# RETRIEVAL LAYERS: FAST END-TO-END RETRIEVAL WITH SPARSE ATTENTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Language model applications commonly require handling massive amounts of external information, for example querying internal documentation, reasoning over codebases, or user personalization based on past interaction history. Retrieval-augmented generation (RAG) is the dominant mechanism for grounding models in such external information. However, standard RAG frameworks are post-hoc scaffolds around the model, using a brittle lookup that cannot be trained end-to-end. Furthermore, RAG incurs significant computational overhead from prefilling large volumes of text, in turn slowing down subsequent generation by increasing the context length. To address these limitations, we introduce Retrieval Layers, end-to-end learnable sparse cross-attention layers that conduct fast lookup of external text in latent space, rather than prefilling retrieved text into the model's context. Retrieval layers can be integrated into existing language models with minor post-training cost. We insert Retrieval Layers into an off-the-shelf Qwen3-4B and finetune it with custom SFT and RLVR datasets. On challenging multihop QA benchmarks, Retrieval Layers achieve Pareto-dominance on an accuracy versus end-to-end latency tradeoff in comparison to RAG baselines. We also show stronger performance than the Pareto frontier on long conversation reasoning and demonstrate that Retrieval Layers can extend the needle-in-a-haystack ability to 30M tokens.

## 1 INTRODUCTION

Large language models (LLMs) increasingly need to reason over large, evolving corpora of information, including internal documentation, codebases, and user interaction histories. These corpora can exceed an LLM's context window or change too frequently for fine-tuning to keep a model's parameters up-to-date (Gao et al., 2024). Currently, retrieval-augmented generation (RAG) (Lewis et al., 2020) is the dominant approach to filling this gap. It retrieves the subset of the corpus that is most relevant to a query and concatenates it to the LLM's context. The LLM must then compute keys and values of all the concatenated text for use in subsequent generation. This process is called prefilling and incurs an expensive quadratic cost. Furthermore, this large addition to the key-value (KV) cache increases peak memory usage and slows down the generation of future tokens. In agentic settings, where the LLM must keep track of the conversation, tool traces, and intermediate reasoning, this concatenation only further exacerbates the issue of limited context sizes (Packer et al., 2023). RAG thus faces an accuracy-efficiency tradeoff, between improving the recall of positive information needed to answer a query, and the quadratic costs of retrieving and concatenating additional tokens (Taguchi et al., 2025). This tradeoff is particularly prominent in multi-hop reasoning, as the LLM must retrieve evidence from multiple documents to answer a query (Yang et al., 2018).

Traditionally, this tradeoff has been addressed by improving the quality of the retrieval pipeline to achieve high recall while retrieving fewer documents. In most RAG deployments, the retrieval pipelines uses a top-$K$ embedding-based search between the query and a corpus of passages (Karpukhin et al., 2020). This pipeline is not trained jointly with the LLM because the discrete retrieval and tokenization steps stop gradient flow. Consequently, this post-hoc application can lead to incompatibility between the retrieval process and the LLM's reasoning (Randl et al., 2026). In principle, an end-to-end training signal can utilize adequate data and compute to optimize the performance of the LLM and retrieval system jointly. Previous work on training the retrieval process end-to-end, such as REALM (Guu et al., 2020), Atlas (Izacard et al., 2023), and RETRO



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.

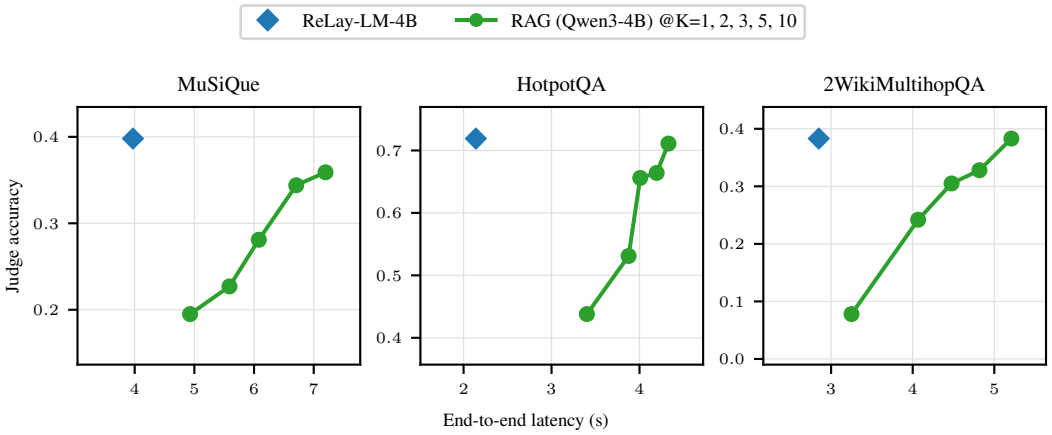


Figure 1: On 3 Multi-hop QA benchmarks that require reasoning over many documents, ReLay-LM-4B is Pareto-dominate to RAG using Qwen3-4B with $K = 1, 2, 3, 5, 10$ from an accuracy against per-query end-to-end latency tradeoff.

(Borgeaud et al., 2022), require jointly pre-training the LLM and encoder model, which is impractical for modern pre-training procedures. A recent attempt, Memory Sparse Attention (MSA) (Chen et al., 2026), uses sparse attention over compressed document latent embeddings to retrieve passages, but still requires prefilling the retrieved passages into the context.

To address the accuracy-efficiency tradeoff and non-differentiability of RAG, we introduce *Retrieval Layers*, sparse attention layers that cross-attend to a KV bank produced by an end-to-end learned encoder. Retrieval Layers offer the following advantages over previous retrieval systems:

- **Fast and memory efficient.** The accompanying encoder can independently pre-compute the KV bank for all the passages of a corpus, eliminating the need to prefill retrieved documents per query. The sparse top-$k$ attention ensures a low inference throughput overhead from attending to documents by taking advantage of fast approximate nearest neighbor search algorithms. Furthermore, we post-train with only a single Retrieval Layer, significantly reducing the required document KV cache size. Overall, Retrieval Layers are much faster and more memory efficient than RAG.
- **End-to-end retrieval training.** The Retrieval Layer encoder can be jointly trained with the LLM, allowing it to directly optimize for better document-grounded QA performance. We also introduce a document-access loss to further encourage strong retrieval through training.
- **Easy integration into existing models.** We propose a relatively minor post-training recipe that preserves the thinking ability of an existing reasoning model. Adding a Retrieval Layer is a small architectural change, which makes it flexible for a variety of model types and applications.

