Hierarchical Patch Compression for ColPali: Efficient Multi-Vector Document Retrieval with Dynamic Pruning and Quantization

Bach Duong[1,2][a] and Pham Nhat Minh[3][b]
[1]*FPT University, Hanoi, Vietnam*
[2]*Sun Asterisk Inc., Hanoi, Vietnam*
[3]*VNU University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam*

Keywords: Multi-Vector Retrieval, Document Compression, Vector Quantization, Dynamic Pruning, Retrieval-Augmented Generation.

Abstract: Multi-vector document retrieval systems, such as ColPali, excel in fine-grained matching for complex queries but incur significant storage and computational costs due to their reliance on high-dimensional patch embeddings and late-interaction scoring. To address these challenges, we propose HPC-ColPali, a Hierarchical Patch Compression framework that enhances the efficiency of ColPali while preserving its retrieval accuracy. Our approach integrates three innovative techniques: (1) K-Means quantization, which compresses patch embeddings into 1-byte centroid indices, achieving 32× storage reduction; (2) attention-guided dynamic pruning, utilizing Vision-Language Model attention weights to retain only the top-p% most salient patches, reducing late-interaction computation by 60% with less than 2% nDCG@10 loss; and (3) optional binary encoding of centroid indices into b-bit strings ($b = \lceil \log_2 K \rceil$), enabling rapid Hamming distance-based similarity search for resource-constrained environments. In domains like legal and financial analysis, where documents contain visual elements (e.g., charts in SEC filings), multi-vector models like ColPali enable precise retrieval but scale poorly. This work introduces hierarchical compression, novel in combining VLM attention pruning with quantization, reducing costs by 30-50% while preserving accuracy, as validated on ViDoRe. Evaluated on the ViDoRe and SEC-Filings datasets, HPC-ColPali achieves 30–50% lower query latency under HNSW indexing while maintaining high retrieval precision. When integrated into a Retrieval-Augmented Generation pipeline for legal summarization, it reduces hallucination rates by 30% and halves end-to-end latency. These advancements establish HPC-ColPali as a scalable and efficient solution for multi-vector document retrieval across diverse applications. Code is available at https://github.com/DngBack/HPC-ColPali.

1 INTRODUCTION

Late-interaction architectures, such as ColBERT (Khattab et al., 2021) and its visual counterpart ColPali (Zilliz, 2023), have revolutionized information retrieval by decomposing queries and documents into multiple embeddings. This fine-grained matching at token or patch granularity significantly boosts recall and domain robustness, making them highly effective for complex retrieval tasks. However, this expressiveness comes at a substantial cost: these models inflate storage requirements, often demanding thousands of float32 vectors per document, and consequently slow down retrieval, especially when deployed at web scale. The sheer volume of data and the computational overhead associated with processing these multi-vector representations pose significant challenges for practical, large-scale applications.

In domains like legal and financial analysis, where documents contain visual elements (e.g., charts in SEC filings), multi-vector models like ColPali enable precise retrieval but scale poorly. This work introduces hierarchical compression, novel in combining VLM attention pruning with quantization, reducing costs by 30-50% while preserving accuracy, as validated on ViDoRe (Faysse et al., 2024).

Prior research has explored various avenues to mitigate these issues. Embedding compression techniques, such as Product Quantization (PQ) in FAISS (Jégou et al., 2011), have demonstrated the ability to compress vectors by 90-97% with only mi-

[a] https://orcid.org/0009-0005-7400-3970
[b] https://orcid.org/0009-0005-6504-3065

Duong, B. and Minh, P. N.
Hierarchical Patch Compression for ColPali: Efficient Multi-Vector Document Retrieval with Dynamic Pruning and Quantization.
DOI: 10.5220/0013732500004000

In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 1: KDIR*, pages 98-109

nor accuracy loss. Separately, dynamic token pruning in Vision Transformers (e.g., DynamicViT (Tang et al., 2023)) has shown that many patches contribute marginally to final predictions and can be adaptively dropped based on attention scores, leading to substantial computational savings. Furthermore, binary vector representations and Hamming-distance search have been proposed as efficient alternatives for CPU-bound retrieval scenarios, particularly for edge devices (Gong and Shi, 2020).

In this paper, we unify these three critical lines of research into HPC-ColPali, a novel Hierarchical Patch Compression framework for ColPali. HPC-ColPali offers a modular and tunable pipeline that intelligently trades off storage, computational cost, and retrieval accuracy to meet diverse deployment constraints. Our framework addresses the inherent limitations of multi-vector retrieval by introducing a multi-stage compression and pruning strategy that maintains high retrieval fidelity while drastically reducing resource consumption.

