A Unified Model and Document Representation for On-Device Retrieval-Augmented Generation

Julian Killingback* **& Hamed Zamani**
University of Massachusetts Amherst
{jkillingback,zamani}@cs.umass.edu

Ofer Meshi, Henry Li, & Maryam Karimzadehgan
Google
{meshi,henryhl,maryamk}@google.com

Abstract

Traditional Retrieval-Augmented Generation (RAG) approaches generally assume that retrieval and generation occur on powerful servers removed from the end user. While this reduces local hardware constraints, it introduces significant drawbacks: privacy concerns regarding data access, recurring maintenance and storage costs, increased latency, and the necessity of an internet connection. On-device RAG addresses these challenges by executing the entire pipeline locally, making it ideal for querying sensitive personal information such as financial documents, contact details, and medical history. However, on-device deployment necessitates a delicate balance between limited memory and disk space. Specifically, the context size provided to the generative model must be restricted to manage KV cache and attention memory usage, while the size of stored embeddings must be minimized to preserve disk space. In this work, we propose a unified model that compresses the RAG context and utilizes the same representations for retrieval. This approach minimizes disk utilization compared to using separate representations, while significantly reducing the context size required for generation. Using, on average, one-tenth of the context, our model matches the performance of a traditional RAG reader without increasing storage requirements compared to a multi-vector retrieval model. This approach represents the first model to unify retrieval and context compression using a shared model and representation. We believe this work will inspire further consolidation of distinct models to optimize on-device performance.

1 Introduction

Retrieval-Augmented Generation (RAG) has become an indispensable part of many Artificial Intelligence (AI) workflows (Lewis et al., 2020; Zamani et al., 2022). RAG augments an original request with additional context retrieved from an information corpus, providing generative models with the external knowledge necessary to satisfy complex queries. Existing RAG systems are typically deployed on dedicated, resource-rich servers for shared tasks, sometimes requiring retrieval from datasets containing trillions of tokens (Borgeaud et al., 2022). While effective, this paradigm has notable drawbacks. First, it requires server access to the information corpus, raising privacy concerns for sensitive data like personal text messages, financial documents, or medical records; even when the corpus is not private, user requests may still be sensitive, furthering the need for privacy preserving RAG. Second, server-based RAG relies on a constant internet connection, introducing network latency and rendering it unreliable in areas with poor coverage. Third, for personalized use cases, the

*Work done while at Google.

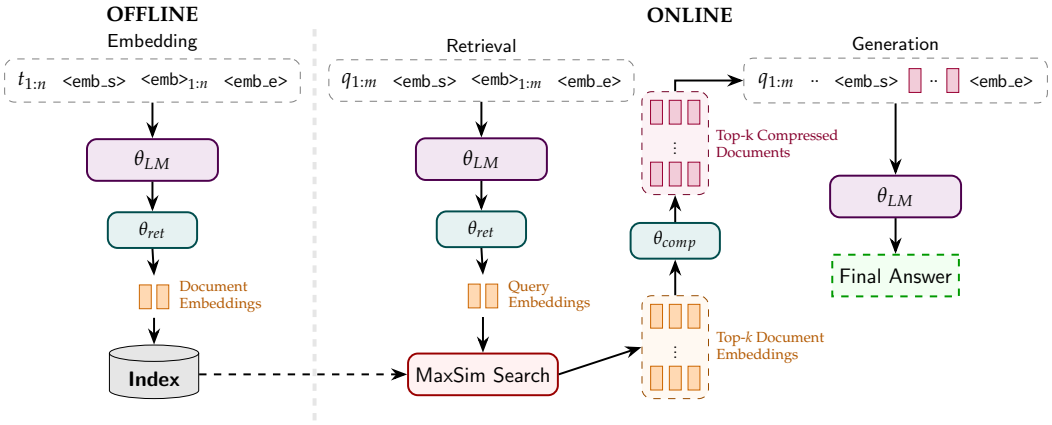


Figure 1: Inference usage of the ECG (Embed, Compress, Generate) models.

cost of constructing and storing user-specific retrieval indices cannot be amortized across many users, increasing service costs and limiting accessibility.

On-device RAG addresses these issues by executing the entire pipeline locally, ensuring strict privacy, offline availability, and zero ongoing server costs. However, user devices pose strict compute and memory constraints, necessitating smaller models and minimized context sizes. To manage context limits, corpus documents can be compressed into a smaller number of vector representations (Li et al., 2025; Rau et al., 2024; Cheng et al., 2024). While creating and storing these representations requires upfront compute and disk space, this preprocessing can be opportunistically scheduled (e.g., while the device is charging), ultimately reducing the computational burden during latency-sensitive user queries.

In this work, we present a unified model for on-device RAG that performs three distinct roles: encoding information for retrieval, compressing text into vector representations, and generating responses based on those compressed contexts. Crucially, we unify the representations themselves, using the same vectors for both retrieval and generation. This eliminates the need to encode and store separate retrieval and compression representations, effectively halving the preprocessing compute, storage requirements, and memory overhead. Compared to standard RAG with a late-interaction retrieval model, our approach uses slightly less disk space while significantly outperforming standard RAG under constrained context budgets and matching standard reader performance with up to a 16$\times$ compression in document representations. Compared to standard compress-and-generate RAG setups, our approach uses half the storage and significantly outperforms the context compression reader with the same context budget.

As outlined, this unification brings significant benefits, but implementing such a system is non-trivial. One approach might be to use a model and representation trained for retrieval or compression and adapt them to the other task, but this can result in catastrophic forgetting of the original task. This can be remedied by freezing the representations and using separate models, but then the representations are not optimized for one of the , which impacts performance. In order to get the best results, all components must be trained jointly. This is difficult, as our ablation results show. We find it requires learned scaling for the retrieval labels to ensure the retrieval information does not overwhelm the compression information. Additionally, the training data and objective must be carefully constructed to maximize performance. We hope that our careful experimentation shows that although difficult, the unification of these skills is possible and comes with substantial benefits for on-device RAG.

Beyond on-device RAG, our strategy of unifying multiple functional roles and representations into a single model provides a blueprint for highly capable, efficient on-device "everything-models." By minimizing architectural redundancy and representation storage, we hope this work demonstrates the viability of consolidated models and expedites the development of advanced on-device AI.

2 Related work

Context compression. Context compression reduces memory and compute usage during generation and falls into two categories: hard and soft. **Hard compression** filters or rewrites text to remove irrelevant tokens. Methods like Selective Context (Li et al., 2023b) and LLMLingua (Jiang et al., 2023) use model probabilities to filter tokens, though this can introduce grammatical errors. Generative rewriting approaches like Nano-Capsulator (Chuang et al., 2024) and RECOMP (Xu et al., 2024) avoid syntax issues but require expensive generation passes. Generally, hard compression is constrained by natural language bounds and often relies on unstable reinforcement learning (Pan et al., 2024; Shandilya et al., 2025; Jung & Kim, 2024). **Soft compression** encodes context into continuous vectors. Early methods learned soft prompts (Wingate et al., 2022) or "gist" tokens (Mu et al., 2023). Recent approaches like ICAE (Ge et al., 2024), 500xCompressor (Li et al., 2025), COCOM (Rau et al., 2024), PISCO (Louis et al., 2025), and ACC-RAG (Guo & Ren, 2025) use various approaches such as auto-encoding or cross-attention to compress longer contexts for LLMs. Our approach resembles COCOM and ACC-RAG, but crucially the compressed context representations are also meaningful as retrieval embeddings. This eliminates the need for a separate retrieval model, allows for end-to-end training, and supports dynamic truncation.

Unified retrieval and generation. Prior work largely combines only subsets of RAG tasks. Several approaches train a single model for both embedding and generation (Muennighoff et al., 2025; Zhang et al., 2025; 2024), whereas xRAG (Cheng et al., 2024) uses fixed retrieval embeddings as context for generation but maintains separate retrieval and generation models. Unlike these methods, our model unifies retrieval, compression, and generation end-to-end using shared multi-vector representations. While the concurrent work CLaRa (He et al., 2026) also unifies context and retrieval representations, its evaluation is limited to reranking settings, leaving its effectiveness for full-scale corpus retrieval unknown. Our approach is explicitly designed for the end-to-end demands of on-device RAG and offers several distinct architectural advantages. First, unlike CLaRa's fixed-length requirement, our model uses variable-length multi-vector embeddings to flexibly control retrieval and storage costs. Second, it minimizes overall disk footprint by employing a single unified model. Finally, rather than relying on frozen document vectors trained solely for compression, our method jointly trains representations for both retrieval and generation contexts.