These core benefits allow Retrieval Layers to achieve Pareto-dominance along the accuracy-efficiency tradeoff on question-answering (QA) tasks that require reasoning over many passages. We show that post-training a model with a Retrieval Layer allows it more accurately and efficiently solve complex multi-hop QA benchmarks than using RAG with the same model (Figure 1). We also show this Pareto-dominance in long-context conversation QA using the BEAM benchmark. With our Retrieval Layer, a model can effectively search over ∼240x longer context lengths in needle-in-a-haystack tasks. We call our model that is post-trained with a Retrieval Layer a *Retrieval Layer Language Model (ReLay-LM)*.

## 2 Related Work

Systems that couple LLMs with large corpora face a number of performance considerations. First, the maximum size of a corpus they can handle while preserving performance affects which downstream applications they are useful for. Then, if the system retrieves a subset of information from that corpus that the LLM should selectively attend to, the prefill cost associated with converting

| Family | Large Corpus | Low Prefill Cost | Post-Trained | End-to-End Training | Efficient Attention |
|---|---|---|---|---|---|
| Vanilla RAG | ✓ | ✗ | ✗ | ✗ | ✗ |
| Agentic / Graph-based RAG | ✓ | ✗ | ✓ | ✗ | ✗ |
| REALM / Atlas / RETRO | ✓ | ✓ | ✗ | ✓ | ✗ |
| MSA | ✓ | ✗ | ✓ | ✓ | ✓ |
| Recurrent / Linear models | ✗ | ✓ | ✗ | ✓ | ✓ |
| KV Cache Compression | ✗ | ✗ | ✗ | ✗ | ✓ |
| **Retrieval Layers** | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 1: Comparison of methods that join LLMs with large amounts of external information.

raw text to KV cache directly contributes to per-query end-to-end latency. If the system benefits from end-to-end training to align the generation and retrieval processes, then the efficiency of that training process influences its practicality. Finally, the accuracy and performance of the attention mechanism that selectively extracts the necessary information from KV cache determines how efficiently the system can correctly answer complex queries. In this section, we detail how RAG, end-to-end variants of RAG, and long-context methods treat each of these considerations. Table 1 summarizes the trade-offs between these families.

### 2.1 Retrieval-Augmented Generation

The original RAG formulation aimed to address the difficulty of reliably storing and updating facts in the parameters of an LLM (Lewis et al., 2020). By conducting an embedding-based top-K retrieval over a corpus of millions of passages and populating the LLM's context with the most relevant passages, it became significantly better on open-domain question-answering. However, this process creates a fundamental accuracy-efficiency trade-off. Retrieving more passages increases the probability that the necessary evidence is retrieved, but also increases the amount of quadratic prefill required to populate the context. Improvements in this trade-off have largely targeted increasing positive passage recall under a fixed amount of retrieval.

Dense Passage Retrieval (Karpukhin et al., 2020) introduced the idea of using passage embeddings to retrieve passages that are most similar to a query embedding with maximum inner product search (MIPS). Thus, a promising direction for improving RAG became independently training the best possible passage embedding model. Notable work along this line includes ColBERT (Khattab & Zaharia, 2020) and Contrastive Pre-Training (Neelakantan et al., 2022). Complementarily, re-ranking takes a broad initial set of retrieved passages and uses a specialized re-ranking model to select a smaller, more relevant subset (Nogueira & Cho, 2019).

Multi-hop questions make the accuracy-efficiency trade-off more difficult because multiple, potentially unrelated passages may be needed to answer the query (Yang et al., 2018). Agentic approaches such as IRCoT (Trivedi et al., 2023) and Search-R1 (Jin et al., 2025) alternate reasoning and retrieval so that sequential reasoning can guide later retrieval queries. Graph-based retrieval systems instead organize passage relationships in a graph that can be traversed to retrieve a connected set of passages (Gutiérrez et al., 2024; Han et al., 2025). While these approaches improve recall for multi-hop questions, they also require multiple retrieval rounds, compounding the expensive prefilling cost of RAG. Retrieval Layers are naturally fitted to solve multi-hop questions because at every decode step they can attend to a different sparse token subset of the retrieved passages. By populating the Retrieval Layer's KV bank with a large number of retrieved passages, we get the benefit of high recall without a significant loss in end-to-end inference latency.

### 2.2 End-to-End Learned Retrieval

In most RAG systems, passages are retrieved with a discrete top-$k$ operation and inserted into the LLM's context as discrete tokens. Thus, the LLM's generation loss cannot back-propagate through the passages to jointly train the passage embedding model with the LLM. In this case, finetuning the LLM to improve retrieval quality reduces to optimizing the retrieval query (Jin et al., 2025). A denser training signal can lead to better compatibility between the retrieval process and the LLM's reasoning and allow us to directly optimize for better QA performance.

REALM (Guu et al., 2020) first approached the issue of producing end-to-end learning signal for a retriever embedding by treating the retrieved documents as a latent variable and marginalizing the document-grounded generation loss over many possible retrieved documents. Atlas (Izacard et al., 2023) extends this marginalization technique with an additional encoder that processes documents independently with a Fusion-in-Decoder technique. RETRO (Borgeaud et al., 2022) introduces chunked cross-attention layers that are populated by a end-to-end learned document retriever. All require pretraining the language model with the additional embedding models. Instead, ReLay-LM integrates a Retrieval Layer into an off-the-shelf LLM for a relatively minor amount of post-training, by relying on the LLM's strong reasoning prior to conduct straightforward SFT and RLVR. Furthermore, previous end-to-end methods still require densely attending to the encoded documents, limiting the number of documents they can attend to without sacrificing decoding efficiency.