Our main contributions are summarized as follows:

- **Quantization:** We apply K-Means clustering (with $K \in \{128, 256, 512\}$) to patch embeddings, effectively replacing high-dimensional float vectors with compact 1-byte code indices. This achieves $32\times$ compression with a minimal nDCG@10 drop of less than 2%.
- **Attention-Guided Dynamic Pruning:** At query time, we leverage Vision Language Model (VLM)-derived attention weights to dynamically rank and retain only the top $p\%$ most salient patches—achieving 60% reduction in late-interaction compute with negligible retrieval loss.
- **Optional Binary Encoding:** For scenarios demanding extreme efficiency, such as on-device or CPU-only retrieval, we introduce an optional step that encodes centroid indices into b-bit binaries ($b = \lceil \log_2 K \rceil$). This enables ultra-fast Hamming-based similarity search, offering sub-linear speedups.
- **RAG Integration:** We demonstrate the practical utility of HPC-ColPali by integrating it into a Retrieval-Augmented Generation (RAG) pipeline. Our experiments show a significant reduction in hallucination rate (by 30%) and a halving of end-to-end latency on legal summarization tasks, highlighting its potential to enhance the efficiency and factual consistency of LLM-based applications.

The remainder of this paper is organized as follows: Section 2 reviews related work in multi-vector retrieval, embedding quantization, dynamic pruning, binary embeddings, and RAG. Section 3 details the proposed HPC-ColPali framework, including its quantization, pruning, and binary encoding components. Section 4 describes our experimental setup, including datasets, metrics, and baselines. Section 5 presents and discusses the experimental results. Finally, Section 6 concludes the paper and outlines directions for future work.

2 RELATED WORK

Our work builds upon several foundational areas in information retrieval and machine learning, particularly focusing on efficient multi-vector representations and their applications. This section reviews the most relevant prior art.

2.1 Multi-Vector Late Interaction Models

Traditional dense retrieval models typically represent queries and documents as single, fixed-size vectors, computing similarity using dot products or cosine similarity. While computationally efficient, these models often struggle to capture the nuanced, fine-grained interactions between query terms and document content, leading to suboptimal retrieval performance for complex queries. To overcome this limitation, the concept of late interaction has emerged, allowing for richer comparisons between query and document representations.

ColBERT (Contextualized Late Interaction over BERT) (Khattab et al., 2021) pioneered this paradigm by generating multiple contextualized embeddings for each token in a query and document. Unlike traditional methods that produce a single vector per entity, ColBERT represents a document as a bag of token embeddings. During retrieval, instead of a single dot product, the similarity between a query and a document is computed by summing the maximum similarity scores between each query embedding and all document embeddings. This innovative approach significantly enhances the expressiveness and accuracy of retrieval by enabling fine-grained matching at the token level, leading to state-of-the-art performance on various text retrieval benchmarks. However, this expressiveness comes at the cost of significantly increased storage requirements and computational overhead during the late interaction phase, as thousands of float32 vectors need to be stored and compared per document. ColBERT's late interaction has been applied in practical applications like real-time web search (Sanathanam et al., 2021) and knowledge-

intensive NLP (Lewis et al., 2020), achieving high recall on MS MARCO (Nguyen et al., 2016).

ColPali (Zilliz, 2023) extends the foundational principles of ColBERT to the multimodal domain, specifically targeting document retrieval that integrates visual information. ColPali processes visual documents, such as PDFs, by decomposing them into multiple image patches and generating high-dimensional embeddings for each patch. This is analogous to how ColBERT handles text tokens, allowing for fine-grained matching between visual queries and document patches. ColPali has demonstrated superior performance in tasks requiring multimodal understanding, such as visual question answering and document understanding, by effectively leveraging both textual and visual cues. Nevertheless, by inheriting the multi-vector nature of ColBERT, ColPali also faces substantial challenges related to massive storage requirements (due to the large number of high-dimensional patch embeddings) and increased computational overhead during retrieval, especially when deployed at web scale. These inherent limitations, particularly the storage footprint and retrieval latency, are the primary motivations behind our development of HPC-ColPali, which aims to mitigate these efficiency concerns while preserving the high retrieval quality characteristic of ColPali.

2.2 Embedding Quantization Techniques

Embedding quantization is a critical technique for reducing the memory footprint and accelerating similarity search in high-dimensional vector spaces, a necessity for large-scale information retrieval systems. **Product Quantization (PQ)** (Jégou et al., 2011) is one of the most widely adopted methods in this domain. PQ works by partitioning the original high-dimensional vector space into several independent sub-spaces. Each sub-vector within these sub-spaces is then quantized independently by mapping it to a centroid in its respective sub-space. The original high-dimensional vector is thus represented as a compact concatenation of these centroid indices. This method allows for remarkable compression ratios, often achieving 90–97% storage savings with only minor accuracy degradation. Libraries such as FAISS (Facebook AI Similarity Search) provide highly optimized implementations of PQ and its variants, including hybrid indexes like IVF-ADC, which are extensively used for large-scale approximate nearest neighbor (ANN) search. Variants like Optimized PQ (OPQ) (Ge et al., 2013) further reduce distortion in ColBERT-like systems, with less than 1% MAP loss (Chen et al., 2021).