On-device LLMs and RAG. Deploying LLMs locally relies heavily on hardware-level optimizations, such as intelligently distributing and loading parameters and optimizing computation graphs (Alizadeh et al., 2024; Wang et al., 2024; Song et al., 2024), while more general optimizations like speculative decoding and quantization are also used (Xu et al., 2023; Navardi et al., 2025; Chen et al., 2025). On-device RAG introduces the added challenge of managing a local index, which prior works address through compartmentalized indexes (Park et al., 2025), lexical filtering (Kang et al., 2026), and improved memory management (Shao et al., 2025). Our approach is largely orthogonal to these low-level optimizations. By significantly reducing context size through compressed document representations, our method can be seamlessly combined with existing techniques. Furthermore, unifying retrieval and generation representations saves disk space and reduces latency, as relevant documents are already loaded into memory immediately following the retrieval stage.

3 Unifying embedding, compression, and generation

We propose a framework for unifying embedding, compression, and generation into a single model. An overview of using this unified model for RAG is shown in Figure 1. We refer to models from our framework as ECG models, denoting their three unified capabilities. This section details the architecture and multi-task training recipe used to achieve this unification.

3.1 Retrieval similarity

Our query and document representations can contain multiple embeddings which enable high-fidelity compression of information. Thus, we need a fast similarity measure suited for

multi-vector representations. With these goals in mind, we adopt a mean-pooled variant of the MaxSim similarity introduced by ColBERT (Khattab & Zaharia, 2020): $S(\mathbf{E}_q, \mathbf{E}_d) = \frac{1}{m}\sum_i \max_j(\mathbf{E}_q^i \cdot \mathbf{E}_d^j)$. We use the mean over the m query embeddings rather than a strict sum to account for variable-length query representations without penalizing queries with fewer embeddings.

3.2 Model representations

To adapt a pretrained decoder transformer θ_{LM} to also act as an encoder (produce continuous embeddings), we prepend a task prompt p_{embed} to the input text t and append a sequence of $n+2$ special tokens: $(s_{start}, s_{emb}, \ldots, s_{emb}, s_{end})$. The final hidden states corresponding to the s_{emb} tokens, denoted $\mathbf{E} \in \mathbb{R}^{n \times d}$, are passed through two projection blocks.

The first block, $\theta_{ret} : \{\mathbf{E}\} \to \mathbf{E}_{ret} \in \mathbb{R}^{n \times m}$, produces retrieval representations optimized for MaxSim similarity. Note the dimension m can differ from d for efficiency, but we keep $m = d$ in all our experiments. The second block, $\theta_{comp} : \{\mathbf{E}_{ret}\} \to \mathbf{E}_{comp} \in \mathbb{R}^{n \times d}$, transforms $\mathbf{E}_{ret}$ into compressed context representations suitable as input to θ_{LM} for generation. Constructing $\mathbf{E}_{comp}$ directly from $\mathbf{E}_{ret}$ ensures that only one set of representations needs to be stored for both retrieval and generation. Implementation details for these blocks are provided in Section 4.1.

To generate text conditioned on k distinct compressed contexts, we construct the input sequence by wrapping each context representation $\mathbf{E}_{comp}^{(i)}$ in special tokens: Input $=$ prefix $\oplus$ $\left(\bigoplus_{i=1}^{k}[s_{start} \oplus \mathbf{E}_{comp}^{(i)} \oplus s_{end}]\right) \oplus$ suffix, where $\oplus$ denotes sequence concatenation.

3.3 Unified training recipe

We employ a multi-task training approach consisting of a self-supervised pretraining phase to adapt the model to continuous representations, followed by RAG-specific fine-tuning.

Self-supervised training In this stage, we utilize an unlabeled passage collection to teach the model to compress and understand context representations and produce meaningful retrieval representations. For a given passage, we split the text in half to create context and target pairs using two distinct strategies with equal probability: (1) *reconstruction*, where the target text is identical to the context text, and (2) *neighboring text*, where the target is the other half of the passage. These strategies, inspired by Rau et al. (2024), provide complementary signals: reconstruction forces the model to exactly compress information while neighboring text requires the compressed context to be useful for predicting surrounding text.

The context text is encoded into retrieval representations $\mathbf{E}_{ret}$ and compressed context representations $\mathbf{E}_{comp}$. The compressed context $\mathbf{E}_{comp}$ is then fed into θ_{LM} to generate the target text via standard next-token prediction. Simultaneously, we use $\mathbf{E}_{ret}$ to compute an in-batch contrastive loss, ensuring that representations for the context and target pairs have higher MaxSim similarity than representations of other random texts in the batch.

To maximize computational efficiency, we perform generation bidirectionally (using the target as context to generate the original context) so that every encoding step provides a training signal. Furthermore, we uniformly vary the number of s_{emb} tokens during this stage, forcing the model to learn effective compression across a dynamic number of representations. The full mathematical formulation of the self-supervised losses and further discussion on how this extends prior work can be found in Appendix B.4 while the full training details can be found in Section 4.2.

RAG-specific training To adapt the model for RAG, we fine-tune it to generate answers from multiple compressed context documents. To prevent the generative component from overfitting, we replace standard next-token prediction with knowledge distillation. Specifically, we utilize two distinct teacher models: a generative reader θ_{reader} for answer distillation, and a teacher scoring model to provide query-document relevance scores.

Given a question, an answer, a positive document, hard negatives, and teacher relevance scores, we encode all queries and documents into retrieval representations $\mathbf{E}_{ret}$ and compressed contexts $\mathbf{E}_{comp}$ using a fixed number of s_{emb} tokens, t. The model is trained jointly to optimize generation and retrieval. For generation, we minimize the KL divergence between our model (conditioned on compressed contexts) and the teacher reader (conditioned on uncompressed documents). For retrieval, we apply an InfoNCE contrastive loss (van den Oord et al., 2018) alongside a Margin MSE loss (Hofstätter et al., 2020) to distill the ranking capability of the teacher scoring model into our student model. The final joint loss is $\mathcal{L} = \mathcal{L}_{gen} + \mathcal{L}_{contrastive} + \mathcal{L}_{margin}$. Full mathematical formulations of these objectives are provided in Appendix B.5.1, and training parameters are detailed in Section 4.2.

4 Experiment details

4.1 Model implementation

We evaluate our approach using the instruction-tuned version of two pre-trained decoder language models chosen to represent sizes plausible for on-device RAG applications: SmolLM-v2 135M (Allal et al., 2025) and Gemma 3 1B (Gemma-Team, 2025). We augment the model and tokenizer with special tokens representing the start, body, and end of the embedding sequence, denoted as `<emb_start>`, `<emb>`, and `<emb_stop>`, respectively. The projection blocks θ_{comp} and θ_{ret} share the same architecture designed to transform the hidden states effectively while maintaining training stability using a gated residual connection and linear layers. The full architectural details for θ_{comp} and θ_{ret} are in Appendix A.1.

4.2 Training details

Self-supervised training details For the self-supervised pre-training stage, we utilize a chunked version of Wikipedia released by Karpukhin et al. (2020). Each chunked passage contains approximately 100 words. We train for a single epoch, using the last checkpoint for our evaluation. For each model trained with the self-supervised pretraining stage, including ablation models, we do a hyperparameter grid search to ensure results are comparable. The hyperparameter search and selected hyperparameters are detailed in Appendix G.

RAG training details Following the self-supervised stage, we do specialized RAG training, utilizing data from the Natural Questions (NQ) (Kwiatkowski et al., 2019) and Trivia QA (Joshi et al., 2017) training datasets. More training data details are in Appendix B.5.

During training, we sample one positive document, two negative documents, and one answer to act as a single example. A batch is constructed with several such examples. The positive document and the answer are picked randomly with equal probability, while the negative documents use a weighted sampling approach detailed in Appendix B.5.3. We found that, compared with sampling negatives uniformly, the weighted approach produced better results; we explore the impact in our ablation study in Appendix E. The number of s_{emb} tokens used for both queries and documents is a constant value, t. For the SmolLM variants, we use $t = 32$ while for the Gemma variants we use $t = 16$. We selected these values to make the total information stored roughly the same as what a standard ColBERT model stores. Because the hidden sizes of Gemma and SmolLM are 1,152 and 576, respectively, the total space required for the representations is equivalent, allowing for a more direct comparison between the two models. Additional training details are presented in Appendix B.5.

