput also uses client-side answer-loop wall time; per-pass routing latency in panel (d) is measured separately on the slower GPU. Table 10 reports answer-loop wall time per completed question.

Model checkpoint and retrieval policy. We use `EverMind-AI/MSA-4B`, based on Qwen3-4B-Instruct-2507, with the rank-128 symmetrized projection and equal eigendirection weights. Layers are

Table 11: **Memory placement at inference.** L18 keys are routing keys; K and V are content states. Store size is the bf16 K+V footprint of all 18 routing layers; RAG index size counts float32 e5 embeddings in CPU DRAM. With two GPUs, MonoRoute shards its memory across them; FlashRAG runs one process per GPU on half of the questions.

Benchmark	GPUs	Store (GB)	L18 keys	L18 K	L18 V	L19–35 K,V	RAG index (GB)
MSMARCO 100M	1	177	GPU	GPU	DRAM	DRAM	3.0
TriviaQA 10M	1	13	GPU	GPU	DRAM	DRAM	0.1
KILT-HQA 100M	2	139	GPU	GPU	DRAM	DRAM	1.9
KILT-HQA 500M	2	704	GPU	DRAM	DRAM	disk (mmap)	9.4
KILT-HQA 1B	2	1407	GPU	DRAM	DRAM	disk (mmap)	18.7
2WikiMultiHopQA 722K	2	1	GPU	GPU	DRAM	DRAM	0.0
MuSiQue	2	2.3	GPU	GPU	DRAM	DRAM	0.04

numbered from zero; the 18 routing layers are 18–35. The setting `last_freeze=18` (LT18) restricts fresh similarity searches to layer 18, with later layers reusing its document selection. On single-hop tasks, generation continues after the first injection without re-routing or re-prefilling (the speculative hop); on multi-hop tasks, the model re-routes after every injection for as many rounds as it chooses. FlashRAG embeds 256-token chunks with e5-base-v2 and retains the three best distinct documents; its multi-hop configuration uses IRCoT as specified in Section 4.1.

Hardware. The shared node has two GPUs with 192 GB total VRAM, 2.28 TiB of host memory, and a 96-core CPU (192 threads). Both systems run one at a time on the otherwise idle node; Table 11 lists GPU use by benchmark.

Memory access. MonoRoute answers from a precomputed memory: for every document the encoder's K and V of the 18 router layers (layers 18–35 of the 36-layer backbone), pooled into 64-token chunks, together with one 128-dimensional routing key per chunk for the single searching layer. The first 18 layers attend only to the current sequence and hold no document memory. Table 11 lists where the precomputed tensors reside during the timed runs; in addition, every layer keeps the ordinary per-request KV cache of the current sequence (template prefix, question, injected document text and generated tokens) in GPU memory for the duration of the request, about 70 MB per sequence, and the router layers hold the copied-in chunks of the 16 selected documents for the current batch. The routing keys of layer 18 always sit in GPU memory, sharded when two GPUs are used, because every query scores them all (an exact scan in slices of 16,384 chunks, keeping a running top-k). Up to 100M tokens the whole store fits in host DRAM; the searching layer's content K is additionally kept on the GPU to speed up gathering. At 500M and 1B tokens, the total content stores occupy 0.7 and 1.4 TB; the 17 remaining routing layers are memory-mapped from local disk (`KV_DISK_OFFLOAD`); per question only the chunks of the 16 selected documents are read from them, about 6 MB, and the operating system's page cache keeps recently used pages in DRAM. Per query the selected chunks of every layer are copied to the GPU, assembled with the template prefix and the question into the sparse attention context, and released after the answer; the resident routing keys and any GPU-resident content K listed in Table 11 persist across questions. FlashRAG keeps its flat FAISS index (float32 e5 embeddings of all 256-token chunks) in CPU DRAM and scans it exhaustively on the CPU (32 threads at 500M/1B, FlashRAG's single-thread default at $\leq$100M); the retrieved documents enter the generator as prompt text, so it holds no per-document state on the GPU.

Page-cache conditions. At 500M and 1B tokens, we neither cleared the operating-system page cache nor explicitly preloaded the memory store. Runs at $b=16$ followed $b=1$ on the same questions and could reuse cached pages. Disk-read time and cache residency were not measured, so these results do not isolate batching gains from cache reuse.

Routing-key storage. The nominal 4 bytes per token counts a 128-dimensional, two-byte routing vector per full 64-token chunk. Using the actual chunk count in Table 9, the 1B-token experiment stores $19{,}088{,}643 \times 128 \times 2 = 4.89$ GB of routing keys, or approximately 4.66 bytes per source token. This payload excludes metadata and the 1,407 GB content-KV store in Table 11.