Hybrid-Vector Retrieval for Visually Rich Documents: Combining Single-Vector Efficiency and Multi-Vector Accuracy

Juyeon Kim[1] **Geon Lee**[1] **Dongwon Choi**[1] **Taeuk Kim**[2*] **Kijung Shin**[1*]
[1]KAIST [2]Hanyang University
{juyeonkim, geonlee0325, cookie000215, kijungs}@kaist.ac.kr
kimtaeuk@hanyang.ac.kr

Abstract

Retrieval over visually rich documents is essential for tasks such as legal discovery, scientific search, and enterprise knowledge management. Existing approaches fall into two paradigms: *single-vector retrieval*, which is efficient but coarse, and *multi-vector retrieval*, which is accurate but computationally expensive. To address this trade-off, we propose HEAVEN, a plug-and-play two-stage hybrid-vector framework. In the first stage, HEAVEN efficiently retrieves candidate pages using a single-vector method over Visually-Summarized Pages (VS-Pages), which assemble representative visual layouts from multiple pages. In the second stage, it reranks candidates with a multi-vector method while filtering query tokens by linguistic importance to reduce redundant computations. To evaluate retrieval systems under realistic conditions, we also introduce VIMDOC, a benchmark for *visually rich*, *multi-document*, and *long-document* retrieval. Across four benchmarks, HEAVEN attains 99.87% of the Recall@1 performance of multi-vector models on average while reducing per-query computation by 99.82%, achieving efficiency and accuracy. Our code and datasets are available at: https://github.com/juyeonnn/HEAVEN

1 Introduction

Document retrieval aims to retrieve relevant document pages from a corpus for a given query, with broad applications including legal discovery, scientific literature search, and enterprise knowledge management. With the rise of large language models (LLMs), it has gained renewed attention as a core component of Retrieval-Augmented Generation (RAG), which grounds model responses in retrieved evidence to enhance factual reliability.

While traditional document retrieval methods have primarily relied on text representations, many real-world pages contain visually complex elements, such as charts, tables, and figures, that are crucial for answering queries, motivating the task of visual document retrieval (VDR). To process such content, optical character recognition (OCR) or complex layout parsing has been employed, which increases indexing time and complexity. Recently, Large Vision-Language Models (LVLMs) have enabled directly encoding each page as an image to obtain visual embeddings, simplifying the retrieval pipeline and improving performance on visually rich documents (Faysse et al., 2025).

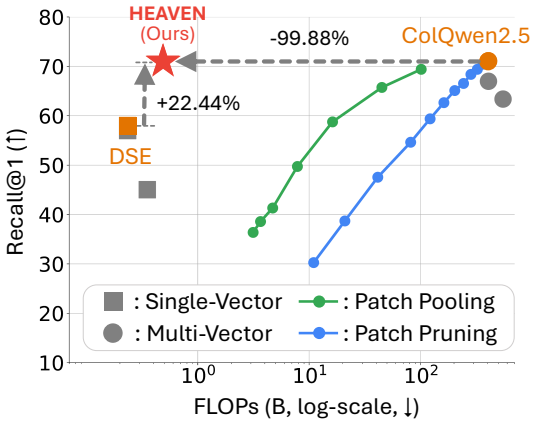

Figure 1: Performance comparison of HEAVEN with (1) single-vector models, (2) multi-vector models, and (3) efficiency-oriented variants of multi-vector models (patch pooling and pruning). HEAVEN yields the best trade-off between efficiency and accuracy on the VIMDOC benchmark. Refer to Section 6.3 for details.

Modern VDR methods fall into two paradigms: single-vector and multi-vector retrieval. *Single-vector retrieval* encodes a query and a document page into single embeddings, enabling efficient similarity computation via a dot product (Yu et al., 2024; Ma et al., 2024a). In contrast, *multi-vector retrieval* encodes them into multiple token- or patch-level embeddings and computes fine-grained interactions across all query–page vector pairs (Faysse et al., 2025; Xu et al., 2025; Xiao et al., 2025).

*Co-corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2026, pages 1073–1089
July 2-7, 2026

Due to their design differences, single-vector and multi-vector retrieval methods exhibit a clear trade-off between efficiency and accuracy. While single-vector retrieval methods (e.g., DSE (Ma et al., 2024a)) are highly efficient but less accurate, multi-vector retrieval methods (e.g., ColQwen 2.5 (Faysse et al., 2025)) achieve higher accuracy at a substantially greater computational cost.

To address this efficiency-accuracy trade-off, we propose HEAVEN (**H**ybrid-vector retrieval for **E**fficient and **A**ccurate **V**isual multi-docum**EN**t), a hybrid framework that combines the efficiency of single-vector retrieval with the accuracy of multi-vector retrieval. Specifically, HEAVEN consists of two stages:

- **(Stage 1) Single-Vector Retrieval of Candidate Pages**: Filtering is first performed at the level of our proposed *visually-summarized pages*, which aggregate key visual elements across multiple pages, before applying filtering at the page level.
- **(Stage 2) Multi-Vector Reranking of Pages**: Candidate pages are reranked using only filtered query tokens, *key tokens*, reducing computation while preserving accuracy.

As shown in Figure 1, HEAVEN provides a significantly improved efficiency–accuracy trade-off. Furthermore, HEAVEN is a plug-and-play framework. Its modular design allows the base encoder at each stage to be selected or swapped separately without further training, making it seamlessly adaptable to specific domains or easily upgraded to higher-performing models.

In addition, we present ViMDoc (**Vi**sually-rich Long **M**ulti-**Doc**ument Retrieval Benchmark), a new benchmark for evaluating visual document retrieval under both multi-document and long-document settings. Most existing VDR benchmarks either assume that queries can be resolved within a single document or focus on short documents. However, real-world applications often require retrieval across massive, multi-document collections where individual documents often span dozens of pages. ViMDoc addresses this gap.

Our contributions are summarized as follows:

- **Method.** We introduce HEAVEN, a plug-and-play two-stage hybrid-vector retrieval framework that effectively addresses the efficiency-accuracy trade-off in visual document retrieval.

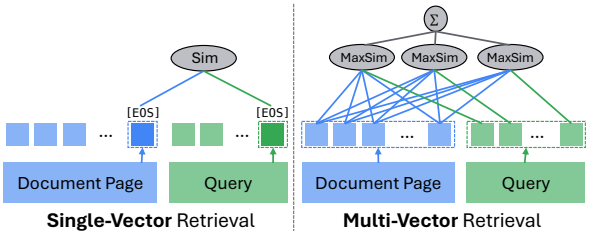

Figure 2: Single- and multi-vector retrieval models.

- **Benchmark.** We propose ViMDoc, a benchmark for visually rich, long-context, and multi-document retrieval.
- **Experiments.** HEAVEN preserves 99.87% of the state-of-the-art retrieval performance of the multi-vector model ColQwen2.5 (in terms of Recall@1) while reducing per-query FLOPs by 99.82% across four multi-document benchmarks.

2 Preliminaries

We define the problem of document retrieval and review two main paradigms for addressing it.

2.1 Problem Definition

Let $\mathcal{D} = \{D_1, D_2, \ldots, D_{|\mathcal{D}|}\}$ denote a collection of documents. Each document $D_k \in \mathcal{D}$ is represented as an ordered sequence of pages, $D_k = (P_{k,1}, P_{k,2}, \ldots, P_{k,|D_k|})$, where $P_{k,i}$ denotes the i-th page of the document D_k. Let $\mathcal{P} = \bigcup_{D_k \in \mathcal{D}} D_k$ denote the set of all pages across the corpus.

We focus on *page-level retrieval*. Given a query q, the goal is to find a set of ground-truth pages $\mathcal{P}_q \subseteq \mathcal{P}$, where the pages in $\mathcal{P}_q$ *collectively* provide the information required to answer the query.

2.2 Retrieval Frameworks

A typical retrieval system represents a query q and a page P as embeddings in a shared latent space (see Figure 2). Specifically, the query is represented as $\mathbf{E}_q \in \mathbb{R}^{n_q \times d}$ and the page as $\mathbf{E}_P \in \mathbb{R}^{n_P \times d}$, where n_q and n_P denote the number of vectors (e.g., tokens or patches) used to represent the query and page, respectively, and d is the embedding dimension. The system ranks pages by the relevance score using $\mathbf{E}_q$ and $\mathbf{E}_P$ and outputs the top-K results.

Single-Vector Retrieval. In single-vector retrieval, both queries and pages are represented by single embeddings rather than all token-level representations. Typically, this vector is obtained from a special token (e.g., an [EOS] token) or by aggregating over all token vectors (e.g., mean pooling). The

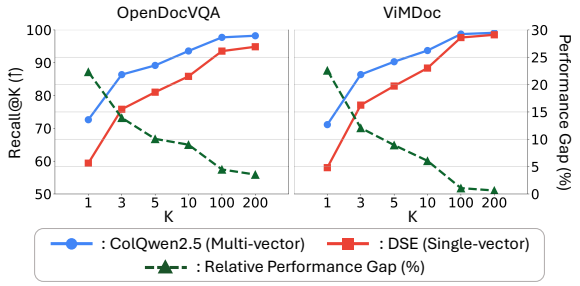

Figure 3: Multi-vector models (e.g., ColQwen2.5) outperform single-vector models (e.g., DSE), with pronounced gaps at fine-grained retrieval (K=1) but much smaller gaps at coarse-grained retrieval (K=200).

relevance score is computed as the similarity (e.g., dot product) between the query and page vectors:

$$S_{\text{SV}}(q, P) = \langle \tilde{\mathbf{E}}_q, \tilde{\mathbf{E}}_P \rangle,$$

where $\tilde{\mathbf{E}}_q, \tilde{\mathbf{E}}_P \in \mathbb{R}^d$ denote the single-vector representations of the query q and the page P, respectively. For each query, a single dot product is computed per page, resulting in $O(d|\mathcal{P}|)$ time to score all pages in the corpus.