Memory Sparse Attention (MSA) (Chen et al., 2026) is closely related to Retrieval Layers in its use of an end-to-end learned sparse attention over latent document embeddings. However, its proposed inference procedure gives up the main benefits of latent retrieval. Their Memory Interleave mechanism uses the sparse attention augmented model to generate document identifiers, retrieves the corresponding documents, and appends them to the context with full prefill cost. Even without this additional prefilling step, its latent bank mean-pools each contiguous 64 token-level keys and values, which incurs the disadvantages of lossy compression towards information retrieval. Furthermore, their training recipe consists of a large continued pre-training stage before SFT. Retrieval Layers instead preserve token-level fidelity in their KV banks for fine-grained information retrieval and avoid large prefill costs by only re-using pre-computed document embeddings.

### 2.3 Long Context

Long-context methods aim to improve the ability of LLMs to reason over large amounts of information directly in the context. Yet, these methods remain bounded by their maximum effective context length and require the entire corpus to be prefilled before generation. State-space (Gu & Dao, 2024) and linear-attention models (Katharopoulos et al., 2020) compress long sequences into recurrent states that allow for faster decoding. KV-cache compression methods summarize or prune an already prefilled context (Li et al., 2024; Kim et al., 2025). Retrieval Layers instead avoid full-corpus prefilling, and conduct fast sparse attention over a KV bank that grows independently of the LLM's context window.

## 3 Retrieval Layers

In corpus-grounded QA, a model is given a question $q$ and a document collection $\mathcal{D}$ and must generate an answer $y$ grounded in the relevant documents of $\mathcal{D}$. ReLay-LM does this in two steps. First, a lightweight bidirectional encoder maps every token of every document to a hidden state. These hidden states are concatenated into a key-value (KV) bank that is computed once per corpus. Second, a Retrieval Layer inserted into the LLM sparsely attends to this bank with top-$k$ attention. The result is then added to the residual stream, and the forward pass continues as usual (Section 3.1). We train the model end-to-end in two main stages. The first is supervised fine-tuning (SFT), which teaches the model to read from the bank with the help of a document-access loss that promotes attending to the relevant documents (Section 3.2.2). This is followed by reinforcement learning, which refines retrieval on questions that require combining several documents (Section 3.2.3). Because the document encoder and the LLM are trained jointly, each training example must pair a question and answer with a reasoning trace, the relevant documents, and hard negatives. We construct these training examples by building custom datasets for the SFT and RL stages (Section 3.2.1). We illustrate the Retrieval Layers architecture and SFT data generation & training in Figure 2.

### 3.1 Architecture

**Key-value bank.** Let $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_D\}$ be a document collection, where each document is a collection of tokens $\mathcal{D}_i = \{\mathbf{d}_{i,j}\}_{j=1}^{D_i}$. A bidirectional encoder $E_\phi$ encodes each document independently $\{\mathbf{h}_{i,j}\}_{j=1}^{D_i} = E_\phi(\mathcal{D}_i)$, and every token hidden state is projected into a key $\mathbf{k}_{i,j} = \mathbf{W}_K \mathbf{h}_{i,j}$ and value $\mathbf{v}_{i,j} = \mathbf{W}_V \mathbf{h}_{i,j}$. Collecting these pairs over all tokens and over all documents gives the

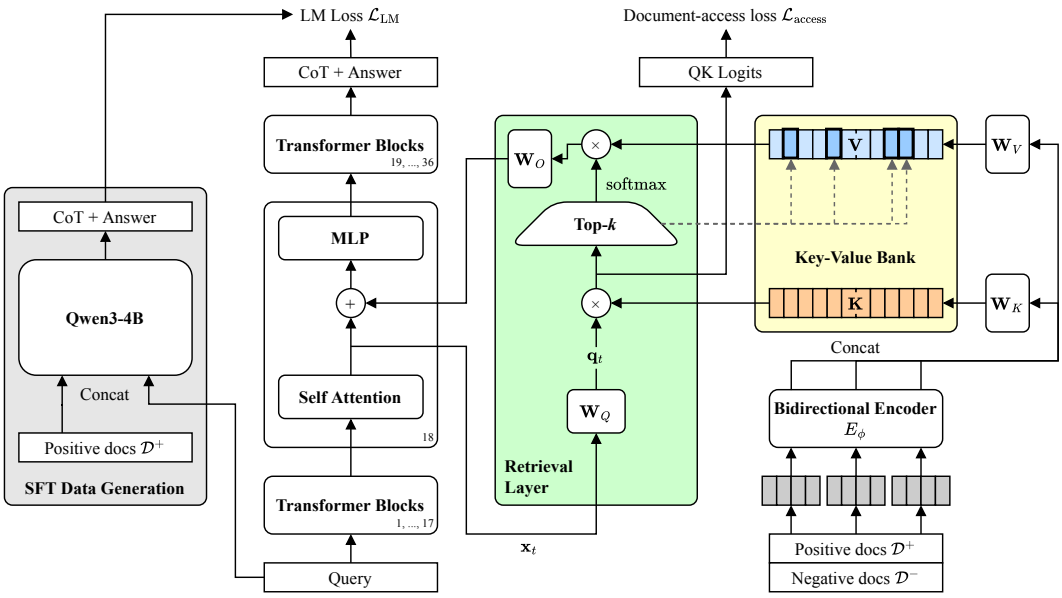


Figure 2: Architecture and SFT framework of Retrieval Layer Language Models. (Left) To generate CoT traces and answers for SFT, we concatenate and input positive documents $\mathcal{D}^+$ and queries to a frozen Qwen3-4B. (Right) We insert the Retrieval Layer after Qwen3-4B layer 18 self-attention. The bidirectional encoder encodes each document independently into key and value tokens stored in the key-value (KV) bank. The Retrieval Layer then reads from the KV bank through multi-query top-$k$ attention via a fast approximate top-$k$ search algorithm (Chern et al., 2022), with the result added to the residual stream. During training, we put both positive and negative documents $\mathcal{D}^+ \cup \mathcal{D}^-$ into the memory bank to train RelayLM to retrieve only the relevant (positive) documents.