4.3 Evaluation

To evaluate our method, we do end-to-end evaluation using the Exact Match (EM) accuracy metric. Although our focus is on-device RAG, due to the lack of realistic datasets for the on-device setting, such as personal document search, we use a standard set of RAG benchmarks. This has the additional benefit of allowing for easier comparison with existing work. Although we are using standard datasets, we wanted to make sure we considered

the limitations that come with on-device RAG. Specifically, as context length is limited in on-device settings, our evaluation highlights performance at given context budgets. We define a context budget as the number of document representations, either representing tokens or compressed vectors, that are used by the generator. For standard reader models, we truncate the document texts to ensure the context budget is met. To get a comprehensive understanding of performance-context-budget trade-off we focus on two evaluation scenarios:

Fixed Context Budget In this setup, we use a fixed number of document representations for our method and the baseline methods. This shows the relative capacity of the various methods with a constrained context budget.

Fixed Performance In this setup, we use the performance of the ECG models, with a single document, as a fixed performance target and increase the baseline context budgets until the baselines match the ECG models' performance (or perform best if no context budget matches our performance). We do a grid search from 32 to 256 in intervals of 32. This evaluation approach shows the compression that the ECG models can achieve while matching or exceeding the baseline performance.

The context budget is applied only to the document representations in the model's input and does not include question tokens or formatting tokens, including the special tokens s_{start} and s_{end}. In validation tests, we found that the ECG models generally performed best with only the top-ranking document and that adding additional documents generally reduced performance. Thus, in our main evaluation, we use only a single document as input to our models unless otherwise specified. We discuss this finding in more detail in Appendix D.

4.3.1 Evaluation datasets

We evaluate our approach on two common RAG datasets: NQ Test (Kwiatkowski et al., 2019) and TriviaQA Test (Joshi et al., 2017). We use the versions released by FlashRAG (Jin et al., 2025). For each dataset we use the Wikipedia passages released by (Karpukhin et al., 2020) to form a corpus. To enable efficient evaluation we use a pooling approach to reduce the large corpus size to a more manageable size. The details can be found in Appendix B.2.

4.3.2 Validation

To select the best checkpoints for both our model and the baseline models, as well as to understand our model's performance, we use two validation sets. The first consists of 1k questions sampled from the training set described in Section 4.2; the validation data are removed from the training data. We use the validation loss from this set to select the best checkpoint for the ECG and baseline models. For model development, we use 500 questions sampled from the NQ validation set with a pooled set of documents to form the corpus.

4.3.3 Baselines

For baselines, we include three alternatives to our approach:

Parametric Knowledge Only This baseline leverages the model's parametric knowledge to answer a question directly without any additional information from retrieved documents. To align the model with the dataset format, we train this model with the same data as our other models, but only provide the question and directly generate the answer.

Standard RAG This baseline uses two models: a retrieval model and a reader model. The reader model takes the question and retrieved documents as text to answer the question. We use three different retrieval models outlined below.

RAG with Context Compression Like Standard RAG, this baseline uses a retrieval model, but instead of using the raw text for documents it uses compressed documents as context. To enable a direct comparison, we use the same number of representations as the ECG models. Our context compression approach is based on COCOM (Rau et al., 2024), which directly uses the transformer's final hidden state as the compressed representation. We

	SmolLM 2 135M			Gemma 3 1B		
Method	**Context Budget**	**NQ**	**Trivia QA**	**Context Budget**	**NQ**	**Trivia QA**
Baselines						
Parametric (No Ret.)	-	0.069	0.110	-	0.142	0.267
RAG:						
BM25	32	0.047	0.195	16	0.061	0.241
Dense	32	0.096	0.231	16	0.099	0.242
ColBERT	32	0.106	0.297	16	0.104	0.303
RAG w/ Context Comp:						
BM25	32	0.087	0.242	16	0.144	0.318
Dense	32	0.192	0.303	16	0.245	0.351
ColBERT	32	0.188	0.351	16	0.247	0.393
Ours						
ECG	32	**0.343**	**0.515**	16	**0.361**	**0.540**

Table 1: Comparison between Exact Match (EM) accuracy for ECG models (ours) and baseline models with a fixed context budget, which is the number of vectors used to represent the context documents.

do not include xRAG as a baseline because, as shown by Rau et al. (2024), it performs far worse than the end-to-end multi-vector representation we adopt. Our choice of a context compression baseline is intended to serve as a reasonable proxy for strong context compression models while removing many confounding factors such as base model, training data, and modeling approach.

To ensure that the comparison with the baseline models is fair, we initialize from the same starting models and use the same training data when applicable. The full training details for the baselines are in Appendix B.3. For the RAG-based methods we consider three retrieval methods that vary in effectiveness and efficiency. These are:

GTE ModernBERT Base (Li et al., 2023c) A dense retrieval model based on ModernBERT (Warner et al., 2025), a modernized encoder model with 149M parameters. It is trained on a large-scale dataset of text pairs across many domains, including Wikipedia, before being trained on high-quality retrieval datasets including NQ and TriviaQA.

GTE ModernColBERT V1 (Chaffin, 2025) This model is similar to the original ColBERT model, but uses GTE ModernBERT Base as the base model. It is trained on the MSMARCO passage dataset and represents one of the strongest retrieval models for its size.

BM25 (Robertson & Walker, 1994) A classic lexical approach which does not have any learned weights, but has been shown to be highly capable and robust across domains.

Additional implementation details for the baseline retrieval approaches are presented in Appendix B.1.

5 Results and discussion

To demonstrate the viability of our unified ECG model for on-device applications, we structure our evaluation around four primary research questions (RQs). On-device deployment involves strict hardware constraints; for instance, typical mobile devices operate with shared, limited unified memory. Large text contexts in standard RAG balloon the KV cache, leading to out-of-memory errors or additional battery drain, while storing multiple retrieval and context representations exhausts precious disk space. Our evaluation addresses these constraints through the following RQs:

- **RQ1:** How does the unified ECG model perform under constrained on-device context budgets compared to the baselines, including standard RAG?
- **RQ2:** What level of context efficiency and disk space reduction does unifying representations achieve while matching standard reader performance?

		SmolLM 2 135M						Gemma 3 1B					
		NQ			Trivia QA			NQ			Trivia QA		
Method	**Disk Space / 10k**	**Context**	**Comp.**	**EM**	**Context**	**Comp.**	**EM**	**Context**	**Comp.**	**EM**	**Context**	**Comp.**	**EM**
Baselines													
Parametric (No Ret.)	0.0	-	-	0.069	-	-	0.110	-	-	0.142	-	-	0.267
RAG:													
BM25	0.0*	256	8×	0.163	256	8×	0.409	256	16×	0.195	256	16×	0.468
Dense	0.031	256	8×	0.327	256	8×	0.477	256	16×	0.358	256	16×	0.521
ColBERT	0.744	256	8×	0.328	160	5×	**0.520**	224	14×	**0.361**	128	8×	**0.545**
RAG w/ Context Comp:													
BM25	0.737	96	3×	0.102	96	3×	0.278	64	4×	0.177	80	5×	0.370
Dense	0.768	96	3×	0.202	96	3×	0.324	64	4×	0.265	80	5×	0.392
ColBERT	1.481	96	3×	0.217	96	3×	0.372	64	4×	0.278	80	5×	0.438
Ours													
ECG	0.737	**32**	-	**0.343**	**32**	-	0.515	**16**	-	**0.361**	**16**	-	0.540

Table 2: Performance comparison of the Exact Match (EM) accuracy between the ECG models and baseline models at a fixed performance level. Disk Space / 10k, represents the size the index/embeddings for 10k passages. For all the embedding based retrieval models and ECG models we only count the size of the embeddings as otherwise the size would be heavily dependent on the indexing approach used which is orthogonal to our contribution. The columns Context and Comp. represent the context budget and relative compression of the ECG model. *The size of BM25 is actually, 9.23e−6, but for consistent formatting we rounded it to 0.0 in the table.

- **RQ3:** Does joint training of retrieval and compression representations outperform isolated context compression?
- **RQ4:** What factors, including architectural and training components, drive ECG's performance?

RQ1: Performance under constrained context budgets Table 1 presents the fixed context budget evaluation, demonstrating that ECG models substantially outperform all baselines under strict active memory (KV cache) constraints. When restricted to a budget of 32 representations for SmolLM and 16 for Gemma, standard RAG models struggle significantly. In contrast, the ECG models achieve over three times the Exact Match score of the best standard RAG models on NQ for both architectures (0.343 vs. 0.106 for SmolLM; 0.361 vs. 0.104 for Gemma). While the context compression baselines fare slightly better than standard RAG at these budgets, ECG still outperforms the strongest isolated context compression baseline by substantial margins across all datasets and model sizes. Ultimately, when active memory is strictly capped, standard RAG fails, whereas ECG remains robust.

RQ2: Context efficiency and disk space Table 2 illustrates the context budget required for baselines to match the single-document performance of the ECG models. Standard compress-and-generate RAG systems typically require storing two distinct sets of vectors: one for retrieval (e.g., ColBERT) and one for context generation (e.g., COCOM). By sharing representations, ECG cuts the disk-space requirement in half, using slightly less space than ColBERT alone, while simultaneously solving the context-length memory issue.