Multi-Vector Retrieval. In multi-vector retrieval, the relevance score between a query and a page is computed by aggregating fine-grained interactions between all pairs of their embeddings:

$$S_{\text{MV}}(q, P) = \sum_{i=1}^{n_q} \max_{j \in \{1,\cdots,n_P\}} \langle \mathbf{E}_q^{(i)}, \mathbf{E}_P^{(j)} \rangle,$$

where $\mathbf{E}_q^{(i)}$ and $\mathbf{E}_P^{(j)}$ denote the i-th and j-th row vectors of $\mathbf{E}_q$ and $\mathbf{E}_P$, respectively. Intuitively, this formulation finds, for each query token, the most relevant patch in the page and then sums these maximum similarities to produce the final score. For a query q, scoring a page P requires $O(d\, n_q\, n_P)$ time, and thus scoring all pages requires $O(d\, n_q \sum_{P \in \mathcal{P}} n_P)$ time.

3 Analysis

We empirically compare the two retrieval frameworks to examine their trade-offs. As shown in Figures 1 and 3, we present two key observations:

(Obs. 1) Single-Vector Retrieval is More Efficient Than Multi-Vector Retrieval. Single-vector retrieval computes only one dot product per query-page pair, while multi-vector retrieval requires multiple comparisons. As shown in Figure 1, single-vector retrieval is notably more efficient, requiring up to 99.94% fewer FLOPs per query in ViMDoc, making it scalable for large corpora.

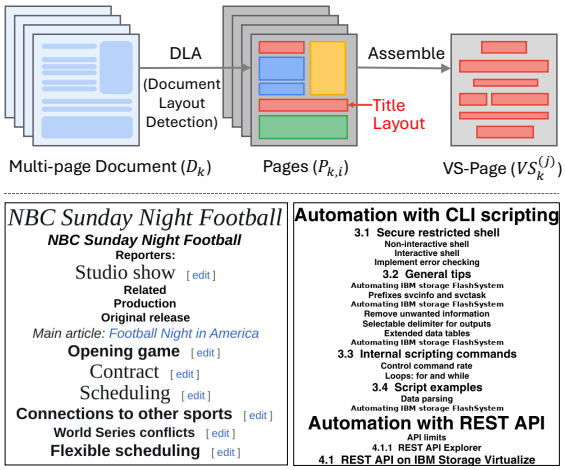

Figure 4: **Top:** Visually-summarized pages (VS-pages) construction process. **Bottom:** Example outputs. More examples are provided in Appendix A.2.

(Obs. 2) Single-Vector Retrieval can be Acceptable for Coarse-Grained Retrieval. As shown in Figure 3, single-vector retrieval is generally less accurate than multi-vector retrieval, as it cannot capture fine-grained token/patch–level interactions. However, this gap narrows as more candidates (larger top-K) are considered. For instance, the performance gap is 22.5% at Recall@1 but only 0.63% at Recall@200 in ViMDoc. Thus, while multi-vector retrieval is superior for precise matching, single-vector retrieval can be sufficient for coarse-grained retrieval with broader candidate sets.

4 Proposed Method: HEAVEN

Motivated by our observations (Section 3), we propose HEAVEN (Figure 5), a two-stage hybrid-vector framework combining the *efficiency* of single-vector and the *accuracy* of multi-vector retrieval for visual document retrieval. Additionally, HEAVEN is plug-and-play. The single-vector and multi-vector models at each stage can be selected or replaced without further training, making it seamlessly adaptable to specific domains or easily upgradeable to stronger models.

4.1 Stage 1. Single-Vector Retrieval of Candidate Pages

In the first stage, HEAVEN leverages our observation that single-vector retrieval is both efficient and sufficiently accurate for coarse-grained retrieval. Thus, we use it to rapidly select a subset of candidate pages from a large document corpus.

Despite its efficiency, existing single-vector retrieval models (e.g., DSE) compute similarities

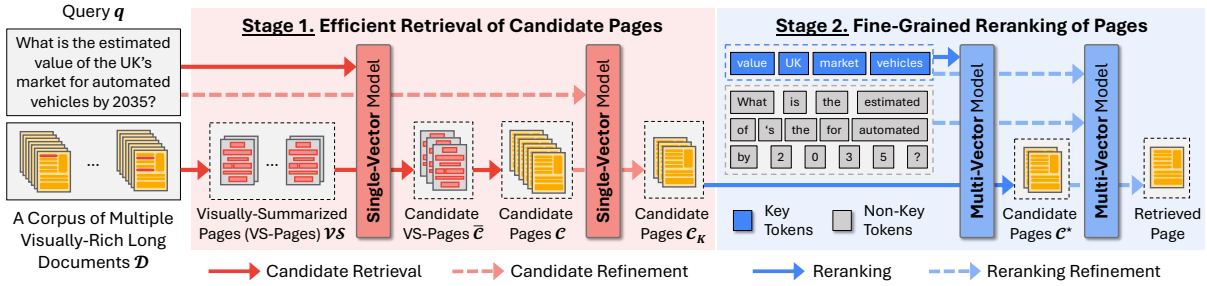

Figure 5: Overall pipeline of HEAVEN. **(Stage 1)** Efficiently retrieves coarse candidate pages via single-vector retrieval, enhanced with visually-summarized pages (VS-pages). **(Stage 2)** Refines and reranks these candidates using a multi-vector model, enhanced with filtered key query tokens, for fine-grained retrieval.

between the query and every page in the corpus, which scales linearly with the total number of pages, i.e., $|\mathcal{P}|$. However, most pages are irrelevant to the query, making computation over the entire corpus redundant. Furthermore, many pages contain repetitive or uninformative content (e.g., recurring logos, headers, or boilerplate text), while only a few elements are actually informative and important for retrieving relevant pages to the query.

Visually-Summarized Pages (VS-pages). To reduce this redundancy, we introduce *visually-summarized pages* (*VS-pages*). As illustrated in Figure 4, multiple pages in a document are compressed into a smaller number of VS-pages, each summarizing several pages by cropping representative layouts (specifically, title layouts) and assembling them into a single page.

VS-page Construction (Indexing Phase Only). Given a document $D_k = (P_{k,1}, P_{k,2}, \ldots, P_{k,|D_k|})$, we first apply Document Layout Analysis (DLA) to each page $P_{k,i} \in D_k$ to extract its title layouts:

$$T_{k,i} = \mathsf{DLA}(P_{k,i}) = \{t_{k,i}^{(1)}, t_{k,i}^{(2)}, \ldots, t_{k,i}^{(|T_{k,i}|)}\},$$

where $t_{k,i}^{(j)}$ is the j-th title layout extracted from page $P_{k,i}$. Let $T_k = \bigcup_{i=1}^{|D_k|} T_{k,i}$ denote the set of layouts aggregated from all pages of document D_k.

Then, we partition T_k into $\lceil |D_k|/r \rceil$ groups, where r is a predefined reduction factor that controls the number of consecutive pages summarized by each VS-page. Each group $T_k^{(j)}$ thus contains approximately $|T_k| \cdot r/|D_k|$ title layouts. The j-th VS-page of D_k is defined as:

$$\mathrm{VS}_k^{(j)} = \mathsf{Assemble}\big(T_k^{(j)}\big), \quad j = 1, \ldots, \lceil |D_k|/r \rceil,$$

where $\mathsf{Assemble}(\cdot)$ is a function that composes the layouts into a single page. Finally, the set of VS-pages for D_k is defined as:

$$\mathrm{VS}_k = \{\mathrm{VS}_k^{(1)}, \ldots, \mathrm{VS}_k^{(\lceil |D_k|/r \rceil)}\}.$$

Let $\mathcal{VS} = \bigcup_{D_k \in \mathcal{D}} \mathrm{VS}_k$ denote the set of all VS-pages across the corpus. Since each VS-page summarizes multiple pages, $|\mathcal{VS}| < |\mathcal{P}|$ holds. We define $\Gamma(\mathrm{VS}_k^{(j)}) = \{P_{k,i} \in \mathcal{P} : T_{k,i} \cap T_k^{(j)} \neq \varnothing\}$ as the set of pages associated with a VS-page $\mathrm{VS}_k^{(j)}$.

Importantly, VS-page construction is performed only once at indexing time, and thus does not add additional overhead during query-time inference. Further details are provided in Appendix A.1.