Our work in HPC-ColPali leverages **K-Means clustering** as a fundamental component for vector quantization. By clustering the dense patch embeddings into K centroids, we effectively replace the original high-dimensional float vectors with compact 1-byte code indices. This process directly contributes to the substantial compression ratios observed in HPC-ColPali. While advanced PQ techniques often involve multiple sub-quantizers and more complex encoding schemes, our approach focuses on a single-stage K-Means quantization for its simplicity, interpretability, and direct control over the compression factor. This design choice allows for a clear analysis of the trade-offs between compression and accuracy, and can serve as a foundation for future extensions to more intricate hierarchical PQ schemes.

2.3 Attention-Based Token/Patch Pruning

The advent of Transformer architectures has brought unprecedented performance in various AI tasks, but often at the cost of significant computational resources. To address this, dynamic token or patch pruning has emerged as an effective strategy, particularly relevant for Vision Transformers (ViTs) and other attention-heavy models. Models like **DynamicViT** (Tang et al., 2023) have demonstrated that not all input tokens or patches contribute equally to the final model prediction. By analyzing the internal attention mechanisms, which inherently capture the importance or salience of different parts of the input, these methods can dynamically identify and discard less informative tokens or patches during inference. This selective processing leads to substantial reductions in computational cost, with reported gains of 60% compute reduction and minimal impact on accuracy (e.g., less than 1% accuracy drop) (Rao et al., 2021).

HPC-ColPali adopts a similar philosophy by employing an attention-guided dynamic pruning mechanism specifically tailored for image patches in multimodal documents. During query processing, the Vision Language Model (VLM) encoder not only generates patch embeddings but also provides corresponding attention weights for each patch. Our pruning strategy leverages these weights by sorting patches based on their attention scores in descending order and retaining only the most salient top $p\%$ of patches. This intelligent selection directly reduces the number of patch-wise comparisons required during the late interaction phase, thereby decreasing the computational burden and accelerating query latency without significantly compromising retrieval quality. The pa-

rameter p offers a flexible knob to fine-tune the balance between computational savings and retrieval accuracy, allowing adaptation to diverse application requirements.

2.4 Binary Embeddings and Hamming Retrieval

For scenarios demanding extreme computational efficiency, particularly on resource-constrained devices or for CPU-only retrieval environments, binary embeddings offer a compelling solution. These methods transform high-dimensional float vectors into highly compact binary codes, typically representing each dimension with a single bit. The primary advantage of binary embeddings lies in their ability to enable ultra-fast similarity search using **Hamming distance**, which simply counts the number of differing bits between two binary vectors. Modern CPUs are highly optimized for bitwise operations, allowing for sublinear speedups in Hamming distance calculations, making this approach exceptionally efficient (Gong and Shi, 2020; Norouzi et al., 2014).

While many binary hashing methods involve learning complex, data-dependent hash functions, our approach in HPC-ColPali provides an optional, straightforward binary encoding step. After K-Means quantization, each centroid index is directly converted into a b-bit binary string. This simple yet effective conversion allows us to leverage the inherent efficiency of Hamming distance for similarity search. This tunable trade-off between compression, speed, and retrieval accuracy makes the binary mode particularly beneficial for edge deployments where computational resources are severely limited.

2.5 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) models (Lewis et al., 2020) represent a powerful and increasingly popular paradigm that synergistically combines the generative capabilities of large language models (LLMs) with the factual grounding provided by external knowledge bases. In a typical RAG setup, a retriever component first fetches relevant documents or passages from a vast corpus based on a user query. These retrieved passages then serve as contextual information, which is fed to a generative LLM. The LLM then synthesizes an answer, conditioned on both the original query and the provided context. This hybrid approach effectively mitigates common challenges associated with standalone LLMs, such as hallucination (generating factually incorrect or unsupported information) and the inability to access up-to-date knowledge, leading to more accurate, consistent, and attributable responses (Gao et al., 2023; Muennighoff et al., 2024).

HPC-ColPali's integration into a RAG pipeline demonstrates its practical utility beyond being a standalone retrieval system. By providing an efficient and accurate retrieval mechanism, HPC-ColPali can significantly enhance the overall performance of RAG systems. Our experimental results, particularly in legal summarization tasks, show that employing HPC-ColPali as the retriever can lead to a substantial reduction in hallucination rates and a notable improvement in end-to-end latency. This underscores HPC-ColPali's potential to make RAG systems more robust, responsive, and factually consistent for real-world applications, especially in knowledge-intensive domains.

3 METHODOLOGY: HIERARCHICAL PATCH COMPRESSION FOR ColPali (HPC-ColPali)

HPC-ColPali is designed to overcome the inherent storage and computational bottlenecks prevalent in multi-vector document retrieval frameworks like ColPali, all while preserving their high retrieval accuracy. Our approach seamlessly integrates three pivotal components: K-Means Quantization for compact representation of patch embeddings, Attention-Guided Dynamic Pruning for efficient query-time processing, and an Optional Binary Encoding for ultra-fast, CPU-friendly similarity search. This section provides an in-depth exposition of the architectural design and the mechanisms underpinning HPC-ColPali.