The results show that the vast majority of baselines—including all parametric, BM25, Dense RAG, and Context Compression models—fail to reach ECG's performance entirely, even when given up to 8× or 16× the context budget. Standard RAG with ColBERT retrieval is the only baseline capable of matching ECG's accuracy, but it requires a substantial context increase to do so. Specifically, it requires 5× the budget on SmolLM TriviaQA, 14× the budget on Gemma NQ, and 8× the budget on Gemma TriviaQA. Furthermore, on the SmolLM NQ dataset, even when given a maximum budget of 256 representations (8× the ECG budget), RAG ColBERT (0.328) still falls short of the ECG model's performance (0.343).

RQ3: Unified modeling vs. isolated compression As Tables 1 and 2 show, the ECG models perform better than the context compression baselines across the board. We investigate

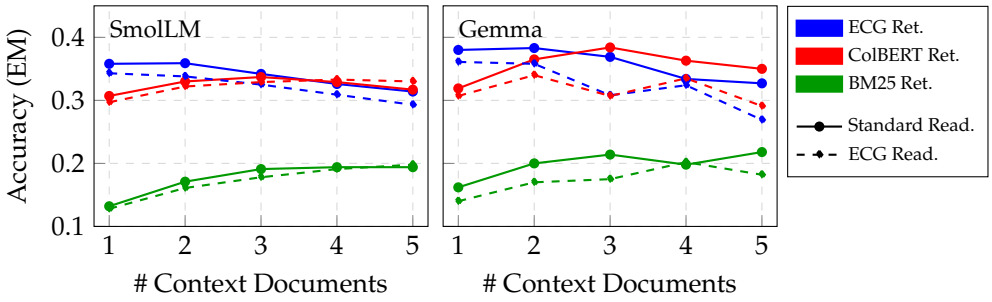


Figure 2: Accuracy (EM) of various retriever and reader combinations on NQ across different values of k for SmolLM and Gemma models.

the underlying cause of this difference by training and evaluating variants which isolate various aspects of each model. Specifically, we train a context compression model with exactly the same architecture as the ECG model, including the MLP blocks and use the ECG model on ColBERT's retrievals to isolate the compression and generation quality. The results in Table 3 show the performance discrepancy is partially due to the architectural differences and retrieval quality, but there is still a substantial gap when these are accounted for (0.224 vs. 0.297 on NQ and 0.386 vs. 0.485 on TriviaQA) which suggests that unifying retrieval and compression not only improves efficiency but also improves effectiveness. We hypothesize that ECG's multi-task training objective—specifically forcing the vectors to satisfy the contrastive retrieval loss—acts as regularization. This yields richer, more robust context representations than compression training alone.

RQ4: Driver of performance and architectural impact The ECG models perform impressively compared to standard RAG as discussed in RQ2. In fact, in several instances standard RAG approaches cannot match the ECG models even with the maximum context. To understand what is behind this strong performance, we investigate hybrid evaluations where the ECG model is used only as a retriever or only as a reader and compare against the baseline readers and retrievers. The NQ results are presented in Figure 2. We can see that the ECG model's performance is bolstered by excellent retrieval quality with standard readers having a clear increase when using the ECG retrieval over the best baseline retriever, ColBERT. Although the ECG reader is generally worse than the standard reader, the gap is generally minimal considering the context size used by the ECG reader is far smaller. We also note that the standard reader also does best with only one or two documents when using the ECG retrievals, showing that the preference for fewer documents is not unique to the ECG models. For additional discussion of this finding, see Appendix D.

As we've discussed, unifying retrieval, compression, and generation into a single representation space is not trivial. To understand these challenges, Table 4 presents an ablation study of the SmolLM ECG model. We find that contrastive loss during pretraining is a necessary addition for performance on NQ with a notable drop from 0.343 to 0.337 when removed, while there is a minimal decrease in TriviaQA. Generative distillation is also crucial, providing substantially higher-quality outputs compared to standard next-token prediction. However, the most significant challenge lies in balancing the retrieval and generation objectives during joint training. Removing dynamic loss scaling—the learned temperature and teacher score scale for the retrieval losses—causes NQ performance to plummet by nearly half (from 0.343 to 0.173). This demonstrates that without precise alignment of the teacher's scores to the student distribution, the multi-task objectives actively compete and degrade the shared representations. An expanded ablation section is available in Appendix E.

Impact of context length on on-device usability metrics In the prior sections, we use context length as a proxy for important on-device restrictions such as KV-cache size and generation latency. In this section, we validate the correlation between context length and real-world usability for on-device RAG systems. Specifically, we generate 50 tokens for both SmolLM and Gemma given various starting context lengths using an M1 MacBook

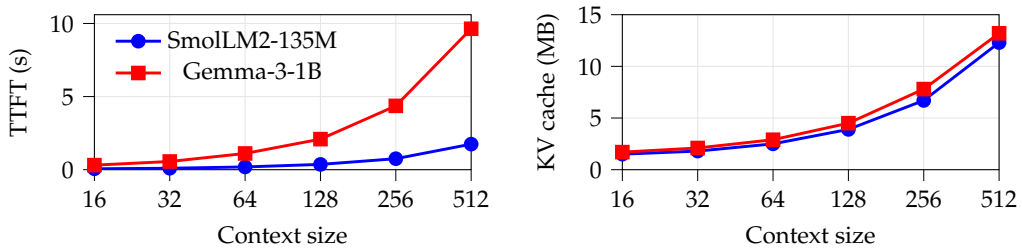


Figure 3: On-device inference costs as context size increases on an M1 MacBook Pro. TTFT shows the mean value over 10 runs. Each run generates 50 new tokens.

Pro. This simulates a RAG task with varying context sizes and allows us to understand how context size impacts the practicality of on-device RAG. The complete details are available in Appendix F. We investigate two measures: (1) time to first token (TTFT), which is important for the user experience because users may be unwilling to wait for an answer, and (2) KV-cache size, which is important given the limited memory budgets inherent in on-device use cases. The impact of context size on these metrics is shown in Figure 3. We can see there is a clear increase in both TTFT and KV cache size as context length increases which validates our use of context size as a proxy for real-world on-device metrics.

Limitations We use standard RAG benchmarks and evaluation procedures to support controlled comparisons with prior work, while leaving more comprehensive validation of the method in realistic on-device settings to future work. We evaluate on Natural Questions and TriviaQA using Wikipedia passages, and therefore do not directly study personal documents or other non-Wikipedia domains that motivate on-device RAG. These datasets also primarily contain questions answerable from a single passage, so further evaluation is needed for settings where evidence must be combined across multiple documents. While we report latency and memory measurements on a consumer laptop using CPU inference, we do not evaluate a complete on-device deployment or optimize the implementation for mobile hardware, specialized accelerators, or mobile inference runtimes. We also do not evaluate quantized models or vector compression; although these techniques could further reduce storage and memory requirements, their impact on both retrieval effectiveness and the ability of the shared representations to support generation remains unknown. Additionally, retrieval is performed using exact similarity computation, and future work could explore how well approximate indexing methods work for ECG representations. Finally, although the number of representation tokens is varied during self-supervised training, RAG-specific training uses a fixed number for each model, leaving evaluation across a wider range of representation sizes to future work.

6 Conclusion

In this work, we show that retrieval and context representations for a document can be shared for RAG and that this can be done with a unified model acting as the compressor, encoder, and generator. Our approach uses the same space as other powerful late-interaction models such as ColBERT, but uses far fewer context representations for comparable performance when compared to standard text reader models. When compared with a stand-alone context compressor paired with a retrieval model, our method substantially outperforms the context compression model even when evaluated with the same context documents, indicating that combining retrieval and compression representations brings additional robustness and performance gain. Furthermore, our approach requires half the storage compared to the standard context compression approach which requires two document representations, one to act as generation context and one as the retrieval representation. Our work enables on-device RAG to have high performance with a relatively tiny context size and minimal disk usage. We hope it inspires novel and more efficient unified approaches to RAG.

Acknowledgments

We used LLMs in various capacities throughout the experimental process and in the writing process. Specifically, LLMs were used to write code for some of the experimental setup, produce tables and figures (but not add or create their content), suggest potentially relevant related work (we never directly use an LLM to write or generate citations; we use it only to suggest papers that are manually reviewed), proofread various writing, and rewrite certain sections of the paper. All LLM outputs were carefully verified by one or more of the authors to ensure correctness. We take full responsibility for all content and findings reported in this paper.