Candidate VS-page Retrieval. Given a query q, we compute similarities with the constructed VS-pages (fewer than the pages). A single-vector retrieval model scores each $\mathrm{VS} \in \mathcal{VS}$ as:

$$S_{\mathrm{SV}}(q, \mathrm{VS}) = \langle \tilde{\mathbf{E}}_q, \tilde{\mathbf{E}}_{\mathrm{VS}} \rangle, \quad \forall\, \mathrm{VS} \in \mathcal{VS},$$

where $\tilde{\mathbf{E}}_q, \tilde{\mathbf{E}}_{\mathrm{VS}} \in \mathbb{R}^d$ are the single-vector representation of the query and the VS-page, respectively. Then, we rank all VS-pages based on their scores and retain the top-$(p_1 \times 100)\%$ as candidates, denoted by $\overline{\mathcal{C}}$, where p_1 is a hyperparameter.

Candidate Page Refinement. From the candidate VS-pages $\overline{\mathcal{C}} \subset \mathcal{VS}$, we expand to their associated pages, i.e., $\mathcal{C} = \bigcup_{\mathrm{VS} \in \overline{\mathcal{C}}} \Gamma(\mathrm{VS})$. To refine these page candidates, we integrate both VS-page-level and page-level scores. For each candidate page $P \in \mathcal{C}$, the combined score is defined as:

$$S_{\mathrm{SV}}^*(q, P) = \alpha S_{\mathrm{SV}}(q, \Gamma^{-1}(P)) + (1-\alpha) S_{\mathrm{SV}}(q, P),$$

where $\Gamma^{-1}(P)$ denotes the VS-page associated with page P, and α is a weighting hyperparameter. This score integrates both the VS-page-level score $S_{\mathrm{SV}}(q, \Gamma^{-1}(P))$ and the page-level score $S_{\mathrm{SV}}(q, P)$. Then, we rank the pages in $\mathcal{C}$ by $S_{\mathrm{SV}}^*(q, P)$ and select the top-K pages as the refined candidate set $\mathcal{C}_K$.

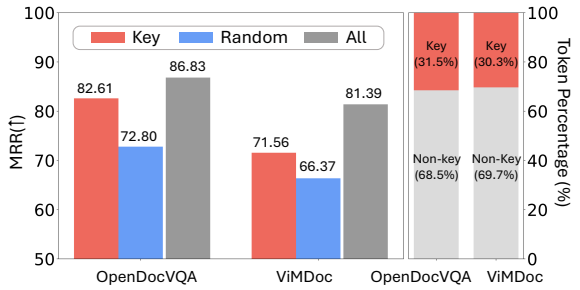

Figure 6: Using only *key tokens* is reasonably effective, despite their small proportion (≈ 30%) within queries.

Benchmarks (Test Split)	Document Page #Total	Avg./Doc	Query #Total	#Cross	Multi-Doc	Long-Doc	Cross-Query
MP-DocVQA	6.2k	6.5	5.0k	-			
VisR-Bench	24.2k	18.5	35.6k	-			
SlideVQA	8.0k	20.0	2.2k	0.6k			✓
MMDocIR	20.4k	65.1	1.7k	0.3k		✓	✓
MMLongBench-Doc	6.5k	47.5	1.1k	0.4k		✓	✓
LongDocURL	33.9k	85.6	2.3k	1.2k		✓	✓
ViDoRE	8.3k	1.0	3.8k	-	✓		
ViDoSeek	5.4k	18.4	1.1k	-	✓		
REAL-MM-RAG	8.6k	52.8	4.6k	-	✓	✓	
M3DocVQA	41.1k	12.2	2.4k	†	✓		✓
VisDoM	21.0k	16.4	2.3k	†	✓		✓
OpenDocVQA	106.7k	3.1	2.5k	0.3k	✓		✓
ViMDoc (Ours)	76.3k	55.4	10.9k	0.7k	✓	✓	✓

† Page-level label not provided.

Table 1: Comparison of VDR datasets. ViMDoc features multiple long documents with cross-page queries.

4.2 Stage 2. Multi-Vector Reranking of Pages

In the second stage, HEAVEN adopts a multi-vector framework to rerank the candidate set $\mathcal{C}_K$ obtained from Stage 1, computing fine-grained similarities across token- or patch-level embeddings of the query and each page for more precise ranking.

However, queries often include tokens that are less informative for the task (e.g., stopwords). Multi-vector models nonetheless compute similarities between all query tokens and all page embeddings, causing redundant computation.

Reranking with Filtered Query Tokens. To reduce redundant computation, we filter query tokens based on their linguistic importance. Given a query q with n_q tokens $q = \{q^{(1)}, q^{(2)}, \ldots, q^{(n_q)}\}$, we apply Part-of-Speech (POS) tagging to identify a subset of *key tokens* $q_{\text{key}} \subseteq q$ (e.g., nouns or named entities), which account for about 30% of tokens on average, as described in Figure 6.

The multi-vector relevance score between a filtered query q_{key} and a candidate page $P \in \mathcal{C}_K$ is:

$$S_{\text{MV}}(q_{\text{key}}, P) = \sum_{i=1}^{|q_{\text{key}}|} \max_{j \in \{1,\ldots,n_P\}} \langle \mathbf{E}^{(i)}_{q_{\text{key}}}, \mathbf{E}^{(j)}_{P} \rangle,$$

where $\mathbf{E}^{(i)}_{q_{\text{key}}}$ and $\mathbf{E}^{(j)}_{P}$ denote the embeddings of the i-th key token and the j-th page patch, respectively. This reduces the similarity computation by a factor of $|q_{\text{key}}|/n_q$. We use these scores to rerank the candidate pages $\mathcal{C}_K$ from Stage 1 and retain the top-$(p_2 \times 100)\%$ as the final candidate set $\mathcal{C}^\star \subseteq \mathcal{C}_K$.

Despite using fewer tokens, it achieves performance comparable to using all tokens and outperforms using the same number of random tokens, as shown in Figure 6. While prior work reduces multi-vector retrieval complexity by pruning or pooling textual tokens (Santhanam et al., 2022a; Clavié et al., 2024) or visual patches (Faysse et al., 2025; Ma et al., 2025; Yan et al., 2025) of the documents, we demonstrate in Section 6 the effectiveness of filtering tokens of the queries instead.

Reranking Refinement. With the reduced set of candidate pages $\mathcal{C}^\star \subset \mathcal{P}$, we now perform a precise reranking using all query tokens, i.e., $S_{\text{MV}}(q, P)$. For each candidate page $P \in \mathcal{C}^\star$, we refine the score by integrating the single-vector score $S^\star_{\text{SV}}(q, P)$ from Stage 1 with the multi-vector score:

$$S^\star_{\text{MV}}(q, P) = \beta\, S^\star_{\text{SV}}(q, P) + (1 - \beta)\, S_{\text{MV}}(q, P),$$

where β is a weighting hyperparameter. Finally, we rank the pages in $\mathcal{C}^\star$ by $S^\star_{\text{MV}}(q, P)$ to obtain the final retrieval results.

5 Proposed Benchmark: ViMDoc

A corpus often contains numerous documents, each spanning multiple pages. Thus, retrieval systems should be evaluated under two realistic settings: (i) the *multi-document* setting, where relevant pages for a query may appear across documents, and (ii) the *long-document* setting, where individual documents are lengthy (e.g., >20 pages). To address this, we propose ViMDoc, a visual document retrieval benchmark that jointly consider both settings.

5.1 Existing Benchmarks

Table 1 shows that existing benchmarks for VDR do not jointly consider both the multi-document and long-context settings. Most focus on retrieving a relevant page from a single gold-standard document, which is often short (e.g., MP-DocVQA (Tito et al., 2023), VisR-Bench (Chen et al., 2025), SlideVQA (Tanaka et al., 2023)) or, in some cases, long but still restricted to a single-document setting (e.g.,

Benchmarks	Avg. Search Space		Corpus
	# Page	# Doc	
VisR-Bench	18.5	1.0	Single-Doc
MMDocIR	65.1	1.0	Single-Doc
MMLongBench-Doc	47.5	1.0	Single-Doc
LongDocURL	85.6	1.0	Single-Doc
REAL-MM-RAG	2151.0	40.8	Multi-Doc
ViMDoc (Ours)	76347.0	1379.0	Multi-Doc

Table 2: Comparison with sourced benchmarks. While most are limited to single-document settings with a small search space, ViMDoc supports retrieval across a larger, multi-document corpus.

MMLongBench-Doc (Ma et al., 2024b), MMDocIR (Dong et al., 2025), LongDocURL (Deng et al., 2025)). Some benchmarks evaluate retrieval across multiple documents, but their documents are generally short, averaging only 1.0 - 18.4 pages (e.g., ViDoRe (Faysse et al., 2025), ViDoSeek (Wang et al., 2025), OpenDocVQA (Tanaka et al., 2025), M3DocVQA (Cho et al., 2025), VisDoM (Suri et al., 2025)). As a result, these benchmarks fail to capture the combined challenge of retrieving relevant information across *multiple long documents*.

5.2 Design of ViMDoc

ViMDoc is designed to evaluate VDR systems under both the *multi-document* and *long-document* settings. It consists of visually rich documents and provides queries with page-level ground-truth annotations, including those with ground truth labels spanning multiple pages (i.e., cross-page queries).

Document Collection. We collect documents that (1) contain visually rich content such as figures, tables, and charts, and (2) consist of many pages, with an average length exceeding 20 pages. Specifically, we include documents from VisR-Bench (Chen et al., 2025), REAL-MM-RAG (Wasserman et al., 2025), MMLongBench-Doc (Ma et al., 2024b), MMDocIR (Dong et al., 2025), and LongDocURL (Deng et al., 2025), resulting in 1,379 documents spanning 76,347 pages. See Appendix B.1 for details.