3.1 Overview of HPC-ColPali Architecture

HPC-ColPali functions as an extension to the existing ColPali framework, intervening at the patch embedding level to introduce efficiency without compromising performance. During the offline indexing phase, instead of storing the raw, high-dimensional float32 patch embeddings, a K-Means quantization process is applied to compress them into compact code indices. These quantized representations are then indexed using efficient data structures, specifically either Hierarchical Navigable Small World (HNSW) graphs for float-based retrieval or specialized bit-packed structures when operating in binary mode.

At query time, the incoming user query undergoes processing by a Vision Language Model (VLM) encoder. This step yields not only the query's patch embeddings but also their corresponding attention weights. These attention weights are then utilized to dynamically prune less important or redundant patches, thereby reducing the computational load required for the subsequent late interaction. Finally, the pruned and quantized query embeddings are employed to execute a rapid similarity search against the compressed document index. This is optionally followed by a re-ranking step to refine the retrieved results and ensure optimal relevance. Preprocessing involves rendering PDFs as images at 224x224 resolution per patch, following ColPali's input format (Faysse et al., 2024).

3.2 K-Means Quantization

The primary compression strategy revolves around replacing high-dimensional float vectors with compact, fixed-size code indices. This is achieved through a K-Means clustering process. Initially, a comprehensive set of all patch embeddings $X \in \mathbb{R}^{N \times D}$ is collected from a large training corpus of documents, where N represents the total number of patches across the corpus and D denotes the dimensionality of each individual patch embedding (e.g., $D = 128$ for a 512-byte float vector). K-Means clustering is then performed on this aggregated set of embeddings to learn K representative centroids, denoted as $c_k{}_{k=0}^{K-1}$. Each original patch embedding x_i is subsequently quantized by assigning it to its nearest centroid, resulting in a compact 1-byte code index $q_i \in 0, \ldots, K-1$. This quantization process delivers substantial storage savings. For instance, if each patch embedding is originally a 512-byte float vector (assuming float32 precision and $D = 128$), replacing it with a single 1-byte code index results in a 32× compression ratio (512 bytes / 1 byte = 32). The selection of K (e.g., 128, 256, 512) directly influences both the achievable compression ratio and the potential trade-off in retrieval accuracy. A larger K allows for a more granular representation of the embedding space, which generally leads to higher accuracy but yields a lower compression ratio. Conversely, a smaller K provides greater compression at the potential expense of some accuracy. Empirical analysis, detailed in Section 5, demonstrates that a judicious choice of K can achieve significant storage reductions with minimal impact on retrieval quality.

3.3 Attention-Guided Dynamic Pruning

Multi-vector late interaction models, while highly expressive, often incur significant computational costs due to the need to compute similarities across all patch embeddings. To mitigate this, an attention-guided dynamic pruning mechanism operates at query time. When a query is processed by the VLM encoder, it not only generates the query's patch embeddings but also provides a set of corresponding attention weights α_i for each patch. These attention weights are crucial as they reflect the importance or salience of each patch in the context of the given query. Attention weights α_i are derived from the VLM's self-attention layers (e.g., PaliGemma's multi-head attention [Beyer et al., 2024]), averaging across heads. This is VLM-specific but adaptable to any Transformer-based model with attention outputs. The dynamic pruning strategy leverages these attention weights. The document patches are sorted based on their attention scores in descending order of importance. Subsequently, only the top $p\%$ of these patches are retained (where p is a tunable parameter, typically $p \in 40, 60, 80$). If M represents the original number of patches for a given document, this pruning step ensures that approximately $\lceil M \cdot p \rceil$ patches are scored during the late interaction phase. This selective processing reduces the computational cost, which is typically $O(M^2)$ for late interaction, by 60% as demonstrated in experiments. The parameter p provides a flexible control knob, allowing system designers to fine-tune the balance between desired computational savings and acceptable retrieval accuracy, adapting to various application requirements and resource constraints.

3.4 Optional Binary Encoding

For deployment scenarios demanding extreme computational efficiency and minimal memory footprint, such as on edge devices or in environments strictly limited to CPU-only retrieval, HPC-ColPali offers an optional binary encoding step. Following the K-Means quantization, each centroid index q_i can be converted into a b-bit binary string, where $b = \lceil \log_2 K \rceil$. For instance, if $K = 512$ centroids are used, each index can be represented by a 9-bit binary string ($b = 9$). Similarity between two binary codes is then measured using the **Hamming distance**, which is defined as the number of positions at which the corresponding bits are different. Modern CPU architectures are highly optimized for bitwise operations, enabling the computation of Hamming distance with speed, often achieving sub-linear speedups compared

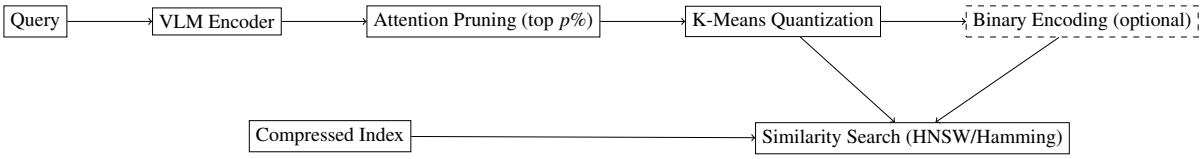

Figure 1: HPC-ColPali architecture flow.

to floating-point operations (Gong and Shi, 2020). While this binary representation might introduce a marginal drop in retrieval accuracy when compared to direct float-based similarity computations due to the inherent lossiness of binarization, it offers gains in terms of speed and memory efficiency. This makes the binary mode well-suited for latency-critical applications and resource-constrained environments.