This work was supported in part by the Center for Intelligent Information Retrieval and in part by the NSF Graduate Research Fellowships Program (GRFP) Award #1938059. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, S. Khatamifard, Minsik Cho, Carlo C. del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. LLM in a flash: Efficient large language model inference with limited memory. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 12562–12584. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.678. URL https://doi.org/10.18653/v1/2024.acl-long.678.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George 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 W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens, 2022. URL https://arxiv.org/abs/2112.04426.

Antoine Chaffin. Gte-moderncolbert, 2025. URL https://huggingface.co/lightonai/GTE-ModernColBERT-v1.

Antoine Chaffin and Raphaël Sourty. Pylate: Flexible training and retrieval for late interaction models. In Meeyoung Cha, Chanyoung Park, Noseong Park, Carl Yang, Senjuti Basu Roy, Jessie Li, Jaap Kamps, Kijung Shin, Bryan Hooi, and Lifang He (eds.), *Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM 2025, Seoul, Republic of Korea, November 10-14, 2025*, pp. 6334–6339. ACM, 2025. doi: 10.1145/3746252.3761608. URL https://github.com/lightonai/pylate.

Zhiyang Chen, Daliang Xu, Haiyang Shen, Mengwei Xu, Shangguang Wang, and Yun Ma. Accelerating mobile language model via speculative decoding and npu-coordinated execution. *CoRR*, abs/2510.15312, 2025. doi: 10.48550/ARXIV.2510.15312. URL https://doi.org/10.48550/arXiv.2510.15312.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural*

Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/c5cf13bfd3762821ef7607e63ee90075-Abstract-Conference.html.

Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Ben Hu. Learning to compress prompt in natural language formats. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 7756–7767. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-LONG.429. URL https://doi.org/10.18653/v1/2024.naacl-long.429.

Maik Fröbe, Andrew Parry, Harrisen Scells, Shuai Wang, Shengyao Zhuang, Guido Zuccon, Martin Potthast, and Matthias Hagen. Corpus subsampling: Estimating the effectiveness of neural retrieval models on large corpora. In Claudia Hauff, Craig Macdonald, Dietmar Jannach, Gabriella Kazai, Franco Maria Nardini, Fabio Pinelli, Fabrizio Silvestri, and Nicola Tonellotto (eds.), *Advances in Information Retrieval - 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6-10, 2025, Proceedings, Part I*, Lecture Notes in Computer Science, pp. 453–471. Springer, 2025. doi: 10.1007/978-3-031-88708-6_29. URL https://doi.org/10.1007/978-3-031-88708-6_29.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 6894–6910. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.552. URL https://doi.org/10.18653/v1/2021.emnlp-main.552.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=uREj4ZuGJE.

Gemma-Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Shuyu Guo and Zhaochun Ren. Enhancing RAG efficiency with adaptive context compression. *CoRR*, abs/2507.22931, 2025. doi: 10.48550/ARXIV.2507.22931. URL https://doi.org/10.48550/arXiv.2507.22931.

Jie He, Richard He Bai, Sinead Williamson, Jeff Z. Pan, Navdeep Jaitly, and Yizhe Zhang. Clara: Bridging retrieval and generation with continuous latent reasoning, 2026. URL https://arxiv.org/abs/2511.18659.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. Improving efficient neural ranking models with cross-architecture knowledge distillation. *CoRR*, abs/2010.02666, 2020. URL https://arxiv.org/abs/2010.02666.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 13358–13376. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.825. URL https://doi.org/10.18653/v1/2023.emnlp-main.825.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 737–740, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713316. doi: 10.1145/3701716.3715313. URL https://doi.org/10.1145/3701716.3715313.

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/.

Hoyoun Jung and Kyung-Joong Kim. Discrete prompt compression with reinforcement learning. *IEEE Access*, 12:72578–72587, 2024. doi: 10.1109/ACCESS.2024.3403426. URL https://doi.org/10.1109/ACCESS.2024.3403426.

Dong Ho Kang, Hyunjoon Lee, Hyeonjeong Cha, Minkyu Choi, and Sungsoo Lim. Pocket rag: On-device rag for first aid guidance in offline mobile environment, 2026. URL https://arxiv.org/abs/2602.13229.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (eds.), *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pp. 39–48. ACM, 2020. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

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.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. Making large language models a better foundation for dense retrieval, 2023a.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 6342–6353. Association for Computational Linguistics, 2023b. doi: 10.18653/V1/2023.EMNLP-MAIN.391. URL https://doi.org/10.18653/v1/2023.emnlp-main.391.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023c.

Zongqian Li, Yixuan Su, and Nigel Collier. 500xcompressor: Generalized prompt compression for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 -*

August 1, 2025, pp. 25081–25091. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.1219/.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pp. 2356–2362, 2021.

Maxime Louis, Hervé Déjean, and Stéphane Clinchant. PISCO: Pretty simple compression for retrieval-augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15506–15521, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.800. URL https://aclanthology.org/2025.findings-acl.800/.

Jesse Mu, Xiang Li, and Noah D. Goodman. Learning to compress prompts with gist tokens. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/3d77c6dcc7f143aa2154e7f4d5e22d68-Abstract-Conference.html.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=BC4lIvfSzv.

Mozhgan Navardi, Romina Aalishah, Yuzhe Fu, Yueqian Lin, Hai Li, Yiran Chen, and Tinoosh Mohsenin. Genai at the edge: Comprehensive survey on empowering edge devices. In Ron P. A. Petrick and Christopher W. Geib (eds.), *Proceedings of the 2025 AAAI Spring Symposium Series, San Francisco, CA, USA, March 31-April 2, 2025*, pp. 180–187. AAAI Press, 2025. doi: 10.1609/AAAISS.V5I1.35586. URL https://doi.org/10.1609/aaaiss.v5i1.35586.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 963–981. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.57. URL https://doi.org/10.18653/v1/2024.findings-acl.57.

Taehwan Park, Geonho Lee, and Min-Soo Kim. Mobilerag: A fast, memory-efficient, and energy-efficient method for on-device RAG. *CoRR*, abs/2507.01079, 2025. doi: 10.48550/ARXIV.2507.01079. URL https://doi.org/10.48550/arXiv.2507.01079.

Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. Context embeddings for efficient answer generation in RAG. *CoRR*, abs/2407.09252, 2024. doi: 10.48550/ARXIV.2407.09252. URL https://doi.org/10.48550/arXiv.2407.09252.

S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pp. 232–241, Berlin, Heidelberg, 1994. Springer-Verlag. ISBN 038719889X.

Shivam Shandilya, Menglin Xia, Supriyo Ghosh, Huiqiang Jiang, Jue Zhang, Qianhui Wu, Victor Rühle, and Saravan Rajmohan. TACO-RL: task aware prompt compression optimization with reinforcement learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 1582–1597. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.81/.

Kunming Shao, Zhipeng Liao, Jiangnan Yu, Liang Zhao, Qiwei Li, Xijie Huang, Jingyu He, Fengshi Tian, Yi Zou, Xiaomeng Wang, Kwang-Ting (Tim) Cheng, and Chi-Ying Tsui. DIRC-RAG: accelerating edge RAG with robust high-density and high-loading-bandwidth digital in-reram computation. In *IEEE/ACM International Symposium on Low Power Electronics and Design, ISLPED 2025, Reykjavík, Iceland, August 6-8, 2025*, pp. 1–7. IEEE, 2025. doi: 10.1109/ISLPED65674.2025.11261807. URL https://doi.org/10.1109/ISLPED65674.2025.11261807.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade GPU. In Emmett Witchel, Christopher J. Rossbach, Andrea C. Arpaci-Dusseau, and Kimberly Keeton (eds.), *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles, SOSP 2024, Austin, TX, USA, November 4-6, 2024*, pp. 590–606. ACM, 2024. doi: 10.1145/3694715.3695964. URL https://doi.org/10.1145/3694715.3695964.

Nandan Thakur, Crystina Zhang, Xueguang Ma, and Jimmy Lin. Hard negatives, hard lessons: Revisiting training data quality for robust information retrieval with LLMs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 9064–9083, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.481. URL https://aclanthology.org/2025.findings-emnlp.481/.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Zhaode Wang, Jingbang Yang, Xinyu Qian, Shiwen Xing, Xiaotang Jiang, Chengfei Lv, and Shengyu Zhang. MNN-LLM: A generic inference engine for fast large language model deployment on mobile devices. In Ruili Wang, Zhiyong Wang, Jiaying Liu, Alberto Del Bimbo, Jun Zhou, Anup Basu, and Min Xu (eds.), *Proceedings of the 6th ACM International Conference on Multimedia in Asia, Workshops, MMAsia 2024, Auckland, New Zealand, December 3-6, 2024*, pp. 11:1–11:7. ACM, 2024. doi: 10.1145/3700410.3702126. URL https://doi.org/10.1145/3700410.3702126.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 2526–2547. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.127/.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models.