Query Processing. As shown in Table 2, many sourced queries are designed for single-document settings. Some are therefore *context-dependent*—relying on generic cues (e.g., "what is the title") or positional hints (e.g., "from the last page")—and thus unsuitable for multi-document retrieval. Following Tanaka et al. (2025) and Wang et al. (2025), we retain only *self-contained* queries with distinctive keywords such as named entities or technical terms, enabling retrieval across the union of all document pages. 45.8% of queries identified as *context-dependent* are removed via a two-stage filtering pipeline: (1) heuristic rule-based filtering and (2) LLM-based filtering. The retained queries can thus be evaluated in a multi-document setting with a larger search space than their sourced benchmarks. Details are provided in Appendix B.2.

6 Experimental Results

In this section, we present the overall experimental setup and results, including comparisons with baselines, ablation study, efficiency analysis, hyperparameter analysis, and plug-and-play analysis.

6.1 Settings

Models. Two categories of retrieval models are utilized for evaluation. For single-vector retrieval, visual embedding models including VisRAG (Yu et al., 2024), GME (Zhang et al., 2025) and DSE (Ma et al., 2024a) are utilized, as well as textual embedding models NV-Embed-V2 (Lee et al., 2024) and BGE-M3 (dense) (Chen et al., 2024). For multi-vector retrieval, ColPali, ColQwen2, ColQwen2.5 (Faysse et al., 2025), and BGE-M3 (multi-vec) (Chen et al., 2024) are employed. Additional implementation details and model checkpoints are provided in Appendix C.1.

Datasets. HEAVEN is tested on 4 benchmarks: ViMDoc, OpenDocVQA (Tanaka et al., 2025), ViDoSeek (Wang et al., 2025) and M3DocVQA (Cho et al., 2025). For OpenDocVQA, only the SlideVQA (Tanaka et al., 2023) and DUDE (Van Landeghem et al., 2023) splits are used, as these are the only multi-page document splits. Detailed benchmark statistics are provided in Appendix C.2.

Evaluation Metrics. We evaluate retrieval performance using page-level Recall@{1,3}, except for M3DocVQA, where document-level metrics are used due to missing page-level labels. Efficiency is measured by per-query FLOPs (billions) and latency (seconds). See Appendix C.3 for details.

Implementation Details. HEAVEN uses DSE for Stage 1 and ColQwen2.5 for Stage 2. For document layout analysis, it uses DocLayout-YOLO (Zhao et al., 2024). For each document D_k, the reduction factor r is set to $\min(15, |D_k|)$. Default hyperparameters are $\alpha = 0.1$, $\beta = 0.3$, $p_1 = 0.5$,

		ViMDoc (Proposed)			OpenDocVQA			ViDoSeek			M3DocVQA			AVERAGE		
	Model	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**
Single-Vector	BGE-M3 (dense)	44.89	61.62	0.156	38.67	52.45	0.165	54.82	76.01	0.011	57.56	77.11	0.084	48.99	66.80	0.104
	NV-Embed-V2	50.06	69.08	0.625	47.37	64.59	0.658	66.46	83.89	0.044	62.65	85.77	0.336	56.64	75.83	0.416
	VisRAG	45.03	64.49	0.352	51.41	67.39	0.370	63.31	84.33	0.025	50.39	69.59	0.189	52.53	71.45	0.234
	GME	57.01	76.62	0.235	54.19	71.93	0.247	71.19	89.40	0.017	59.85	77.83	0.126	60.56	78.95	0.156
	DSE	58.03	77.08	0.235	59.38	75.82	0.247	69.53	87.13	0.017	55.14	71.30	0.126	60.52	77.83	0.156
	HEAVEN (only Stage 1)	57.64	76.58	0.134	59.89	76.05	0.147	69.35	87.39	0.010	59.40	73.96	0.026	61.57	78.50	0.079
	(vs. DSE)	99.34%	99.35%	-42.76%	100.87%	100.32%	-40.47%	99.75%	100.30%	-42.40%	107.73%	103.73%	-79.67%	101.74%	100.86%	-49.31%
Multi-Vector	BGE-M3 (multi)	53.60	70.66	5863.387	46.60	62.80	888.097	56.74	76.01	30.777	56.88	76.96	1408.110	53.46	71.61	2047.590
	ColPali	63.38	80.58	669.670	67.50	82.13	665.152	66.29	84.41	57.086	58.01	78.65	398.295	63.79	81.44	447.551
	ColQwen2	66.99	82.51	407.320	72.27	86.14	482.049	75.13	90.63	41.514	59.32	80.69	288.507	68.43	84.99	304.847
	ColQwen2.5	71.13	86.39	407.320	72.63	86.38	482.049	75.57	91.94	41.514	57.99	78.73	288.507	69.33	85.86	304.847
	HEAVEN	71.05	86.41	0.486	71.56	84.53	0.541	75.04	91.33	0.623	59.31	78.66	0.545	69.24	85.23	0.549
	(vs. ColQwen2.5)	99.88%	100.02%	-99.88%	98.52%	97.86%	-99.89%	99.30%	99.33%	-98.50%	102.27%	99.90%	-99.81%	99.87%	99.27%	-99.82%

Table 3: Efficiency–accuracy comparison with various single-vector and multi-vector models for visual document retrieval. We report Recall@{1,3} and per-query FLOPs (billions) for HEAVEN, compared with both single-vector and multi-vector models. The relative performance (%) for Recall is highlighted in blue, and the FLOPs is highlighted in red. indicates a textual embedding model and indicates a visual embedding model.

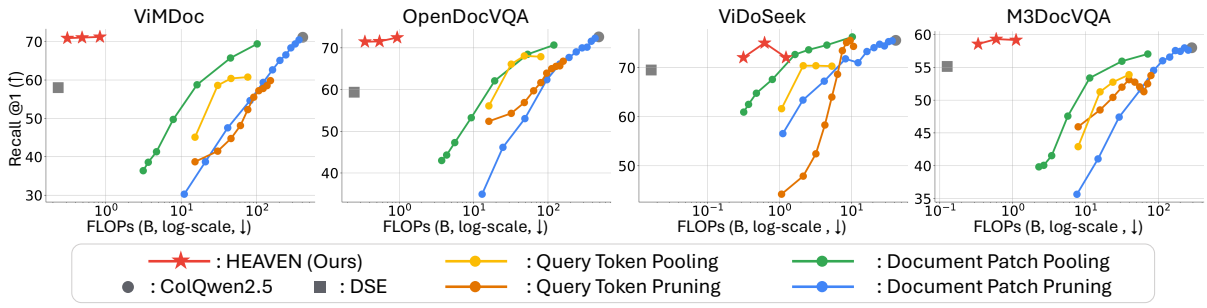

Figure 7: Efficiency-accuracy comparison with four efficiency-oriented variants of the ColQwen2.5-based multi-vector model: (1) **document patch pooling** (Faysse et al., 2025) and (2) **document patch pruning** (Ma et al., 2025; Yan et al., 2025) are applied to document patch embeddings; (3) **query token pooling** and (4) **query token pruning** are applied to query tokens using the same strategies.

$p_2 = 0.25$, and $K = 200$. For M3DocVQA, α and β are set to 0.4 due to document-level evaluation. `nltk` is used for POS tagging for *key token* selection (Appendix A.3), and Tesseract (Smith, 2007) is used to preprocess documents for textual retrieval.

6.2 Main Results

We first evaluate HEAVEN against state-of-the-art textual and visual retrieval models. As shown in Table 3, on average, HEAVEN preserves 99.89% of ColQwen2.5's retrieval performance while reducing FLOPs by 99.82%. In M3DocVQA, its Stage 1 surpasses DSE by 7.73% in Recall@1 with 79.67% fewer FLOPs, and Stage 2 further improves performance by 2.27% while using 99.81% fewer FLOPs than ColQwen2.5. Notably, the clear performance gap between textual and visual methods in ViMDoc and OpenDocVQA reveals the unique challenges of visual document retrieval. Table 12 in Appendix C.5 further reports results separately for cross-page and single-page query types.

6.3 Comparison with Efficiency Variants

We evaluate HEAVEN's efficiency against four approaches designed to enhance efficiency in multi-vector retrieval models: **(1) Document Patch Pooling** (Faysse et al., 2025), which pools adjacent patches within each page; **(2) Document Patch Pruning** (Ma et al., 2025; Yan et al., 2025), which randomly removes patch embeddings; **(3) Query Token Pooling**, which aggregates adjacent special query tokens; and **(4) Query Token Pruning**, which randomly drops them. Each approach is evaluated under varying pooling factors and pruning ratios. Refer to Appendix C.4 for details.

Figure 7 shows that HEAVEN achieves the best efficiency-accuracy trade-off. While pooling generally outperforms pruning, its performance drops

	ViMDoc (Proposed)			OpenDocVQA		
Stage 1	**R@100**	**R@200**	**FLOPs**	**R@100**	**R@200**	**FLOPs**
HEAVEN	97.20	97.96	0.134	93.02	94.59	0.147
w/o VS-pages	97.68	98.51	0.235	93.54	94.86	0.247
w/o refinement	59.47	68.02	0.017	59.47	68.02	0.024
Stage 2	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**
HEAVEN	71.05	86.41	0.486	71.56	84.53	0.541
w/o query filtering	71.08	86.38	0.871	71.76	85.10	0.957
w/o refinement	58.05	77.08	0.358	59.53	75.82	0.402

Table 4: Ablation study of Stage 1 and Stage 2. Each component of HEAVEN is effective for coarse-grained retrieval (Recall@{100,200}) in Stage 1 and fine-grained retrieval (Recall@{1,3}) in Stage 2.