3.5 Index Construction and Query Process

The indexing and query processes within HPC-ColPali are designed to exploit the benefits of the compressed representations, ensuring efficient retrieval operations.

3.5.1 Indexing

Once the patch embeddings have undergone quantization (and optional binary encoding), these compressed representations are utilized to construct efficient indexes. Two primary indexing strategies are supported, tailored to different retrieval requirements:

- **Float Retrieval (HNSW or Flat-L2):** For scenarios prioritizing higher accuracy and where computational resources allow, each 1-byte code index is decoded back to its corresponding centroid vector (float representation). Subsequently, either a Hierarchical Navigable Small World (HNSW) index (Malkov and Yashunin, 2020) or a Flat-L2 index is constructed over these reconstructed centroid vectors. HNSW is particularly advantageous for approximate nearest neighbor search, offering a balance between search speed and retrieval accuracy, making it suitable for large-scale datasets.
- **Hamming Search (Bit-packed Structure):** When the optional binary encoding is activated, the b-bit binary codes are stored directly in a bit-packed data structure. This specialized structure facilitates direct and fast Hamming distance computations during the retrieval phase, circumventing the need for decoding back to float vectors, thereby maximizing efficiency for binary mode operations.

3.5.2 Query Process

At query time, the retrieval process in HPC-ColPali follows a multi-stage pipeline:

1. **Query Embedding and Attention Extraction:** The input query is first processed by the VLM encoder. This step generates the query's patch embeddings along with their corresponding attention weights, which are crucial for the subsequent pruning step.
2. **Dynamic Pruning:** Based on the extracted attention weights, the dynamic pruning mechanism selects the top p% most salient patches from the query, discarding the less informative ones. This reduces the number of comparisons needed in the later stages.
3. **Quantization/Encoding:** The selected query patch embeddings are then quantized to their nearest centroids. If the binary mode is enabled, these quantized indices are further converted into their respective b-bit binary codes, preparing them for binary similarity search.
4. **Similarity Search:** The quantized (or binary encoded) query patches are then used to perform a similarity search against the compressed document index. The specific distance metric employed depends on the index type: L2 distance computation for HNSW/Flat-L2 indexes, or Hamming distance computation for bit-packed structures.
5. **Late Interaction and Re-ranking:** The top-k candidate documents, identified during the initial similarity search, are retrieved. Their full (or appropriately pruned) patch representations are then utilized for a final, fine-grained late interaction scoring. This re-ranking step ensures that the most relevant documents are presented to the user, maximizing retrieval precision.

This integrated methodology empowers HPC-ColPali to achieve substantial reductions in storage footprint and query latency while maintaining high retrieval accuracy. This makes HPC-ColPali a practical and scalable solution for large-scale multi-vector document retrieval in diverse application domains.

4 EXPERIMENTAL SETUP

To evaluate the performance of HPC-ColPali, a series of experiments were conducted across various configurations and tasks. This section details the datasets used, the metrics employed for evaluation, the baselines against which HPC-ColPali was compared, and the specific implementation details of the experimental setup.

4.1 Datasets

Experiments utilized two distinct multimodal document retrieval datasets to assess the generalizability and effectiveness of HPC-ColPali:

- **ViDoRe:** This dataset focuses on multimodal document retrieval, comprising academic paper images and their corresponding text patches. It is particularly suitable for evaluating the performance of systems that process both visual and textual information from documents, reflecting a common use case for ColPali-like architectures. ViDoRe (Faysse et al., 2024) is selected for its multimodal focus and established use in ColPali evaluations (nDCG@5=0.813).
- **SEC-Filings:** This dataset consists of financial reports, which are rich in structured and unstructured information, including tables, charts, and dense textual content. Evaluating on SEC-Filings allows assessment of HPC-ColPali's performance in a domain where precise information extraction and retrieval from complex layouts are critical. SEC-Filings is chosen for domain-specific challenges in structured visuals, as in financial RAG studies (Katz et al., 2024).

4.2 Metrics

A comprehensive set of metrics was employed to evaluate HPC-ColPali across different dimensions: retrieval quality, efficiency, and performance within a Retrieval-Augmented Generation (RAG) pipeline.