In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 5621–5634. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-EMNLP.412. URL https://doi.org/10.18653/v1/2022.findings-emnlp.412.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. Llmcad: Fast and scalable on-device large language model inference. *CoRR*, abs/2309.04255, 2023. doi: 10.48550/ARXIV.2309.04255. URL https://doi.org/10.48550/arXiv.2309.04255.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=mlJLVigNHp.

Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. Retrieval-enhanced machine learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pp. 2875–2886, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531722.

Caojin Zhang, Qiang Zhang, Ke Li, Sai Vidyaranya Nuthalapati, Benyu Zhang, Jason Liu, Serena Li, Lizhu Zhang, and Xiangjun Fan. GEM: empowering LLM for both embedding generation and language understanding. *CoRR*, abs/2506.04344, 2025. doi: 10.48550/ARXIV.2506.04344. URL https://doi.org/10.48550/arXiv.2506.04344.

Jintian Zhang, Cheng Peng, Mengshu Sun, Xiang Chen, Lei Liang, Zhiqiang Zhang, Jun Zhou, Huajun Chen, and Ningyu Zhang. OneGen: Efficient one-pass unified generation and retrieval for LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4088–4119, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.237. URL https://aclanthology.org/2024.findings-emnlp.237/.

A Modeling details

A.1 Implementation of θ_{comp} and θ_{ret}

The projection blocks θ_{comp} and θ_{ret} share the same architecture designed to transform the hidden states effectively while maintaining training stability. Both blocks consist of a stack of $L = 4$ identical layers with hidden dimension d matching the underlying language model. Formally, let h_l denote the input to the l-th layer. The output h_{l+1} is computed via a gated residual connection:

$$h_{l+1} = \sigma(W_{gate} h_l + b_{gate}) \odot h_l + \text{ReLU}(W_{proj}\text{LN}(h_l) + b_{proj})$$

where σ is the sigmoid function, $\odot$ denotes element-wise multiplication, $W_{gate} \in \mathbb{R}^{1\times d}$ represents the weight for the gated residual, $W_{proj} \in \mathbb{R}^{d\times d}$ represents the weight for the linear layer, and LN represents Layer Normalization. For the final layer of each block, we do not apply the ReLU activation so as not to restrict the representations to only positive values. To facilitate gradient flow and preserve information from the pre-trained model during the

early stages of training, the bias b_{gate} is initialized to a large positive value, ensuring the gate output is initially close to 1.

The final value of $\mathbf{E}_{ret}$ is found by dividing by $\sqrt{m}$, where m is the dimension of the output from θ_{ret}. Before passing this into θ_{comp} we multiply by $\sqrt{m}$. This is done to normalize the inner products produced by $\mathbf{E}_{ret}$ similar to what is done during multi-head attention (Vaswani et al., 2017). A more common approach in retrieval models is to normalize vectors to unit length, but in our context this may reduce information and impact the generation input. It would be possible to store the magnitude and rescale in the same way we do with $\sqrt{m}$, but this adds additional complexity, so, for simplicity, we use $\sqrt{m}$.

B Additional implementation, evaluation, and training Details

B.1 Implementation and resources

We implemented our models and performed training using PyTorch (Paszke et al., 2019) and the Huggingface Transformers library (Wolf et al., 2020). We used 8 NVIDIA H100 GPUs for training tasks and NVIDIA A100s for inference. For GTE ModernColBERT V1 we use the Pylate (Chaffin & Sourty, 2025) library implementation. For BM25 we use Pyserini (Lin et al., 2021) with default arguments.

We perform retrieval by doing exact similarity computations for each document in the corpus (note that we use pooling to reduce the corpus sizes as mentioned in Section 4.3.1). This was done using custom PyTorch code. We choose this over an approximate approach because it reduces the confounding factors and because we found that existing implementations for approximate late-interaction retrieval such as PLAID were not designed for large embedding dimensions (e.g. > 256). Note that, in principle, the algorithms are dimension-agnostic; it is only certain choices in current implementations that cause problems.

B.2 Evaluation corpus details

To enable efficient evaluation we use a pooling approach to reduce the large corpus size of the Wikipedia passage corpus to a more manageable size. Our pooling procedure uses four different retrieval models to find the top 100 passages for each test question and combines the retrieved documents into a unified corpus. We use the three baseline retrieval models described in Section 4.3.3 and an early version of our model to construct the pooled corpora. Though this changes the retrieval task, prior work has shown pooling can be an effective way to reduce the corpus size while still ensuring a valid evaluation of full-scale retrieval performance (Fröbe et al., 2025). After the pooling process, the corpus constructed for NQ has approximately 840k passages and the corpus constructed for TriviaQA has approximately 2.1M passages.

B.3 Baseline training details

To ensure that the comparison with the baseline models is fair, we initialize from the same starting models and use the same training data when applicable. For the Parametric and Standard RAG reader models we train on only the RAG training data, though the parametric model only sees the question without retrieved documents. For the context compression approach we include a pretraining phase like the ECG models. The pretraining phase is identical to the ECG models except there is no contrastive loss. For the RAG training phase the data and batch processing are the same as the ECG models, but without any retrieval losses. Like the ECG models the context compression models use knowledge distillation as the main generative loss. The parametric models and reader models use next-token prediction. To ensure that hyperparameter selection is not responsible for any performance disparities, we perform a learning rate and weight decay sweep for all the baseline models. The full sweep information and final hyperparameters are detailed in Appendix G.

B.4 Self-supervised training details and loss equations

During the self-supervised training stage, our model learns to compress and leverage compressed representations by predicting target text T from compressed context text. Our approach builds upon the first stage of training in COCOM (Rau et al., 2024), but with key modifications to support both compression and retrieval representation learning.

B.4.1 Generation loss

Unlike COCOM, which uses a full passage for reconstruction and half-passages for the next-segment generation task, we consistently split passages in half. This ensures there is no difference in the distribution while encoding for either the reconstruction or neighboring text strategies. The generative component uses a standard next-token prediction loss over the target text tokens, conditioned on the compressed context $\mathbf{E}_{comp}$:

$$\mathcal{L}_{lm} = -\sum_{i=1}^{|T|} \log p(t_i \mid t_{<i}, \mathbf{E}_{comp})$$

The loss is applied exclusively to the target text tokens, ignoring formatting tokens and the $\mathbf{E}_{comp}$ vectors.

B.4.2 Contrastive retrieval loss

To simultaneously train $\mathbf{E}_{ret}$ to produce meaningful representations for retrieval (using the similarity function defined in Section 3.1), we apply an in-batch contrastive loss. We treat the $\mathbf{E}_{ret}$ of the context text ($\mathbf{E}^c_{ret}$) and the target text ($\mathbf{E}^t_{ret}$) as a positive pair.

To prevent computational waste when encoding the target text solely for the retrieval loss, we perform generation in both directions (context $\rightarrow$ target, and target $\rightarrow$ context). Given a batch of B positive pairs, the contrastive loss is computed as:

$$\mathcal{L}_{contrastive} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(S(\mathbf{E}^{c,i}_{ret}, \mathbf{E}^{t,i}_{ret})/\tau)}{\sum_{j=1}^{B} \exp(S(\mathbf{E}^{c,i}_{ret}, \mathbf{E}^{t,j}_{ret})/\tau)}$$

where τ is a learnable temperature parameter.

Note that for the reconstruction task, the model encodes the exact same text twice. However, because we randomly vary the number of s_{emb} tokens for each encoding pass, the model effectively learns to map the same information to similar representations across different capacity constraints. This acts as a form of augmentation, allowing identical text to be used in contrastive training (similar to how standard dropout acts as augmentation in models like SimCSE (Gao et al., 2021)).

The final joint loss for the self-supervised stage is the sum of the generative and contrastive components:

$$\mathcal{L} = \mathcal{L}_{contrastive} + \mathcal{L}_{lm}$$

B.5 Additional RAG training details

During RAG training, the order of the context documents provided to the generative model is randomized to prevent the model from learning any positional bias. All documents are used in the retrieval losses, but to make the generative component of the model more robust, we randomize the number of documents provided for generation from one to three while ensuring that the positive document is always included.

As the generative teacher model for the knowledge distillation, we employ the reader model trained as a baseline; further details on the teacher model can be found in Section 4.3.3. As in the pretraining stage, we perform a hyperparameter search for our models using the validation set to select the best checkpoint. The full training details including hyperparameter search details can be found in Appendix G.

B.5.1 RAG-Specific training loss equations

During the RAG-specific fine-tuning stage, we optimize a joint loss function comprising generation, contrastive, and margin-based components: $\mathcal{L} = \mathcal{L}_{gen} + \mathcal{L}_{contrastive} + \mathcal{L}_{margin}$.