		ViMDoc (Proposed)				
		Offline Indexing (min)			**Online Retrieval (sec, B)**	
		OCR	VS-page	Encode	Latency	FLOPs
Single	NV-Embed-V2	473.0	NA	99.5	0.133	0.625
	DSE	NA	NA	146.7	0.115	0.235
Multi	BGE-M3 (multi)	473.0	NA	32.3	20953.468	5863.387
	ColQwen2.5	NA	NA	129.1	2006.361	407.320
	HEAVEN	NA	58.9	286.4	2.412	0.486
	Stage 1	NA	58.9	157.3	0.079	0.134
	Stage 2	NA	NA	129.1	2.333	0.352

Table 5: Efficiency analysis for offline indexing and online retrieval on ViMDoc. Offline indexing efficiency is reported for the full document corpus, and online retrieval efficiency is reported per query. VS-page construction latency includes DLA and Assemble.

as FLOPs decrease. In contrast, HEAVEN attains higher accuracy with significantly fewer FLOPs.

6.4 Ablation Study

To verify the effectiveness of each component in HEAVEN, we conduct ablation studies comparing it with its variants. For Stage 1, we evaluate two variants, one that omits VS-page construction and another that skips candidate refinement. Table 4 reports that without VS-pages, FLOPs notably increase, as all raw pages are compared, while yielding marginal accuracy gains. Skipping candidate refinement leads to a severe performance drop.

For Stage 2, we examine variants that disable query token filtering and reranking refinement. Without query filtering, FLOPs increase as similarities are computed for all query tokens, including redundant ones. Without reranking refinement, performance degrades significantly. These results confirm the importance of the elements in two stages of HEAVEN for both efficiency and accuracy.

6.5 Efficiency Analysis

To evaluate the efficiency of HEAVEN, we measure two types of latency: **(1) Offline Indexing Latency**, referring to the time required to preprocess documents before retrieval, and **(2) Online Retrieval Latency**, referring to the time required to process a query during retrieval.[1]

As shown in Table 5, HEAVEN requires additional offline processing time for VS-page construction. However, its online retrieval latency is *significantly* lower than that of multi-vector methods, e.g., 99.88% faster than ColQwen2.5 per query. Despite this efficiency, as shown in Table 3, HEAVEN achieves accuracy comparable to multi-vector models while substantially outperforming single-vector ones, confirming its strong trade-off between efficiency and accuracy.

[1]For fair comparison, all experiments are conducted on 16 Intel(R) Xeon(R) Platinum 8480C CPUs and a single NVIDIA H200 GPU. All results are averaged over five runs.

6.6 Hyperparameter Analysis

We analyze the sensitivity of HEAVEN to hyperparameters: the reduction factor r and the filtering ratios for each stage, p_1 and p_2, and the weighting hyperparameters α and β. Figure 8 confirms that HEAVEN remains robust across different filtering ratios in both stages, as well as across varying reduction factor values. Notably, VS-pages are particularly effective in ViMDoc, which consists of long documents. Thus, the impact of the reduction factor is more pronounced than in other benchmarks with shorter documents (e.g., OpenDocVQA). Figure 9 shows that HEAVEN is also robust to the weighting hyperparameters α and β for stage 1 and 2, respectively.

6.7 Plug-and-Play Analysis

HEAVEN is designed to be modular and plug-and-play, allowing the single-vector (Stage 1) and multi-vector (Stage 2) encoders to be replaced with other pretrained models. Table 6 reports results across combinations of two Stage 1 and three Stage 2 base encoders. Among the evaluated combinations, DSE (Stage 1) and ColQwen2.5 (Stage 2) achieve the best overall performance on both ViMDoc and OpenDocVQA, and are therefore adopted as the default configuration.

7 Related Work

Visual Document Retrieval. Visual document retrieval matches queries against document image corpora. Recent methods based on LVLMs (Jiang

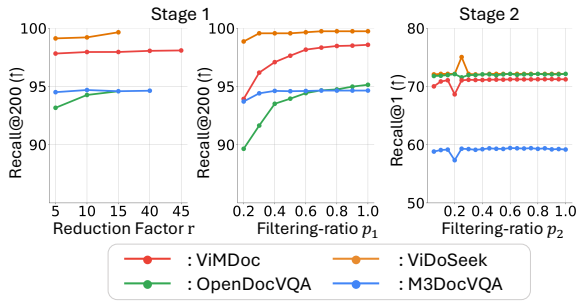

Figure 8: Hyperparameter analysis for reduction factor r, and filtering ratios p_1 and p_2.

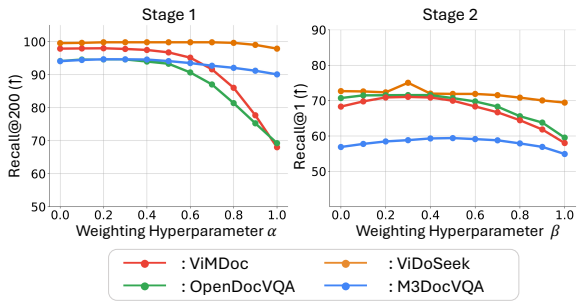

Figure 9: Hyperparameter analysis for weighting hyperparameters α and β.

et al., 2025; Wei et al., 2024; Lin et al., 2025), directly encode entire document pages as images, bypassing OCR (Meng et al., 2025; Zhang et al., 2025; Ma et al., 2024a). Concurrently, visual RAG leverages such visual embeddings, while multimodal approaches combine visual and textual signals using OCR (Yu et al., 2024; Suri et al., 2025; Wang et al., 2025; Sun et al., 2025) or textual summaries from LVLMs and LLMs (Jain et al., 2025; Gong et al., 2025). In contrast, HEAVEN introduces visually-summarized pages (VS-pages), an OCR-free method that enables fully-visual indexing.

Multi-Vector Retrieval. Building on the late-interaction mechanism of ColBERT (Khattab and Zaharia, 2020), recent work has extended multi-vector retrieval to VDR (Faysse et al., 2025; Xu et al., 2025; Xiao et al., 2025). Although accurate, these models incur high computational overhead. Textual domain work addresses this via compression (Santhanam et al., 2022b), approximation (Jayaram et al., 2024), and token reduction via pruning (Santhanam et al., 2022a; Acquavia et al., 2023) or pooling (Clavié et al., 2024), while VDR methods apply patch pruning (Yan et al., 2025) and pooling (Faysse et al., 2025; Ma et al., 2025) to enhance efficiency with minimal loss in accuracy. In contrast, HEAVEN reduces computation by filtering query tokens while maintaining performance.

Base Encoder		ViMDoc (Proposed)			OpenDocVQA		
Stage1	**Stage2**	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**
VisRAG	ColPali	60.95	79.19	1.298	65.27	80.02	1.287
VisRAG	ColQwen2	62.45	80.07	0.553	67.79	81.28	0.615
VisRAG	ColQwen2.5	64.06	82.07	0.553	67.54	81.40	0.615
DSE	ColPali	63.46	80.68	1.231	67.50	81.81	1.214
DSE	ColQwen2	66.63	82.63	0.486	69.91	83.31	0.541
DSE	ColQwen2.5	71.05	86.41	0.486	71.56	84.53	0.541

Table 6: Plug-and-play analysis of HEAVEN with different combinations of Stage 1 and Stage 2 base encoders on ViMDoc and OpenDocVQA. R@{1,3} and per-query FLOPs (billions) are reported.

8 Conclusion

We propose HEAVEN, a plug-and-play hybrid-vector framework that bridges the efficiency-accuracy trade-off in VDR. By combining single-vector retrieval over VS-pages with multi-vector reranking using selective query tokens, HEAVEN achieves near state-of-the-art accuracy with over 99% lower computation. We also present ViMDoc, a benchmark for visually rich, multi-document, and long-document retrieval. Together, they establish a scalable foundation for efficient and accurate VDR.

Limitations

While HEAVEN substantially improves efficiency in visual document retrieval, several limitations remain. First, it relies on pretrained vision-language encoders, and its performance may vary with model scale and domain adaptation quality. Second, the visually-summarized pages used in Stage 1 depend on document layout analysis, which can be sensitive to noisy or irregular layouts. Third, HEAVEN focuses on retrieval efficiency and does not yet integrate retrieval augmentation generation, which we leave as future work.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd. (IO251103-13845-01). This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)).

References

Antonio Acquavia, Craig Macdonald, and Nicola Tonellotto. 2023. Static pruning for multi-representation dense retrieval. In *DocEng*.

Jian Chen, Ming Li, Jihyung Kil, Chenguang Wang, Tong Yu, Ryan Rossi, Tianyi Zhou, Changyou Chen, and Ruiyi Zhang. 2025. Visr-bench: An empirical study on visual retrieval-augmented generation for multilingual long document understanding. *arXiv preprint arXiv:2508.07493*.