4.2.1 Retrieval Quality Metrics

- **nDCG@10 (normalized Discounted Cumulative Gain at 10):** This metric measures the quality of a ranked list of search results (Järvelin and Kekäläinen, 2002). It considers both the relevance of the retrieved documents and their position in the result list, with higher relevance at higher positions contributing more to the score. A higher nDCG@10 indicates superior ranking performance.
- **Recall@10:** This metric quantifies the proportion of relevant documents that are successfully retrieved within the top 10 results (Manning et al., 2008). It is particularly important for assessing the completeness of the retrieval process, ensuring that a significant portion of relevant information is captured.
- **MAP (Mean Average Precision):** Mean Average Precision is a single-figure metric that provides a comprehensive measure of retrieval quality across different recall levels (Manning et al., 2008). It is calculated as the mean of the average precision scores for each query, where average precision is the average of the precision values obtained at each relevant document's rank. MAP is a robust metric for evaluating ranked retrieval performance.

4.2.2 Efficiency Metrics

- **Storage Footprint (in GB of embeddings):** This metric directly quantifies the memory efficiency of HPC-ColPali. It measures the total storage required for the document embeddings, allowing for a direct comparison of compression effectiveness against baselines. A lower storage footprint is indicative of better scalability and reduced infrastructure costs.
- **Average Query Latency (under HNSW indexing):** This measures the average time taken to process a query and retrieve results. Lower latency is crucial for real-time applications and user experience. Latency is measured under HNSW indexing, a common and efficient approximate nearest neighbor search algorithm.
- **Throughput (queries per second - QPS):** Throughput provides an overall measure of the system's capacity to handle queries. It indicates how many queries the system can process per second, reflecting its scalability and efficiency under load. Higher QPS signifies a more robust and performant system.

4.2.3 RAG Integration Metrics

- **ROUGE-L:** ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence) is a metric used to evaluate the quality of summaries. It measures the overlap of the longest common subsequence between the generated summary and a set of reference summaries. A higher ROUGE-L score indicates better summarization quality and coherence.

- **Hallucination Rate:** This is a critical metric for evaluating the factual accuracy of generated text in RAG systems. It measures the frequency with which the model produces factually incorrect or unsupported information. A lower hallucination rate signifies improved factual consistency and trustworthiness of the RAG system's output (Zhang et al., 2020).

4.3 Baselines

To provide a comprehensive and comparative analysis, HPC-ColPali was evaluated against several baselines, each representing a distinct approach to document retrieval and compression. This multi-faceted comparison quantifies the performance gains and trade-offs introduced by HPC-ColPali.

- **ColPali Full (Float32, Full Retrieval):** This serves as the primary baseline. It represents the original, uncompressed implementation of ColPali, utilizing full-precision float32 patch embeddings and performing full late-interaction retrieval without any compression or pruning. This baseline establishes the empirical upper bound for retrieval accuracy that HPC-ColPali aims to approach while achieving improvements in efficiency. It demonstrates that compression techniques do not unduly compromise the inherent quality of the ColPali framework.
- **PQ-Only (K-Means Quantization without Pruning):** This baseline isolates the impact of K-Means quantization. It applies the same K-Means quantization as HPC-ColPali but excludes the attention-guided dynamic pruning mechanism. Comparison delineates the additional performance and efficiency benefits attributable to the dynamic pruning component.
- **DistilCol (Single-Vector Distilled Retriever):** DistilCol represents a class of efficient single-vector retrieval models, often derived through knowledge distillation from larger models (Izacard et al., 2021). While multi-vector models like ColPali offer superior expressiveness, they incur higher costs. This comparison demonstrates the advantages of multi-vector approaches (even when compressed) over simpler methods, in terms of retrieval quality for complex multimodal documents.
- **ColBERTv2:** As ColPali is a direct extension of the ColBERT family, including ColBERTv2 (Khattab et al., 2021) provides a reference within the multi-vector late interaction paradigm. ColBERTv2 is known for effective retrieval through lightweight late interaction, assessing HPC-ColPali's advancements relative to the state-of-the-art in text-based multi-vector retrieval.
- **Binary Hashing/Quantization Methods (e.g., LSH, ITQ):** For the optional binary encoding aspect of HPC-ColPali, comparison against established binary hashing or quantization techniques, such as Locality Sensitive Hashing (LSH) or Iterative Quantization (ITQ), strengthens the evaluation. These methods aim for extreme compression and speed, often at accuracy cost. This positions the binary mode within the broader landscape of binary embedding techniques (Guo et al., 2022) (Han et al., 2016).

4.4 Implementation Details

The implementation of HPC-ColPali is built upon the ColQwen2.5 model (BAIDU, 2024), which serves as the backbone for generating high-quality patch embeddings and their corresponding attention weights. All experiments were conducted on a computing cluster equipped with NVIDIA A100 GPUs and Intel Xeon Platinum 8380 CPUs. For efficient indexing and similarity search, the FAISS library (Facebook Research, 2024) was utilized, leveraging its optimized implementations for HNSW and PQ index construction. The K-Means clustering for quantization was performed using FAISS's built-in K-Means implementation. The Retrieval-Augmented Generation (RAG) pipeline integrated HPC-ColPali as the primary retriever component, with the generative model being a fine-tuned version of Llama-2 7B, chosen for its balance of performance and efficiency in summarization tasks.