key-value (KV) bank

$$\mathcal{B} = \bigcup_{i=1}^{D} \{(\mathbf{k}_{i,j},\, \mathbf{v}_{i,j})\}_{j=1}^{D_i} = \{(\mathbf{k}_j, \mathbf{v}_j)\}_{j=1}^{M}, \tag{1}$$

where $M$ is the total number of tokens over all documents. We encode each document with its own positional embedding and not with corpus-wide positional embeddings, as doing so removes the long-context degradation of RoPE (Chen et al., 2023) when attending over the whole bank. Furthermore, in cases where only a subset of the corpus is used per query, the documents do not need to be re-encoded, thus saving end-to-end latency.

**Retrieval Layer.** We insert the Retrieval Layer after the $\ell$-th self-attention layer. Given the residual state $\mathbf{x}_t$ at position $t$, the Retrieval Layer reads from the KV bank through multi-query top-$k$ attention. First, we compute multi-query cross-attention logits between queries $\mathbf{q}_{t,n}$ projected from $\mathbf{x}_t$ and each document key vector $\mathbf{k}_j$:

$$\mathbf{q}_{t,n} = \mathrm{RMSNorm}\big(\mathbf{W}_{Q,n}\,\mathrm{RMSNorm}(\mathbf{x}_t)\big) \tag{2}$$

$$a_{t,n,j} = \mathbf{q}_{t,n}^{\top}\,\mathrm{RMSNorm}(\mathbf{k}_j)\,/\sqrt{d_k} \tag{3}$$

Then, we extract the indices of the $k$ best-scoring KV bank entries and compute the weighted sum of their values per head

$$\mathcal{I}_{t,n} = \underset{j}{\mathrm{TopK}}\ a_{t,n,j} \tag{4}$$

$$\mathbf{r}_{t,n} = \textstyle\sum_{j\in\mathcal{I}_{t,n}} \mathrm{softmax}_{j\in\mathcal{I}_{t,n}}\big(a_{t,n,j}\big)\,\mathbf{v}_j. \tag{5}$$

Lastly, we merge the weighted sums for each head and sum the result into the residual stream

$$\mathbf{x}_t \leftarrow \mathbf{x}_t + \mathbf{W}_O\,[\mathbf{r}_{t,1}; \ldots; \mathbf{r}_{t,N}]. \tag{6}$$

The remaining layers then run unchanged. In our implementation, for both training and inference, Equations 3 and 4 are combined with TPU-KNN (Chern et al., 2022) for fast approximate top-$k$ search, but any approximate nearest neighbor algorithm also suffices.

### 3.2 Training Recipe

#### 3.2.1 Data

Training the encoder and the LLM end-to-end places two demands on every sample. First, finding the most relevant subset of bank tokens among potentially similar-looking text, and second, reasoning about the retrieved information to answer complex queries. For retrieval, it is common practice to introduce *hard* negative distractors that are very semantically similar to the positive document (Xiong et al., 2021). Each training sample is therefore a tuple $(q, \mathcal{D}^+, \mathcal{D}^-, c, y)$: a question, its positive documents, hard negatives, a reasoning trace, and an answer.

We collect questions, positive documents, and answers from seven existing sources. The largest is the document-grounded QA data of Nemotron (NVIDIA et al., 2025), from which we filter for the Science-QA and Diverse-QA splits. We add the training splits of MS-MARCO (Bajaj et al., 2018), SQuAD (Rajpurkar et al., 2016), Natural Questions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), and TriviaQA (Joshi et al., 2017). In total, we use 780M tokens of CoT supervision and 5B tokens of documents. To elicit the capability of chaining evidence across many documents, we also synthesize multi-hop questions. We sample relation chains as random walks over the DBpedia 2022 Wikipedia knowledge graph and attach to each hop a Wikipedia paragraph that states it. An LLM then phrases the chain as a question and gives its answer. We include additional details of our SFT dataset in A.1.

We construct the remaining two elements of a sample ourselves. Hard negatives should be documents that a retriever would plausibly confuse with the evidence, so for every question we retrieve the top-$k$ documents with Qwen3-Embedding-0.6B (Zhang et al., 2025), re-score them with Qwen3-Reranker-4B, and keep those ranked highly as $\mathcal{D}^-$. Since we aim to post-train an existing LLM that is already a strong reasoning model, we must run SFT on reasoning traces that preserve the same quality and style of reasoning. These reasoning traces must also illicit the behavior of extracting important information from the positive document and reasoning about the information to answer the question. Thus, for each $(q, \mathcal{D}^+, y)$ triple, we take the same LLM we plan on post-training and prompt it with the query and positive documents. After verifying that the predicted answer matches the gold answer $y$, we extract the Chain-of-Thought (Wei et al., 2022) that produced the answer and save it as $c$. For the reinforcement learning stage, we use the training splits of multi-hop datasets HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), and 2WikiMultihopQA (Ho et al., 2020), whose short answers can be accurately verified by an LLM judge.

#### 3.2.2 Loss

We train with cross entropy language-modeling loss $\mathcal{L}_{\text{LM}}$ plus a document-access loss,

$$\mathcal{L} = \lambda_{\text{LM}}\,\mathcal{L}_{\text{LM}} + \lambda_{\text{acc}}\,\mathcal{L}_{\text{access}}. \tag{7}$$

$\mathcal{L}_{\text{LM}}$ alone is a weak signal for information retrieval since gradients only flow through the top-$k$ selected tokens. We take advantage of labeled positive documents to provide supervision over all the bank tokens simultaneously using a contrastive document-access loss inspired by InfoNCE (van den Oord et al., 2019) .