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

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. 2025. M3docrag: Multi-modal retrieval is what you need for multi-page multi-document understanding. In *ICCV Workshop*.

Benjamin Clavié, Antoine Chaffin, and Griffin Adams. 2024. Reducing the footprint of multi-vector retrieval with minimal performance impact via token pooling. *arXiv preprint arXiv:2409.14683*.

Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, and 1 others. 2025. Longdocurl: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating. In *ACL*.

Kuicai Dong, Yujing Chang, Xin Deik Goh, Dexun Li, Ruiming Tang, and Yong Liu. 2025. Mmdocir: Benchmarking multi-modal retrieval for long documents. In *EMNLP*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, CELINE HUDELOT, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. In *ICLR*.

Ziyu Gong, Yihua Huang, and Chengcheng Mai. 2025. Mmrag-docqa: A multi-modal retrieval-augmented generation method for document question-answering with hierarchical index and multi-granularity retrieval. *arXiv preprint arXiv:2508.00579*.

Chelsi Jain, Yiran Wu, Yifan Zeng, Jiale Liu, Zhenwen Shao, Qingyun Wu, Huazheng Wang, and 1 others. 2025. Simpledoc: Multi-modal document understanding with dual-cue page retrieval and iterative refinement. In *EMNLP*.

Rajesh Jayaram, Laxman Dhulipala, Majid Hadian, Jason D Lee, and Vahab Mirrokni. 2024. Muvera: Multi-vector retrieval via fixed dimensional encoding. In *NeurIPS*.

Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. 2025. Vlm2vec: Training vision-language models for massive multimodal embedding tasks. In *ICLR*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *SIGIR*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Sheng-Chieh Lin, Chankyu Lee, Mohammad Shoeybi, Jimmy Lin, Bryan Catanzaro, and Wei Ping. 2025. Mm-embed: Universal multimodal retrieval with multimodal llms. In *ICLR*.

Xueguang Ma, Sheng-Chieh Lin, Minghan Li, Wenhu Chen, and Jimmy Lin. 2024a. Unifying multimodal retrieval via document screenshot embedding. In *EMNLP*.

Yubo Ma, Jinsong Li, Yuhang Zang, Xiaobao Wu, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Jiaqi Wang, Yixin Cao, and 1 others. 2025. Towards storage-efficient visual document retrieval: An empirical study on reducing patch-level embeddings. In *ACL (Findings)*.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, and 1 others. 2024b. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. In *NeurIPS*.

Rui Meng, Ziyan Jiang, Ye Liu, Mingyi Su, Xinyi Yang, Yuepeng Fu, Can Qin, Zeyuan Chen, Ran Xu, Caiming Xiong, and 1 others. 2025. Vlm2vec-v2: Advancing multimodal embedding for videos, images, and visual documents. *arXiv preprint arXiv:2507.04590*.

Jingfen Qiao, Jia-Huei Ju, Xinyu Ma, Evangelos Kanoulas, and Andrew Yates. 2025. Reproducibility, replicability, and insights into visual document retrieval with late interaction. In *SIGIR*.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. Plaid: an efficient engine for late interaction retrieval. In *CIKM*.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *NAACL*.

Ray Smith. 2007. An overview of the tesseract ocr engine. In *ICDAR*.

Hao Sun, Yingyan Hou, Jiayan Guo, Bo Wang, Chunyu Yang, Jinsong Ni, and Yan Zhang. 2025. Unveil: Unified visual-textual integration and distillation for multi-modal document retrieval. In *ACL*.

Manan Suri, Puneet Mathur, Franck Dernoncourt, Kanika Goswami, Ryan A Rossi, and Dinesh Manocha. 2025. Visdom: Multi-document qa with visually rich elements using multimodal retrieval-augmented generation. In *NAACL*.

Ryota Tanaka, Taichi Iki, Taku Hasegawa, Kyosuke Nishida, Kuniko Saito, and Jun Suzuki. 2025. Vdocrag: Retrieval-augmented generation over visually-rich documents. In *CVPR*.

Ryota Tanaka, Kyosuke Nishida, Kosuke Nishida, Taku Hasegawa, Itsumi Saito, and Kuniko Saito. 2023. Slidevqa: A dataset for document visual question answering on multiple images. In *AAAI*.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2023. Hierarchical multimodal transformers for multipage docvqa. *Pattern Recognition*.

Jordy Van Landeghem, Rubèn Tito, Łukasz Borchmann, Michał Pietruszka, Pawel Joziak, Rafal Powalski, Dawid Jurkiewicz, Mickaël Coustaty, Bertrand Anckaert, Ernest Valveny, and 1 others. 2023. Document understanding dataset and evaluation (dude). In *ICCV*.

Qiuchen Wang, Ruixue Ding, Zehui Chen, Weiqi Wu, Shihang Wang, Pengjun Xie, and Feng Zhao. 2025. Vidorag: Visual document retrieval-augmented generation via dynamic iterative reasoning agents. In *EMNLP*.

Navve Wasserman, Roi Pony, Oshri Naparstek, Adi Raz Goldfarb, Eli Schwartz, Udi Barzelay, and Leonid Karlinsky. 2025. Real-mm-rag: A real-world multimodal retrieval benchmark. In *ACL*.

Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. 2024. Uniir: Training and benchmarking universal multimodal information retrievers. In *ECCV*.

Zilin Xiao, Qi Ma, Mengting Gu, Chun-cheng Jason Chen, Xintao Chen, Vicente Ordonez, and Vijai Mohan. 2025. Metaembed: Scaling multimodal retrieval at test-time with flexible late interaction. *arXiv preprint arXiv:2509.18095*.

Mengyao Xu, Gabriel Moreira, Ronay Ak, Radek Osmulski, Yauhen Babakhin, Zhiding Yu, Benedikt Schifferer, and Even Oldridge. 2025. Llama nemoretriever colembed: Top-performing text-image retrieval model. *arXiv preprint arXiv:2507.05513*.

Yibo Yan, Guangwei Xu, Xin Zou, Shuliang Liu, James Kwok, and Xuming Hu. 2025. Docpruner: A storage-efficient framework for multi-vector visual document retrieval via adaptive patch-level embedding pruning. *arXiv preprint arXiv:2509.23883*.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and 1 others. 2024. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. In *ICLR*.

Xin Zhang, Yanzhao Zhang, Wen Xie, Mingxin Li, Ziqi Dai, Dingkun Long, Pengjun Xie, Meishan Zhang, Wenjie Li, and Min Zhang. 2025. Bridging modalities: Improving universal multimodal retrieval by multimodal large language models. In *CVPR*.

Zhiyuan Zhao, Hengrui Kang, Bin Wang, and Conghui He. 2024. Doclayout-yolo: Enhancing document layout analysis through diverse synthetic data and global-to-local adaptive perception. *arXiv preprint arXiv:2410.12628*.

Appendix

A HEAVEN

A.1 Detailed VS-page Construction

For Document Layout Detection (DLA), we utilized DocLayout-YOLO (Zhao et al., 2024)[2], which classifies layouts into 8 categories: title, plain text, abandon, figure, figure caption, table, table caption, isolated formula, and formula caption. To construct the VS-page, only the title layouts were extracted by cropping each detected title region according to its bounding box. Following extraction, the cropped title images were assembled into a single page via vertical stacking, a process defined as $\mathsf{Assemble}(\cdot)$. No resizing or scaling operations were applied to preserve the original visual context.

A.2 VS-page Examples

In this section, we further provide more examples of the constructed VS-page. Figure 10, 11, 12, and 13 show example output across the benchmarks.

A.3 Detailed Query *Key Token* Filtering

Part-of-Speech tagging is used to identify the subset of *key tokens*. `nltk`[3] is used for Part-of-Speech(POS) tagging. Using the tokenized query from the model, tokens with these four tags are filtered as *key tokens*: NN, NNS, NNP, and NNPS.

A.4 *Key Token* Distribution

Table 7 details the token distribution per benchmark. *Key tokens* comprise only 30-37% of all query tokens, with the remainder being non-*key tokens*.

Benchmarks	#Avg. *key*	(%)	#Avg. non-*key*	(%)
ViMDoc	6.8 ± 4.1	(30.32%)	15.5 ± 7.9	(69.68%)
OpenDocVQA	6.6 ± 3.4	(31.51%)	14.2 ± 7.6	(68.49%)
ViDoSeek	11.2 ± 3.9	(37.48%)	18.7 ± 5.8	(62.52%)
M3DocVQA	9.1 ± 5.2	(34.63%)	17.2 ± 8.6	(65.37%)
Average	7.4 ± 4.5	(31.83%)	15.9 ± 7.9	(68.17%)

Table 7: Average number and percentage of *key tokens* and non-*key tokens* across benchmarks.

[2] `juliozhao/DocLayout-YOLO-DocStructBenct/doclayout_yolo_docstructbench_imgsz1024.pt`

[3] We use `nltk` 3.9.2 version.

B ViMDoc Benchmark

B.1 Detailed Data Collection

In this section, we provide details of the data collection process.

- REAL-MM-RAG (Wasserman et al., 2025): All four splits (FinReport, FinSlides, TechSlides, and TechReport) were included.
- VisR-Bench (Chen et al., 2025): Only English splits were retained, excluding the Multilingual split, since baseline models were pre-trained mainly on English datasets. Within the English split, all three query types (figure, table, and text) were included.
- MMDocIR (Dong et al., 2025): Only the evaluation set was used.
- LongDocURL (Deng et al., 2025): All documents included in the paper were used.
- MMLongBench-Doc (Ma et al., 2024b): Similar to LongDocURL, all documents were included.