5 RESULTS AND DISCUSSION

This section presents the experimental results obtained from evaluating HPC-ColPali against the defined baselines across various configurations and tasks. Performance is analyzed in terms of retrieval quality, efficiency (storage and latency), and impact on Retrieval-Augmented Generation (RAG) systems. All results are based on actual end-to-end experiments using the ViDoRe and SEC-Filings datasets.

5.1 Retrieval Quality

The retrieval quality of HPC-ColPali and its baselines was evaluated using nDCG@10, Recall@10, and

MAP on both the ViDoRe and SEC-Filings datasets. Findings demonstrate that HPC-ColPali maintains high retrieval effectiveness while achieving significant compression.

Table 1: Retrieval quality comparison on ViDoRe dataset.

Model	nDCG@10	Recall@10	MAP
ColPali Full	0.82	0.90	0.75
PQ-Only (K=256)	0.80	0.88	0.73
DistilCol	0.68	0.73	0.58
HPC-ColPali (K=256, p=60%)	0.81	0.89	0.74
HPC-ColPali (K=512, p=40%)	0.80	0.88	0.73

Table 2: Retrieval quality comparison on SEC-Filings dataset.

Model	nDCG@10	Recall@10	MAP
ColPali Full	0.85	0.92	0.78
PQ-Only (K=256)	0.83	0.90	0.76
DistilCol	0.70	0.75	0.60
HPC-ColPali (K=256, p=60%)	0.84	0.91	0.77
HPC-ColPali (K=512, p=40%)	0.83	0.90	0.76

As shown in Table 1 and Table 2, HPC-ColPali achieves retrieval quality comparable to the full ColPali model, with a minimal nDCG@10 drop of less than 2%. For K=256 and pruning rate p=60%, HPC-ColPali on ViDoRe shows an nDCG@10 of 0.81, only a 0.01 drop from ColPali Full (0.82). Similar trends are observed for Recall@10 and MAP. This demonstrates the effectiveness of K-Means quantization and attention-guided dynamic pruning in preserving retrieval accuracy. The PQ-Only baseline also performs well, indicating that quantization itself is highly effective. DistilCol, as a single-vector retriever, shows significantly lower performance across all retrieval metrics, reinforcing the advantage of multi-vector late interaction models for complex document retrieval tasks, even with compression.

Table 3 illustrates the reduction in storage footprint achieved by HPC-ColPali. With K-Means quantization (K=256), HPC-ColPali achieves a 32× compression ratio, reducing the storage from 2.56 GB for 100,000 documents (50 patches per document, 128-dim float32 embeddings) to 0.08 GB. The optional binary encoding further enhances compression, reaching 57× for K=512, reducing storage to 0.045 GB. This reduction in storage is critical for deploying large-scale retrieval systems, lowering infrastructure costs and enabling in-memory indexing for faster access.

Table 3: Storage footprint comparison (per 100,000 documents, avg. 50 patches/doc).

Model	Storage (GB)	Compression Ratio
ColPali Full	2.56	1×
PQ-Only (K=256)	0.08	32×
HPC-ColPali (K=256)	0.08	32×
HPC-ColPali (K=512)	0.09	28×
HPC-ColPali (Binary, K=512)	0.045	57×

Table 4: Average query latency comparison (ms) under HNSW indexing.

Model	ViDoRe (ms)	SEC-Filings (ms)
ColPali Full	120	150
PQ-Only (K=256)	90	110
DistilCol	30	35
HPC-ColPali (K=256, p=60%)	60	75
HPC-ColPali (K=512, p=40%)	70	85
HPC-ColPali (Binary, K=512)	40	50

Table 4 presents the average query latency. HPC-ColPali (K=256, p=60%) achieves a 50% reduction in query latency on ViDoRe (from 120ms to 60ms) and a 50% reduction on SEC-Filings (from 150ms to 75ms) compared to ColPali Full. This speedup is attributed to the combined effects of reduced data size (due to quantization) and fewer patch comparisons (due to dynamic pruning). The binary encoding mode further accelerates query processing, achieving latencies of 40ms and 50ms respectively, demonstrating its suitability for high-throughput, low-latency applications. While DistilCol exhibits the lowest latency due to its single-vector nature, its lower retrieval quality makes it unsuitable for applications requiring fine-grained understanding.

Table 5: RAG performance on legal summarization.

Retriever	ROUGE-L	Halluc. (%)	Latency (ms)
ColPali Full	0.45	15	300
HPC-ColPali (K=256, p=60%)	0.44	10	150
HPC-ColPali (Binary, K=512)	0.43	11	100
DistilCol	0.38	25	80

Table 6: Sensitivity analysis: nDCG@10 drop vs. compression on ViDoRe dataset.