Consider a batch of $B$ QA samples. Sample $b$ comes with labeled relevant (positive) documents $\mathcal{D}_b^+$ and curated hard negatives $\mathcal{D}_b^-$ (Section 3.2.1). The KV bank for the batch contains the tokens of all these documents,

$$\mathcal{B} \;=\; \bigcup_{b=1}^{B} E_\phi(\mathcal{D}_b^+) \cup E_\phi(\mathcal{D}_b^-). \tag{8}$$

For sample $b$, the positive tokens are $\mathcal{D}_b^+$, while every other slot in $\mathcal{B}$ is a negative. This includes the sample's own hard negatives and every document of the other $B-1$ samples as in-batch negatives.

Let $a_{b,t,n,j}$ be the score between the query of head $n$ at generation position $t$ of sample $b$ and bank token $j$, taken before top-$k$ selection, and let $\mathcal{R}_b$ be the non-masked generation positions. The probability that this query reads a positive slot is

$$p^+_{b,t,n} \;=\; \frac{\sum_{j\in\mathcal{D}_b^+}\exp\big(a_{b,t,n,j}\big)}{\sum_{j\in\mathcal{B}}\exp\big(a_{b,t,n,j}\big)}, \tag{9}$$

and the loss is its negative log-likelihood averaged over heads and response tokens,

$$\mathcal{L}_{\text{access}} = -\frac{1}{N \cdot \sum_b |\mathcal{R}_b|} \sum_{b=1}^{B} \sum_{t \in \mathcal{R}_b} \sum_{n=1}^{N} \log p^{+}_{b,t,n}. \tag{10}$$

#### 3.2.3 TRAINING STAGES

The LLM and encoder are initialized from existing trained models, while the Retrieval Layer that joins them is initialized from scratch. Training all the components at once would let noisy gradients from the new Retrieval Layer disturb both pretrained models. Following the staged recipe of LLaVA (Liu et al., 2023), we introduce trainable components and learning signals one at a time during SFT. For the first 5k steps, we train only the Retrieval Layer (its projections and norms) with $\mathcal{L}_{\text{access}}$ ($\lambda_{\text{LM}} = 0$), so that queries learn to find the relevant documents. From 5k to 10k steps, we also unfreeze the encoder, so that document representations adapt to the lookup. From 10k to 15k steps, we turn on $\mathcal{L}_{\text{LM}}$ ($\lambda_{\text{LM}} = 1$), so that the values read from the bank become useful for generation. After 15k steps, we unfreeze the LLM and train all components jointly. $\mathcal{L}_{\text{access}}$ remains active throughout supervised fine-tuning with $\lambda_{\text{acc}} = 0.1$.

This is followed by reinforcement learning with verifiable rewards (Shao et al., 2024), which refines both the model's reasoning traces and what it retrieves while producing them. Starting from the supervised checkpoint, we optimize with DAPO (Yu et al., 2025) on the multi-hop training data (Section 3.2.1). The binary reward is produced by an LLM judge that only compares the final answer with the ground truth. We turn off $\mathcal{L}_{\text{access}}$ in this stage, so retrieval is shaped only by whether it leads to the correct answer.

## 4 RESULTS

### 4.1 EXPERIMENTAL SETUP

To evaluate the efficacy of ReLay-LM, we evaluate on challenging multi-hop question-answering (QA) tasks and long-context tasks such as needle-in-a-haystack and conversation history recall.

**Benchmarks.** For multi-hop QA, we use HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). These benchmarks require traversing multiple documents from large corpuses (up to 1.5M tokens) in a reasoning chain to obtain an answer. We report LLM judge accuracy against end-to-end latency. For long-context tasks we use BEAM (Tavakoli et al., 2026) and RULER (Hsieh et al., 2024). BEAM tests factual recall over long, multi-turn conversations, which is an important skill for reliable agentic systems. Using the 128K length split of the benchmark, we report LLM judge accuracy and per-query end-to-end latency on the information extraction questions. RULER is a synthetic needle-in-a-haystack benchmark that evaluates the ability of the model to retrieve short pieces of information from long distractor text. We report accuracy on the single needle tasks.

**Implementation Details.** We initialize the post-training recipe for ReLay-LM with Qwen3-4B (Yang et al., 2025) for the decoder and Qwen3-Embedding-0.6B (Zhang et al., 2025) for the embedding model. We insert the Retrieval Layer in the exact depth-wise center block of Qwen3-4B, in between the self-attention and MLP layers. The Retrieval Layer uses 4 query heads, 1,024-dimensional keys and values, and for top-$k$, $k = 64$.

In the SFT stage, we train for 100k steps total at batch size 16. The base learning rate (LR) for the Retrieval Layer is 1e-4 and for the LLM and encoder is 1e-5. For the first 3 stages, there is a 500 step linear warm-up from 1e-8 to the base LR. For the final stage where everything is unfrozen, there is an 8,000 step linear warm-up to the base LR, followed by a cosine decay to 0. The optimizer is AdamW. We use a sequence length of 1024 that supervises the CoT and final generated answer. For each sample in the batch, we build its document bank starting with positive documents followed by its hard negatives. We limit the number of document tokens to 4096, which is enough for the positive documents to always be present per sample. For the RL stage, we use the default DAPO (Yu et al., 2025) hyperparameters from the Tunix post-training library (Bao et al., 2025). We train for 600