B.2 Detailed Query Processing

Table 8 shows that the two-stage filtering removes 45.8% of queries, retaining only *self-contained* ones suitable for multi-document retrieval setting. **Heuristic rule-based filtering** removes 'Unanswerable' queries from LongDocURL (Deng et al., 2025) and MMLongBench-Doc (Ma et al., 2024b) that are not associated with any page, and queries that explicitly reference 'Table N' or 'Figure N', which are inherently *context-dependent*. However, some *context-dependent* queries lack explicit keywords and are thus difficult to catch with heuristic rules alone. We therefore apply **LLM-based filtering** to remove such cases. Figure 14 shows the prompt used for LLM-based filtering following (Wang et al., 2025). We employ `gpt-5-mini` for LLM-based filtering.

	#Avg. Page	#Doc	#Query (%Filtered)
ViMDoc (Proposed)	**55.4**	**1379**	**10904** (-45.8%)
REAL-MM-RAG	52.8	162	3939 (-13.5%)
VisR-Bench	18.5	373	5142 (-50.9%)
MMDocIR	65.1	313	651 (-60.7%)
LongDocURL	85.6	396	890 (-61.7%)
MMLongBench-Doc	47.5	135	282 (-73.9%)

Table 8: Dataset splits included in ViMDoc.

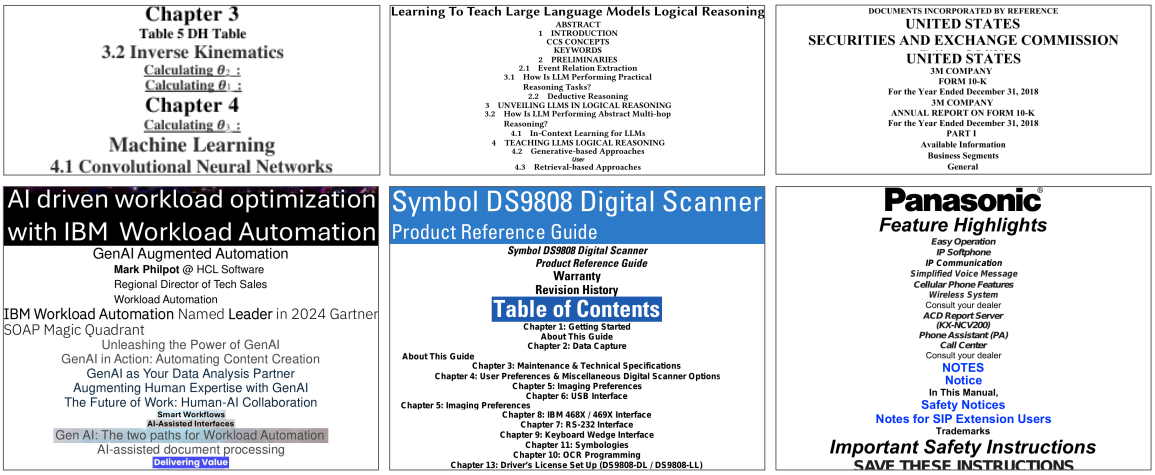

Figure 10: VS-page examples from ViMDoc.

Figure 11: VS-page examples from OpenDocVQA.

Figure 12: VS-page examples from ViDoSeek.

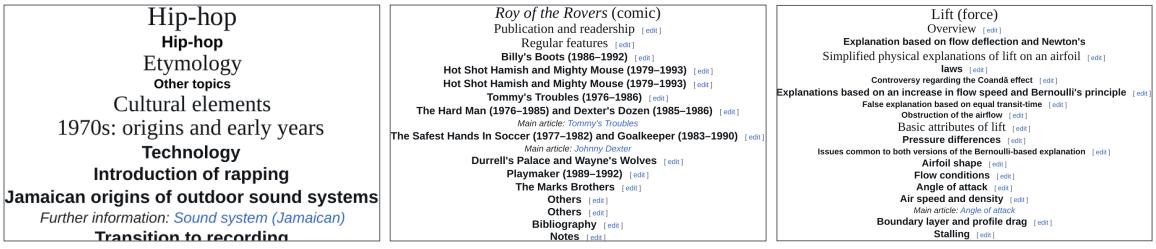

Figure 13: VS-page examples from M3DocVQA.

Prompt Template

"system" :
I have some QA data here, and you can observe that the questions can be divided into two categories:
Category A: Self-Sufficient Queries
- When you see this question alone without a given document, you are sure to find a unique document in a corpus to provide a unique answer.
- The question has key words that help you locate the document from the corpus.
Category B: Context-Dependent Queries
- When you see this question alone without a given document, you will find it hard to locate a document to give a deterministic answer.
- You will find multiple candidate documents in a corpus, which may lead to different answers for this question.
They reference generic document parts (e.g., 'the title', 'the author'), use positional descriptors (e.g., 'the first table', 'the last page'), or ask broad questions without specific search terms (e.g., 'What is the conclusion?', 'How many tables are in this document?').
- The question does not have special key words to help you locate the document from the corpus.

Category A Examples
- What is the full form of PUF?
- Who presented the results on cabin air quality study in commercial aircraft?
- who were bothered by cigarette odors?
- which cigarette would be better if offered on a thicker cigarette?
- Cigarettes will be produced and submitted to O/C Panel for what purpose?
- What is RIP-6 value for KOOL KS?
- Which test is used to evaluate ART menthol levels that has been shipped?
- How much percent had not noticed any difference in the odor of VSSS?
- What is the cigarette code of RIP-6(W/O Filter) 21/4SE?
- what mm Marlboro Menthol were subjectively smoked by the Richmond Panel?
- What are the steps of Weft Preparation between Spinning bobbin and Weaving?
- What level comes between Middle Managers and Non-managerial Employees?
- What are the six parts of COLLABORATION MODEL of the organization where James has a role of leading the UK digital strategy?

Category B Examples
- The number mentioned on the right of the leftside margin?
- What is the date mentioned in the second table?
- What is the number at the bottom of the page, in bold?
- What is the name of the corporation?
- Which part of Virginia is this letter sent from?
- What is the heading of first table?"""

"user" :
Classify the following query as category A or B.
Query: {query}

Figure 14: Prompt template for Query Filtering. Modified from Quality Reviewer prompt in (Wang et al., 2025).

C Experimental Details

C.1 Models

Table 9 shows embedding dimension, base model, batch size and model checkpoints from Hugging Face used in experiments.

C.2 Benchmark Statistics

Table 10 provides detailed statistics for the four benchmarks used in our experiment. For OpenDocVQA, the SlideVQA and DUDE splits were merged, which consist of multi-page documents. This follows the *all-pool* setting in the original paper, but excludes the InfoVQA and ChartQA splits, which consist of single-page documents.

C.3 Evaluation Metric

Document-level Retrieval Given a document $D_k = (P_{k,1}, P_{k,2}, \ldots, P_{k,|D_k|})$, document-level retrieval accuracy $S(q, D_k)$ is defined using the maximum value of page-level retrieval accuracy:

$$S(q, D_k) = max_{P \epsilon D_k} S(q, P) \quad (1)$$

where S can be either S_{SV} or S_{MV}.

	Model	Model Size	Dimension	Base Model	Batch Size	Checkpoint
Single-Vector	BGE-M3 (dense)	560M	1,024	XLMRoberta	16	BAAI/bge-m3
	NV-Embed-V2	7.85B	4,096	Mistral-7B	16	nvidia/NV-Embed-v2
	VisRAG	3.43B	2,304	MiniCPM-V-2	1	openbmb/VisRAG-Ret
	GME	2.21B	1,536	Qwen2-VL-2B	1	Alibaba-NLP/gme-Qwen2-VL-2B-Instruct
	DSE	2.21B	1,536	Qwen2-VL-2B	1	MrLight/dse-qwen2-2b-mrl-v1
Multi-Vector	BGE-M3 (multi)	560M	$n_{token} \times 1,024$	XLMRoberta	16	BAAI/bge-m3
	ColPali	2.92B	$n_{patch} \times 128$	PaliGemma-3B	1	vidore/colpali-v1.3
	ColQwen2	2.21B	$n_{patch} \times 128$	Qwen2-VL-2B	1	vidore/colqwen2-v1.0
	ColQwen2.5	3.75B	$n_{patch} \times 128$	Qwen2.5-VL-3B	1	vidore/colqwen2.5-v0.1

Table 9: Model size, output dimension, base model, batch size and checkpoint from Hugging Face model used in experiments. indicates a textual embedding model and indicates a visual embedding model.

Benchmarks	# Pages (Images)	# Query
ViMDoc (Ours)	76,347	10,904
OpenDocVQA	80,335	1,256
ViDoSeek	5,385	1,142
M3DocVQA	41,071	2,441

Table 10: Statistics of the used benchmarks.

Recall@K Recall@K measures the proportion of ground truth pages that appear in the top-K retrieved results. Given a query, let P a set of all ground truth pages and P_K a set of ground truth pages in the top-K retrieved pages. Recall@K is defined as:

$$\text{Recall@K} = \frac{|P_K|}{|P|} \tag{2}$$

where $|P_K|$ is the number of ground truth pages retrieved in the top-K results, and $|P|$ is the total number of ground truth pages for the query.