For the generation loss, let $\mathcal{D} = \{d_1, \ldots, d_k\}$ be the uncompressed context documents and $\mathcal{C} = \{\mathbf{E}^1_{comp}, \ldots, \mathbf{E}^k_{comp}\}$ be their corresponding compressed representations. The loss is the KL divergence between our model and the teacher reader:

$$\mathcal{L}_{gen} = \sum_t D_{KL}(P_{\theta_{reader}}(a_t \mid a_{<t}, q, \mathcal{D}) \parallel P_\theta(a_t \mid a_{<t}, q, \mathcal{C}))$$

For the retrieval representations $\mathbf{E}_{ret}$, we apply an InfoNCE contrastive loss over the positive document d^+ and all negatives $\mathcal{N}$ (hard negatives plus in-batch negatives), using a learned temperature τ:

$$\mathcal{L}_{contrastive} = -\log \frac{\exp(S(\mathbf{E}^q_{ret}, \mathbf{E}^{d^+}_{ret})/\tau)}{\exp(S(\mathbf{E}^q_{ret}, \mathbf{E}^{d^+}_{ret})/\tau) + \sum_{d^- \in \mathcal{N}} \exp(S(\mathbf{E}^q_{ret}, \mathbf{E}^{d^-}_{ret})/\tau)}$$

Finally, we distill the ranking capability of the teacher scoring model via a Margin MSE loss (Hofstätter et al., 2020) over the hard negatives $\mathcal{N}_{hard}$. We introduce a learnable scaling parameter α to account for score scale differences between the teacher and our student model:

$$\mathcal{L}_{margin} = \frac{1}{|\mathcal{N}_{hard}|} \sum_{d^- \in \mathcal{N}_{hard}} (\Delta(S, d^-) - \alpha \Delta(T, d^-))^2$$

where $\Delta(\phi, d^-) = \phi(q, d^+) - \phi(q, d^-)$ is the margin between the positive and negative document for a given scoring function ϕ (student S or scoring teacher T).

B.5.2 RAG training data construction

For the NQ subset, we use the data released by (Thakur et al., 2025), which include a known positive passage, hard negative passages, and query-passage scores using Gemma 2 Reranker (Li et al., 2023a).[1] As this dataset lacks the answer text required for the generative component of our model, we augment the data with answers obtained from FlashRAG (Jin et al., 2025). To reduce the chance of false negatives, we remove any passage which has a higher reranker score than the known positive. For TriviaQA, we use the released training set, and use BM25 on the Wikipedia corpus to retrieve hard negatives.[2] We score the top 100 documents using the same Gemma 2 Reranker as the NQ data. Because the TriviaQA data do not include a positive passage, we select the passage which has the highest Gemma Reranker score and contains the answer string. We remove any other passages that also have the answer string to prevent potential false-negatives. As a final step, we remove any questions that do not contain at least eight negatives from both the NQ and TriviaQA subsets. The final dataset contains a combination of these subsets. The final training dataset contains approximately 110k questions. We use a random (disjoint) subset of 1k questions to act as a validation set.

B.5.3 RAG weighted negative sampling

During RAG training, we sample a negative d_i^- with teacher score s_i^- with probability:

$$p_i = \frac{\exp(s_i^-/\tau)}{\sum_{j=1}^{|\mathcal{N}|} \exp(s_j^-/\tau)}, \tag{1}$$

where $\mathcal{N}$ is the full set of negatives for a given query and τ is a hyperparameter which controls the temperature. Smaller values of τ mean the documents with higher teacher scores are selected more often, while higher values make the distribution closer to uniform.

[1] https://huggingface.co/datasets/rlhn/default-680K-bge-reranker-v2-gemma

[2] https://huggingface.co/datasets/mandarjoshi/trivia_qa/viewer/rc.wikipedia

C Additional Tables

	NQ			Trivia QA		
Method	**EM ($k = 1$)**	**EM ($k = k'$)**	k'	**EM ($k = 1$)**	**EM ($k = k'$)**	k'
Baselines						
Compression	0.188	0.217	3	0.351	0.372	3
Compression w/ Same Arch	0.224	0.259	4	0.386	0.421	3
Ours						
ECG - Reader	0.297	0.333	4	0.485	**0.518**	3
ECG	**0.343**	**0.343**	1	**0.515**	**0.518**	2

Table 3: Performance comparison between context compression baseline and a modified context compression model with the same architecture as the ECG models using ColBERT retrievals. ECG-Reader is the ECG model used only as a reader on top of the ColBERT retrievals. We show the EM for a single document and the best EM found for documents (1-5) where the number of documents is given as k'.

D Discussion of accuracy drop off with more context documents

As displayed in Figure 2, when the ECG model is used as the retriever, both the ECG reader and the standard RAG reader perform best with fewer documents—generally one or two—than when other retrieval models are used. In this section, we provide a theory for this behavior. Specifically, we believe it comes down to the signal-to-noise ratio of the top documents and how this is impacted by retrieval quality.

To illustrate this, imagine a perfect retriever where the relevant document is always first. Additionally, assume that each question also only has a single relevant document. Thus, with a single context document, the reader is guaranteed to have the necessary information to answer correctly, but as we expand to include more context documents we do not provide more useful information; instead, we add noise to the model's context. If we assume the reader will degrade to some extent with additional noise, we can see how adding additional documents is actually harmful. We believe something similar is happening with the ECG model though to a lesser extent as it is not a perfect retriever. If, for many of the questions, the most relevant document is first, additional documents are only going to dilute the relevant information, but if the retriever is weak and the first document does not contain the answer adding context documents improves the signal-to-noise ratio because there is no signal without a relevant document.

This theory is supported by the performance trends for ColBERT and BM25. ColBERT has the best performance around three documents with a gradual drop with additional documents, while BM25 shows pretty consistent improvement as more documents are added even up to five context documents. As BM25 is far weaker than ColBERT, this supports the theory that more context documents are beneficial for worse retrievers.

This theory and the results for the ECG reader using other retrievers demonstrate that the ECG reader is not limited to leveraging information from one relevant document, but when acting on a strong retriever's document ranking, adding additional documents generally provides minimal useful information while often adding noise. We leave a more comprehensive study of this phenomenon for future work.

E ECG model ablations

To fully understand the impact of each component in our training pipeline, we conduct a comprehensive set of ablation studies for the SmolLM ECG model. To minimize computational cost, we used the same hyperparameters as the SmolLM ECG model except for

the ablations that changed the pretraining stages, for which we performed a hyperparameter sweep only for the pretraining stage. The results of the ablations are in Table 4, and descriptions and analyses of the individual ablations follow.

Without Contrastive (Pretraining) This ablation removes the in-batch contrastive loss during the self-supervised pretraining stage. The results show that while it is not strictly necessary for all retrieval tasks (as TriviaQA performance remains similar), it provides a notable performance boost on NQ, demonstrating that it is a useful addition for shaping the unified representations.

Without Generative Distillation This replaces the knowledge distillation objective from the teacher reader with a standard next-token prediction loss during RAG fine-tuning. The resulting drop in performance highlights that generative distillation is crucial for producing high-quality generative outputs.

Without Pretraining This skips the self-supervised pretraining phase entirely, moving straight to RAG-specific training from the base language model. The decreased accuracy confirms that teaching the model to compress and understand context representations on unlabeled data is a meaningful factor in the ECG model's strong end-to-end performance.

Without Loss Scaling This removes the dynamic loss scaling—specifically, the learned temperature and teacher score scale for the retrieval losses. The massive performance drop (falling to nearly half on NQ for $k = 1$) indicates that without precise alignment of the teacher's scores to the student distribution, the multi-task objectives actively compete and degrade the shared representations.

Without Hard Negatives (Random) This trains the model using uniformly random negatives instead of the weighted hard-negative selection used in the final ECG models. The performance gap demonstrates that leveraging harder negatives is a significant factor in developing robust retrieval capabilities.

With Negative Temperature 0.5 & 0.25 These configurations alter the τ hyperparameter used for weighted negative sampling from the 0.15 used for the standard models to 0.25 and 0.5. The severe degradation in performance under these constraints emphasizes that the model is highly sensitive to the exact hard-negative temperature, which dictates the difficulty of hard negatives during training.