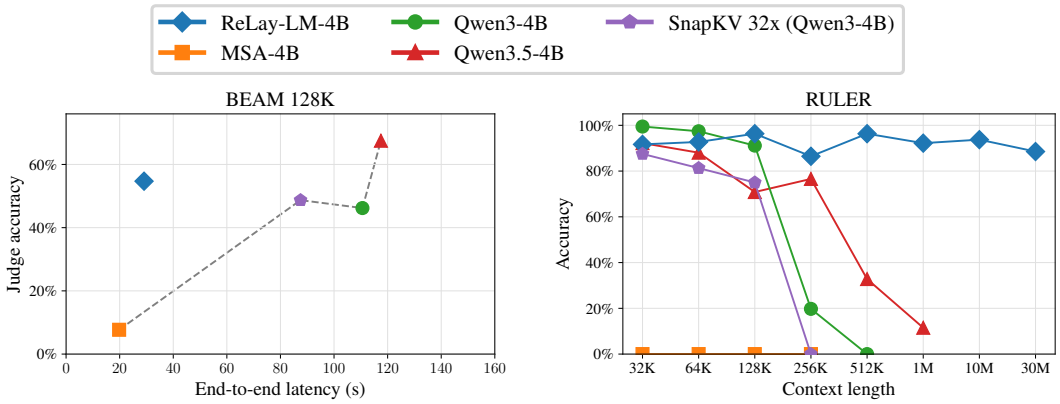


Figure 3: Results on long-context tasks. (Left) On BEAM 128K Tavakoli et al. (2026), ReLay-LM performs substantially better than the Pareto frontier in terms of an accuracy against end-to-end latency tradeoff. (Right) On RULER Hsieh et al. (2024), ReLay-LM is the only method which maintains high accuracy at very long contexts.

steps of batch size 16, group size 8, maximum rollout length of 2048 tokens, and with 50 documents in the Retrieval Layer. We use a 50 step linear warm-up to a base LR of 3e-6.

For Multi-hop QA, we demonstrate the ability for ReLay-LM to efficiently reason over a large number of documents by retrieving 50 documents with the same retriever as RAG and populate the Retrieval Layer bank with the KV of those documents. Since we can pre-compute KV for all documents, the encoding step does not factor into per-query end-to-end latency. For BEAM and RULER, the full supporting context is encoded and populated into the Retrieval Layer per query. To speed up the encoding step, we split documents into chunks of 256 tokens to do batched encoding.

For SFT we used a TPU v6-8 node and for RL we used a TPU v6-16 node. Evals are conducted on a TPU v6-4 node which has a total VRAM of 128 GB.

**Baselines.** For the multi-hop QA benchmarks, we compare ReLay-LM-4B against vanilla RAG that uses Qwen3-4B (Yang et al., 2025) as the decoder and Qwen3-Embedding-0.6B (Zhang et al., 2025) as the retriever, matching the initialization of ReLay-LM-4B. The RAG baseline splits every document in the corpus into 256-token chunks and places the top-$K$ retrieved chunks into the decoder's prompt. We focus RAG on the regime where $K$ varies between 1 and 10 (RAG@$K$).

On both long-context benchmarks, we compare against two models that read the entire corpus from the context. The first is our base model, Qwen3-4B, with its context window extended beyond the native 32K tokens by YaRN (Peng et al., 2024) (scaling factor 6 on BEAM, which covers its longest 184K-token conversation). We also compare to a comparable size Qwen3.5-4B from the linear attention family in which 24 of 32 layers use Gated DeltaNet linear attention. For a KV-cache compression method, we implemented SnapKV (Li et al., 2024) on Qwen3-4B, where it prefills the same context once, scores each cached position by the attention it receives, and evicts irrelevant cache to achieve a x32 compression rate.

We also compare against Memory Sparse Attention (MSA-4B) (Chen et al., 2026) on BEAM and RULER. Although MSA is trained to natively retrieve document information in latent space, they still rely on a RAG-style procedure that appends retrieved documents to the context. In order to compare ReLay-LM with another sparse-attention based retrieval method, we evaluate MSA without the prefilling procedure, "Memory Interleave". We evaluate using the provided MSA-4B checkpoint that includes memory sparse attention layers at all blocks of a Qwen3-4B-Instruct model and a 64x token compression rate over the memory chunks.

### 4.2 Multi-hop Question Answering

In Figure 1 we show that our ReLay-LM-4B Pareto-dominates RAG across a sweep of retrieved documents on the accuracy against per-query end-to-end latency tradeoff. Across the three benchmarks,

ReLay-LM is on average $1.89\times$ faster on end-to-end latency than RAG@10 with accuracies matching or better than RAG@10 despite retrieving $50$ documents. This shows that not only is ReLay-LM able to accurately retrieve and reason over a large number of documents, but is able to do so exceptionally efficiently. Furthermore, in Table 3 we see that our RL stage provided gains of up to 18.7 % in accuracy compared to just the SFT stage. This highlights another benefit of ReLay-LM being end-to-end trainable: we can benefit from the full modern post-training stack.

### 4.3 Long-Context Tasks

**BEAM.** In Figure 3 (left) we show that our ReLay-LM-4B improves upon the Pareto-frontier of baselines on the BEAM benchmark. Compared to the Qwen3-4B that it was initialized with, it achieves $\sim$3.8x faster latency while being $\sim$8% more accurate. This balance of accuracy and efficiency is critical for scenarios such as long-horizon agents.

**RULER.** Figure 3 (right) reports mean RULER single-needle accuracy tasks (SA1-SA3) across context lengths. The base model, Qwen3-4B, remains competitive within its pre-trained dense attention context length of 128k, while it begins to significantly decay from 91% to $\tilde{2}0\%$ and eventually 0% above 128k tokens. ReLay-LM remains competitive throughout the context length sweep, only declining to 88.5% until 30M tokens, which is more than $\sim$240x the maximum effective context length of its initialization. Other models such as Qwen3.5-4B declines to 32.8% at 512K, while SnapKV declines to 33% on merely 256k tokens. From Table 5, we discovered that SnapKV consistently fails to retrieve on the S3 tasks where the needle is a lengthy UUID. We hypothesize this occurs because an eviction of a single KV pair of the UUID could result in a failed retrieval. Moreover, MSA consistently scores 0% over all context lengths. This is likely due to the 64x mean-pooling of the KV bank that completely prevents the model from retrieving precise needles. Overall, we find that ReLay-LM can retrieve precise facts over extremely large contexts, highlighting its potentially to efficiently reason over massive corpora.