FLOPs FLOPs were used as a retrieval efficiency evaluation metric along with latency in Table 5. It was observed that token length variations in queries and documents significantly affect scoring FLOPs during batch processing due to padding. For fair comparison, a batch size of 1 is assumed for all FLOPs calculations. All embeddings were processed and compared using float16 precision. Detailed analysis is provided in Table 11.

Method	FLOPs per Query
Single-Vector Retrieval	$O(d\|\mathcal{P}\|)$
Multi-Vector Retrieval	$O(d\, n_q \sum_{P\in\mathcal{P}} n_P)$
HEAVEN (Ours)	
Stage 1: Candidate Retrieval	$O(d\|\mathcal{VS}\|)$, where $\|\mathcal{VS}\| \approx \|\mathcal{P}\|/r$
Stage 1: Refinement	$O(d\|\mathcal{C}\|)$, where $\|\mathcal{C}\| \approx p_1\|\mathcal{P}\|$
Stage 2: Reranking	$O(d\|q_{\text{key}}\| \sum_{P\in\mathcal{C}_K} n_P)$, where $\|q_{\text{key}}\| \approx 0.3\, n_q$, $\|\mathcal{C}_K\| = K$
Stage 2: Refinement	$O(d\, n_q \sum_{P\in\mathcal{C}^\star} n_P)$, where $\|\mathcal{C}^\star\| = p_2 K$

Table 11: Computation cost analysis (FLOPs per query).

C.4 Hyperparameter for Efficiency Variants

We detail the specific hyperparameter ranges used to scale the existing efficiency variants of multi-vector models.

- **HEAVEN:** C_k, which defines the number of refine candidates from Stage 1, is used for scaling and tested using the set $C_k = \{100, 200, 400\}$.
- **Document Patch Pooling:** Hierarchical Pooler from ColPali (Faysse et al., 2025)[4] is used for patch pooling. Pool factors are varied across the range from 2^2 (4 patches) to 12^2 (144 patches).
- **Document Patch Pruning:** Following (Ma et al., 2025; Yan et al., 2025), random pruning is employed as a baseline, which has shown superior performance descpite its simplicity. Pruning ratios are set from 0.1 to 0.9 (using 0.1 increments). Additional fine-grained ratios of $\{0.075, 0.095, 0.925\}$ are included to allow for comparison at similar Floating Point Operations (FLOPs) scales.
- **Query Token Pooling:** The same pooler used for document patch pooling is utilzed. Considering the variance in query token length, pooling are performed on the fixed-length query special tokens[5], which has proven effective in (Qiao et al., 2025; Faysse et al., 2025). Pool factors of $\{2, 3, 5, 10\}$ are selected.
- **Query Token Pruning:** The same random pruning method is applied to the query augmentation tokens. Pruning ratios range from 0.1 to 0.9 (using 0.1 increments).

	#Cross-Page Query (%)	Methods	FLOPs	Cross-page Query			Single-page Query		
				R@1	R@3	R@5	R@1	R@3	R@5
ViMDoc	677 (6.2%)	DSE	0.235	23.21	44.76	55.10	60.33	79.22	84.74
		HEAVEN (only Stage 1)	0.134	22.65	44.07	54.86	59.97	78.71	84.28
		ColQwen2.5	407.320	25.92	50.25	60.13	74.10	88.77	92.30
		HEAVEN	0.486	25.22	48.16	57.83	73.58	88.38	91.90
OpenDocVQA	259 (10.6%)	DSE	0.247	36.97	64.96	73.58	65.20	78.64	82.95
		HEAVEN (only Stage 1)	0.147	37.93	65.35	74.68	65.60	78.84	82.95
		ColQwen2.5	482.049	42.18	72.94	79.18	80.54	89.87	91.78
		HEAVEN	0.541	43.15	72.68	78.22	77.73	86.76	88.67

Table 12: Experimental results for Cross-page Query (requiring retrieval of multiple pages) and Single-page Query. For the Cross-page Query, the average number of pages to be retrieved is 2.4, 2.0 for ViMDoc, OpenDocVQA, respectively. Recall@{1, 3, 5} is reported for both query types.

C.5 Extended Experimental Results

In this section, we provide further experimental results. Table 12 shows detailed results of cross-page query, which requires more than one page to be retrieved, and single-page query results. Following (Tanaka et al., 2025), Table 13 and Table 14 show experimental results where each retrieval is performed under each data split.

[4] https://pypi.org/project/colpali-engine/

[5] Also referred to as query augmentation tokens.

		ViMDoc (Proposed)														
	Data Split	**REAL-MM-RAG**			**VisR-Bench**			**MMDocIR**			**LongDocURL**			**MMLongBench-Doc**		
	Model	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**
Single-Vector	BGE-M3 (dense)	38.69	56.13	0.018	58.75	75.32	0.014	31.49	46.06	0.042	30.89	46.32	0.069	39.95	60.34	0.013
	NV-Embed-V2	49.48	68.52	0.070	59.98	79.52	0.057	35.28	53.66	0.167	35.58	54.66	0.278	45.79	69.99	0.053
	VisRAG	36.56	53.87	0.040	60.79	81.10	0.032	40.13	62.62	0.094	41.87	61.98	0.156	41.34	61.58	0.030
	GME	53.31	73.72	0.026	68.86	86.31	0.021	50.44	69.74	0.063	44.78	67.19	0.104	51.28	69.62	0.020
	DSE	53.74	73.85	0.026	69.88	87.07	0.021	52.71	71.02	0.063	45.08	66.96	0.104	45.07	66.40	0.020
	HEAVEN (only Stage 1)	53.82	73.80	0.015	68.92	85.53	0.013	52.10	70.10	0.036	44.41	66.26	0.059	45.42	65.87	0.011
	(vs. DSE)	100.14%	99.93%	-42.61%	98.64%	98.24%	-40.40%	98.83%	98.70%	-42.97%	98.51%	98.95%	-43.13%	100.79%	99.20%	-42.92%
Multi-Vector	BGE-M3 (multi)	48.67	66.03	157.599	68.26	84.23	101.84	32.89	49.24	491.229	33.34	50.95	837.209	43.11	65.10	164.000
	Colpali	59.25	77.61	75.914	75.83	90.22	55.602	50.29	67.84	189.206	47.69	69.79	402.024	43.26	62.94	68.038
	ColQwen2	60.88	77.43	53.484	80.47	92.82	39.549	54.80	71.72	109.115	51.90	69.67	227.017	48.53	69.17	48.420
	ColQwen2.5	68.85	85.61	53.484	81.95	93.27	39.549	59.41	77.06	109.115	52.13	73.17	227.017	51.37	70.76	48.420
	HEAVEN	70.68	87.31	0.416	80.90	93.23	0.378	61.10	77.93	0.403	53.18	73.84	0.567	52.72	70.92	0.536
	(vs.ColQwen2.5)	102.65%	101.99%	-99.22%	98.72%	99.96%	-99.65%	102.84%	101.13%	-98.98%	102.02%	100.92%	-99.75%	102.62%	100.23%	-98.89%

Table 13: Efficiency–accuracy comparison with various single-vector and multi-vector models for visual document retrieval on ViMDoc performed within each data split. We report Recall@{1,3} and per-query FLOPs (billions) for HEAVEN, compared with both single-vector and multi-vector models. The relative performance (%) for Recall is highlighted in blue, and the FLOPs is highlighted in red. indicates a textual embedding model and indicates a visual embedding model.

		OpenDocVQA					
	Data Split	**SlideVQA**			**DUDE**		
	Model	**R@1**	**R@3**	**FLOPs**	**R@1**	**R@3**	**FLOPs**
Single-Vector	BGE-M3 (dense)	41.25	56.71	0.107	37.35	50.15	0.057
	NV-Embed-V2	50.66	69.34	0.429	44.41	60.74	0.229
	VisRAG	57.50	75.26	0.241	44.71	59.66	0.129
	GME	59.67	78.46	0.161	49.83	66.75	0.086
	DSE	64.93	81.12	0.161	54.59	70.72	0.086
	HEAVEN (only Stage 1)	64.34	80.79	0.089	55.49	72.33	0.058
	(vs. DSE)	99.09%	99.59%	-45.00%	101.66%	102.28%	-31.98%
Multi-Vector	BGE-M3 (multi)	46.71	63.36	221.422	50.05	65.47	420.909
	Colpali	71.45	84.91	479.763	63.56	80.19	193.785
	ColQwen2	73.55	88.33	350.585	70.92	84.69	138.214
	ColQwen2.5	73.46	89.32	350.585	72.77	83.08	138.214
	HEAVEN	73.27	87.81	0.544	70.55	81.84	0.36
	(vs.ColQwen2.5)	99.73%	98.31%	-99.61%	96.95%	98.50%	-99.90%

Table 14: Efficiency–accuracy comparison with various single-vector and multi-vector models for visual document retrieval on OpenDocVQA performed within each data split. We report Recall@{1,3} and per-query FLOPs (billions) for HEAVEN, compared with both single-vector and multi-vector models. The relative performance (%) for Recall is highlighted in blue, and the FLOPs is highlighted in red. indicates a textual embedding model and indicates a visual embedding model.