A major takeaway is that the contrastive loss during pretraining is not needed for all retrieval tasks as TriviaQA performance is similar with and without it. NQ shows a more significant difference, though, which demonstrates it is still a useful addition. We also find that the generative distillation loss (as opposed to standard next-token prediction) is important for high-quality generations. Doing self-supervised pretraining and leveraging harder negatives are also shown to be meaningful factors in the strong performance of the ECG models. Dynamic loss scaling, the learned temperature and teacher score scale for the retrieval losses, is a highly important factor; without it, the performance drops significantly, falling to around half (0.173 versus 0.343) for NQ when $k = 1$. This indicates balancing the two losses and aligning the teacher's scores to the student distribution is crucial for high-quality representations. The exact hard-negative temperature seems to also be highly impactful, with a large degradation in performance for several configurations. Interestingly, we can see that as k increases sometimes there is a large uptick in performance. We hypothesize that the model may have a consistent bias towards certain documents which usually appear in the top spots which pushes relevant information to lower k values. Although the change from the negative temperature is extreme, we suspect this might indicate this parameter is more sensitive to hyperparameter choices and thus might do worse than it would do with tuned hyperparameters.

F On-Device Experiment Details

We benchmarked inference efficiency on a 16 GB Apple M1 Pro MacBook Pro using PyTorch with CPU execution. We evaluated the SmolLM-v2 135M and Gemma 3 1B models with a batch size of one. Each configuration generated 50 new tokens and was evaluated over 10 runs. We varied the input context length over $\{16, 32, 64, 128, 256, 512, 1024\}$ tokens. For

	NQ					TRIVIAQA				
Method	**k=1**	**k=2**	**k=3**	**k=4**	**k=5**	**k=1**	**k=2**	**k=3**	**k=4**	**k=5**
ECG	**0.343**	**0.338**	**0.325**	**0.309**	**0.293**	**0.515**	**0.518**	**0.504**	**0.488**	**0.482**
Ablations										
w/o Contrastive (Pretraining)	0.337	0.335	0.325	0.313	0.295	0.518	0.516	0.503	0.488	0.477
w/o Generative Distillation	0.313	0.313	0.296	0.282	0.274	0.462	0.464	0.451	0.432	0.427
w/o Pretraining	0.300	0.302	0.293	0.276	0.262	0.470	0.480	0.469	0.450	0.430
w/o Loss Scaling	0.173	0.181	0.175	0.166	0.165	0.385	0.420	0.427	0.424	0.425
w/o Hard Negatives (Random)	0.308	0.320	0.314	0.301	0.302	0.514	0.518	0.507	0.493	0.493
w/ Negative Temp 0.5	0.325	0.334	0.318	0.308	0.305	0.051	0.065	0.083	0.442	0.478
w/ Negative Temp 0.25	0.077	0.219	0.308	0.317	0.324	0.042	0.031	0.028	0.025	0.022

Table 4: Ablation study results for ECG on NQ and TriviaQA datasets across different k values.

each context length, we measured time to first token (TTFT) and KV-cache memory usage. TTFT measures the elapsed time required to process the input context and produce the first generated token. We report the mean TTFT and KV-cache memory usage across the 10 runs. All measurements were collected using the same generation length, batch size, execution backend, and set of context lengths to provide a consistent comparison of how increasing context size affects inference latency and memory usage.

G Hyperparameters

This section contains the hyperparameters used for each ECG and baseline model trained (see Tables 5, 6, 7, 8, 9, 10, and 11). Additionally, we include the information on the hyperparameter sweep we conducted for the ECG models and baseline models in Table 12.

Hyperparameter	**SmolLM-v2 (w/ MLPs)**	**SmolLM-v2**	**Gemma 3**
Includes MLPs	✓	✗	✗
Single Batch Size	16	16	6
Gradient Accumulation	1	1	2
Learning Rate	3e-4	3e-4	2e-5
Weight Decay	0.0	0.0	0.0
Train Epochs	1	1	1

Table 5: Hyperparameters for the self-supervised pretraining stage for models without contrastive loss (i.e., with only the generative loss). These models were used to initialize the context compression baseline models including the SmolLM model with the same architecture as the ECG model. The SmolLM model trained with MLPs was also used for the ECG ablation which does not use contrastive loss for pretraining.

Hyperparameter	**SmolLM-v2**	**Gemma 3**
Single Batch Size	16	6
Gradient Accumulation	1	2
Learning Rate	4e-4	9e-5
Weight Decay	0.0	0.0
Train Epochs	1	1

Table 6: Hyperparameters for the self-supervised pretraining stage for models with contrastive loss. These models were used as the starting checkpoints for RAG training for the ECG models including the ablation ECG models if they did not use a different pretraining strategy.

Hyperparameter	SmolLM-v2	Gemma 3
Single Batch Size	16	8
Gradient Accumulation	1	2
Learning Rate	1e-4	5e-6
Weight Decay	0.1	0.1
Train Epochs	10	10

Table 7: Hyperparameters for RAG training for the baseline Reader Models.

Hyperparameter	SmolLM-v2 (w/ MLPs)	SmolLM-v2	Gemma 3
Includes MLPs	✓	✗	✗
Single Batch Size	8	8	3
Gradient Accumulation	2	2	4
Learning Rate	3e-4	3e-4	1e-4
Weight Decay	0.15	0.05	0.05
Train Epochs	20	20	20

Table 8: Hyperparameters for RAG training for the baseline context compression models.

Hyperparameter	SmolLM-v2	Gemma 3
Single Batch Size	16	6
Gradient Accumulation	1	2
Learning Rate	3e-4	9e-5
Weight Decay	0.05	0.0
Train Epochs	20	20

Table 9: Hyperparameters for RAG training for the ECG models. For the ablation models we use the same hyperparameters.

Hyperparameter	SmolLM-v2	Gemma 3
Single Batch Size	16	8
Gradient Accumulation	1	2
Learning Rate	8e-5	7e-6
Weight Decay	0.2	0.0

Table 10: Hyperparameters for RAG training for the baseline parametric models.

Hyperparameter	SmolLM-v2	Gemma 3
Optimizer	AdamW	AdamW
Scheduler	Linear	Linear
Warmup Ratio	0.05	0.05
Negative Temperature	0.15	0.15
# GPUs	8	8

Table 11: Shared hyperparameters for all training runs.

Model Category	Learning Rates (LR) Explored	Weight Decay (WD)
Pretraining (All)	7e-4, 6e-4, 5e-4, 4e-4, 3e-4, 2e-4, 1e-4, 9e-5, 8e-5, 7e-5, 6e-5, 5e-5, 4e-5, 3e-5, 2e-5, 1e-5, 9e-6, 3e-6	0.0, 0.1, 0.2
Reader	4e-4, 3e-4, 2e-4, 1e-4, 9e-5, 8e-5, 7e-5, 6e-5, 5e-5, 4e-5, 3e-5, 2e-5, 1e-5, 9e-6, 8e-6, 7e-6, 6e-6, 5e-6, 4e-6, 3e-6, 2e-6, 1e-6, 9e-7	0.0, 0.1, 0.2
Parametric	4e-4, 3e-4, 2e-4, 1e-4, 9e-5, 8e-5, 7e-5, 6e-5, 5e-5, 4e-5, 3e-5, 2e-5, 1e-5, 9e-6, 8e-6, 7e-6, 6e-6	0.0, 0.1, 0.2
Context Comp.	4e-4, 3e-4, 2e-4, 1e-4, 9e-5, 8e-5, 7e-5, 6e-5, 5e-5, 4e-5	0.0, 0.05, 0.1, 0.15, 0.2
ECG	6e-4, 5e-4, 4e-4, 3e-4, 2e-4, 1e-4, 9e-5, 8e-5, 7e-5, 6e-5, 5e-5, 4e-5, 3e-5, 2e-5	0.0, 0.05, 0.1, 0.15, 0.2

Table 12: Hyperparameter Search Space. We performed a grid search over the union of learning rates (LR) and weight decay (WD) values explored for both SmolLM-v2 and Gemma 3 architectures. Note that for each model we started with an expected reasonable learning rate based on initial experiments and continued the search until the best LR was neither the largest nor the smallest LR tested; thus, each model was not necessarily evaluated at every LR value
. This is also why some models have substantially different starting and ending points.

H Prompts

Reader Model Prompt

Generate a response to the following question using the context.
Question: {question}
Context: {context text}
Response:

Figure 4: Prompt used for standard RAG reader model. Note that context text was separated by two new lines.

Parametric Model Prompt

Generate a response to the following question.
Question: {question}
Response:

Figure 5: Prompt used for parametric (no retrieval) model.

Encoding/Compression Prompt

Encode the following text into embedding tokens. {text} <emb_start> <emb> ··· <emb> <emb_stop>

Figure 6: Prompt used for encoding/compression for the ECG and context compression models.

Generation Prompt from Compressed Context

Generate a response to the following question using the embedded context.
Question: {question}
Context: {compressed context vectors}
Response:

Figure 7: Prompt used for generating with compressed context representations. This is used for both the context compression and ECG models.