## 5 Conclusion

In this work, we considered the question of how to achieve an end-to-end retrieval system that can not only be integrating into modern reasoning models, but also strong QA performance in an efficiency manner. We introduced Retrieval Layers as a sparse top-$k$ attention with a learned encoder that can be end-to-end trained to retrieve relevant information from a large set of documents. With a specialized dataset and post-training recipe that aims to preserve reasoning of an LLM, we train a model called ReLay-LM. We show Pareto-dominance on complex multi-hop QA compared to RAG, demonstrating that with adequate end-to-end training, we can greatly improve retrieval ability. In a similar light, we showed strong accuracy-efficiency results on user conversation recall, and discovered that this learned sparse attention approach can significantly increase the effective context length of a model. As retrieval from large corpora become increasingly vital for industrial and agentic applications, Retrieval Layers can efficiently and effectively support their growth.

### AI use statement

In this work, we used generative AI (LLMs) to generate synthetic datasets, particularly the CoT traces for our SFT data. We also used generative AI tools to help with implementing our method and running experiments and baselines. We *have not* used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, or interpreting results, and helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and assisting with translation are not applicable to this work. We have reviewed all AI-assisted work. We spot-checked our synthetic datasets to make sure that the CoT traces are sensible and the answers reasonable, and we read and verified the correctness of our AI-assisted implementations of our method, experiments, and baseline scripts. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

## REFERENCES

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, 2018. URL https://arxiv.org/abs/1611.09268.

Tianshu Bao, Jeff Carpenter, Lin Chai, Haoyu Gao, Yangmu Jiang, Shadi Noghabi, Abheesht Sharma, Sizhi Tan, Lance Wang, Ann Yan, Weiren Yu, et al. Tunix (tune-in-jax). https://github.com/google/tunix, 2025.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023. URL https://arxiv.org/abs/2306.15595.

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, 2026. URL https://arxiv.org/abs/2603.23516.

Felix Chern, Blake Hechtman, Andy Davis, Ruiqi Guo, David Majnemer, and Sanjiv Kumar. TPU-KNN: K nearest neighbor search at peak FLOP/s. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=OoNmOfYVhEU.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=hkujvAPVsg.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. Retrieval-augmented generation with graphs (graphrag), 2025. URL https://arxiv.org/abs/2501.00309.

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/.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

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. URL http://jmlr.org/papers/v24/23-0037.html.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

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 Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 39–48, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=JFygzwx8SJ.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In

H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916. Curran Associates, Inc., 2023. doi: 10.52202/075280-1516. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. Text and code embeddings by contrastive pre-training, 2022. URL https://arxiv.org/abs/2201.10005.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019. URL http://arxiv.org/abs/1901.04085.

NVIDIA, :, Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, Alex Kondratenko, Alex Shaposhnikov, Alexander Bukharin, Ali Taghibakhshi, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amy Shen, Andrew Tao, Ann Guan, Anna Shors, Anubhav Mandarwal, Arham Mehta, Arun Venkatesan, Ashton Sharabiani, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Banghua Zhu, Barnaby Simkin, Bilal Kartal, Bita Darvish Rouhani, Bobby Chen, Boris Ginsburg, Brandon Norick, Brian Yu, Bryan Catanzaro, Charles Wang, Charlie Truong, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christian Munley, Christopher Parisien, Dan Su, Daniel Afrimi, Daniel Korzekwa, Daniel Rohrer, Daria Gitman, David Mosallanezhad, Deepak Narayanan, Dima Rekesh, Dina Yared, Dmytro Pykhtar, Dong Ahn, Duncan Riach, Eileen Long, Elliott Ning, Eric Chung, Erick Galinkin, Evelina Bakhturina, Gargi Prasad, Gerald Shen, Haifeng Qian, Haim Elisha, Harsh Sharma, Hayley Ross, Helen Ngo, Herman Sahota, Hexin Wang, Hoo Chang Shin, Hua Huang, Iain Cunningham, Igor Gitman, Ivan Moshkov, Jaehun Jung, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jian Zhang, Jiaqi Zeng, Jimmy Zhang, Jinze Xue, Jocelyn Huang, Joey Conway, John Kamalu, Jonathan Cohen, Joseph Jennings, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kari Briski, Katherine Cheung, Katherine Luna, Keith Wyss, Keshav Santhanam, Kezhi Kong, Krzysztof Pawelec, Kumar Anik, Kunlun Li, Kushan Ahmadian, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Luis Vega, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Mark Cai, Markus Kliegl, Marta Stepniewska-Dziubinska, Matvei Novikov, Mehrzad Samadi, Meredith Price, Meriem Boubdir, Michael Boone, Michael Evans, Michal Bien, Michal Zawalski, Miguel Martinez, Mike Chrzanowski, Mohammad Shoeybi, Mostofa Patwary, Namit Dhameja, Nave Assaf, Negar Habibi, Nidhi Bhatia, Nikki Pope, Nima Tajbakhsh, Nirmal Kumar Juluru, Oleg Rybakov, Oleksii Hrinchuk, Oleksii Kuchaiev, Oluwatobi Olabiyi, Pablo Ribalta, Padmavathy Subramanian, Parth Chadha, Pavlo Molchanov, Peter Dykas, Peter Jin, Piotr Bialecki, Piotr Januszewski, Pradeep Thalasta, Prashant Gaikwad, Prasoon Varshney, Pritam Gundecha, Przemek Tredak, Rabeeh Karimi Mahabadi, Rajen Patel, Ran El-Yaniv, Ranjit Rajan, Ria Cheruvu, Rima Shahbazyan, Ritika Borkar, Ritu Gala, Roger Waleffe, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Sahil Jain, Samuel Kriman, Sanjeev Satheesh, Saori Kaji, Sarah Yurick, Saurav Muralidharan, Sean Narenthiran, Seonmyeong Bak, Sepehr Sameni, Seungju Han, Shanmugam Ramasamy, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shizhe Diao,

Shreya Gopal, Shrimai Prabhumoye, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Siddhartha Jain, Somshubra Majumdar, Soumye Singhal, Stefania Alborghetti, Syeda Nahida Akter, Terry Kong, Tim Moon, Tomasz Hliwiak, Tomer Asida, Tony Wang, Tugrul Konuk, Twinkle Vashishth, Tyler Poon, Udi Karpas, Vahid Noroozi, Venkat Srinivasan, Vijay Korthikanti, Vikram Fugro, Vineeth Kalluru, Vitaly Kurin, Vitaly Lavrukhin, Wasi Uddin Ahmad, Wei Du, Wonmin Byeon, Ximing Lu, Xin Dong, Yashaswi Karnati, Yejin Choi, Yian Zhang, Ying Lin, Yonggan Fu, Yoshi Suhara, Zhen Dong, Zhiyu Li, Zhongbo Zhu, and Zijia Chen. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. URL https://doi.org/10.48550/arXiv.2310.08560.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Korbinian Randl, Guido Rocchietti, Aron Henriksson, Ziawasch Abedjan, Tony Lindgren, and John Pavlopoulos. Quantifying retriever-generator alignment in rag with local explanations, 2026. URL https://arxiv.org/abs/2601.21803.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Chihiro Taguchi, Seiji Maekawa, and Nikita Bhutani. Efficient context selection for long-context QA: No tuning, no iteration, just adaptive-$k$. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20105–20130, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1017. URL https://aclanthology.org/2025.emnlp-main.1017/.

Mohammad Tavakoli, Alireza Salemi, Carrie Ye, Mohamed Abdalla, Hamed Zamani, and J Ross Mitchell. Beyond a million tokens: Benchmarking and enhancing long-term memory in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=y59hf5lrMn.

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. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zeFrfgyZln.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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 Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

# A Appendix

## A.1 SFT Data Mixture

Table 2 details the data mixture we use for the SFT stage of post-training.

Table 2: SFT training data composition.

| Source | Rows | CoT+Ans total | Doc. total | Total |
|---|---|---|---|---|
| Science Document QA | 400,000 | 215.97M | 1.94B | 2.14B |
| Diverse Document QA (hard-neg, think) | 400,000 | 204.56M | 2.29 B | 2.49 B |
| Wikipedia Multihop QA (custom) | 400,000 | 216.80M | 599.60M | 816.40M |
| MS-MARCO | 240,257 | 115.56M | 84.33M | 199.89M |
| TriviaQA | 28,932 | 9.20M | 89.92M | 99.12M |
| Natural Questions | 47,457 | 11.44M | 27.29M | 38.72M |
| SQuAD | 41,862 | 17.04M | 35.75M | 52.79M |
| HotpotQA | 41,492 | 10.04M | 23.28M | 33.32M |
| **Total** | **1,600,000** | **780.08M** | **5.08 B** | **5.86 B** |

## A.2 Additional Experiment Results

Table 3: Multi-hop QA results.

| Method | Acc. (%) | Latency (s) |
|---|---|---|
| *HotpotQA* | | |
| RAG@1 | 43.8 | 3.40 |
| RAG@2 | 53.1 | 3.88 |
| RAG@3 | 65.6 | 4.01 |
| RAG@5 | 66.4 | 4.20 |
| RAG@10 | 71.1 | 4.33 |
| ReLay-LM (Pre-RL) | 57.8 | 4.36 |
| ReLay-LM | **71.9** | 2.14 |
| *MuSiQue* | | |
| RAG@1 | 19.5 | 4.93 |
| RAG@2 | 22.7 | 5.59 |
| RAG@3 | 28.1 | 6.08 |
| RAG@5 | 34.4 | 6.71 |
| RAG@10 | 35.9 | 7.20 |
| ReLay-LM (Pre-RL) | 21.1 | 4.72 |
| ReLay-LM | **39.8** | 3.97 |
| *2WikiMultihopQA* | | |
| RAG@1 | 7.8 | 3.25 |
| RAG@2 | 24.2 | 4.07 |
| RAG@3 | 30.5 | 4.48 |
| RAG@5 | 32.8 | 4.82 |
| RAG@10 | 38.3 | 5.21 |
| ReLay-LM (Pre-RL) | 33.6 | 3.47 |
| ReLay-LM | **38.3** | 2.85 |

Table 4: BEAM (128K) information-extraction questions results.

| Method | Acc. (%) | Latency (s) |
|---|---|---|
| **ReLay-LM** | 54.7 | 29.1 |
| Qwen3-4B (YaRN, full context) | 46.2 | 110.6 |
| Qwen3.5-4B (full context) | 67.5 | 117.5 |
| SnapKV (Qwen3-4B) | 48.7 | 87.5 |
| MSA-4B | 7.7 | 19.8 |

Table 5: RULER needle-in-a-haystack results.

| | ReLay-LM | | | Qwen3.5-4B | | | Qwen3-4B | | | SnapKV | | | MSA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Context | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| 32K | 0.984 | 1.000 | 0.766 | 0.891 | 0.938 | 0.938 | 0.984 | 1.000 | 1.000 | 1.000 | 0.875 | 0.000 | 0.000 | 0.000 | 0.000 |
| 64K | 0.969 | 0.844 | 0.969 | 0.922 | 0.875 | 0.844 | 0.953 | 1.000 | 0.969 | 1.000 | 0.813 | 0.000 | 0.000 | 0.000 | 0.000 |
| 128K | 0.984 | 0.906 | 1.000 | 0.594 | 0.734 | 0.797 | 0.766 | 0.984 | 0.984 | 1.000 | 0.750 | 0.000 | 0.000 | 0.000 | 0.000 |
| 256K | 0.875 | 0.969 | 0.750 | 0.594 | 0.891 | 0.812 | 0.031 | 0.531 | 0.031 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 512K | 0.984 | 0.906 | 1.000 | 0.203 | 0.406 | 0.375 | 0.000 | 0.000 | 0.000 | OOM | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 1M | 0.953 | 0.938 | 0.875 | 0.062 | 0.266 | 0.017 | OOM | OOM | OOM | OOM | OOM | OOM | – | – | – |
| 10M | 1.000 | 0.953 | 0.859 | – | – | – | – | – | – | – | – | – | – | – | – |
| 30M | 0.906 | 0.906 | 0.844 | – | – | – | – | – | – | – | – | – | – | – | – |