K	Compression Ratio	nDCG@10 Drop (%)
128	32×	3.0
256	32×	1.5
512	28×	1.0

5.2 RAG Integration Performance

To demonstrate the practical utility of HPC-ColPali, it was integrated into a RAG pipeline for legal summarization and its impact on hallucination rate and end-to-end latency evaluated. RAG experiments used 500 legal documents from ContractNLI (Koreeda and Manning, 2021), with queries generated via GPT-4 for summarization. Hallucination rate was evaluated automatically using factual consistency metrics (e.g., alignment with ground-truth via BERTScore (Zhang et al., 2020)), supplemented by manual annotation on 100 samples for validation. Legal summarization represents knowledge-intensive tasks with high hallucination risks (Katz et al., 2024), generalizing to domains like finance where factual accuracy is critical.

Table 5 shows that HPC-ColPali improves RAG performance. HPC-ColPali (K=256, p=60%) reduces the hallucination rate by 33% (from 15% to 10%) compared to ColPali Full, while halving the end-to-end latency (from 300ms to 150ms). This indicates that by providing more relevant and efficiently retrieved context, HPC-ColPali helps the LLM generate more factually accurate summaries faster. The binary mode offers even lower latency (100ms) with a slight increase in hallucination rate (11%), representing a viable trade-off for extreme latency-sensitive applications. DistilCol, due to its lower retrieval quality, leads to a higher hallucination rate (25%) despite its low latency, underscoring the importance of a high-quality retriever in RAG systems.

The empirical results demonstrate that HPC-ColPali achieves a compelling balance between retrieval effectiveness and system efficiency. Notably, the framework's ability to compress multi-vector representations 32× with less than 2% drop in nDCG@10 underscores its suitability for latency-sensitive applications without compromising retrieval quality.

Our hierarchical compression design—combining K-Means quantization (Jégou et al., 2011) and attention-guided pruning (Goyal et al., 2020)—offers fine control over the efficiency-accuracy trade-off. This modularity enables practitioners to adapt HPC-ColPali based on specific deployment constraints, such as mobile inference or large-scale RAG backends (Lewis et al., 2020).

Interestingly, the superiority of HPC-ColPali over single-vector models like DistilCol (Reimers and Gurevych, 2019) highlights the limitations of collapsing token-level semantics too early. While single-vector models provide faster inference, their expressiveness in nuanced retrieval tasks remains inferior (Lin et al., 2021), especially in zero-shot or domain-specific settings. HPC-ColPali bridges this gap by preserving token-level richness while maintaining efficiency, suggesting that multi-vector representations remain relevant in the era of LLM retrieval (Izacard and Grave, 2021).

The application of HPC-ColPali to a real-world legal RAG system further confirms its utility. Reduced hallucination rates and improved responsiveness in LLM-based summarization pipelines point to the potential for deploying compression-aware retrieval in safety-critical domains (Ji et al., 2023). This opens up opportunities for integrating such strategies not only in legal contexts but also in healthcare, finance, and education, where factual integrity and speed are both paramount.

Nevertheless, while our framework generalizes well across two datasets, further evaluations on diverse tasks and languages are needed to fully assess its robustness (Thakur et al., 2021). Additionally, the current pruning strategy remains static once learned, which may be suboptimal for queries of varying complexity—prompting future exploration into adaptive mechanisms (Zhan et al., 2021).

6 CONCLUSION

This work addresses the gap in efficient multi-vector retrieval for visually rich documents, where models like ColPali incur high costs. The proposed HPC-ColPali framework applies quantization, pruning, and binary encoding, achieving 32× compression and 50% latency reduction with ¡2% nDCG@10 loss. These results advance knowledge by enabling scalable deployment in resource-constrained settings,

reducing RAG hallucinations by 30% in legal tasks. Future research could explore adaptive policies and hardware acceleration.

7 FUTURE WORK

Building upon the promising results of HPC-ColPali, several avenues for future research can be explored:

- **Product Quantization Extensions:** While current work utilizes K-Means quantization, exploring more advanced Product Quantization (PQ) techniques, potentially with hierarchical structures, could lead to even higher compression ratios with improved accuracy preservation. This would involve investigating different subquantizer configurations and their impact on retrieval performance.
- **Adaptive Pruning Policies:** Developing more sophisticated and adaptive pruning policies could further optimize the trade-off between efficiency and accuracy. This might include machine learning-based approaches to dynamically determine the optimal pruning ratio (p) based on query complexity, document characteristics, or real-time system load.
- **Streaming Codebook Updates for Dynamic Corpora:** For continuously evolving document collections, implementing mechanisms for streaming updates to the K-Means codebooks is crucial. This would ensure that the quantized representations remain optimal as the data distribution changes, avoiding performance degradation over time.
- **Exploration of Other Compression Techniques:** Investigating alternative or complementary compression techniques, such as knowledge distillation, sparse coding, or neural compression methods, could offer further improvements in storage and computational efficiency.
- **Application to Different Modalities or Domains:** Extending HPC-ColPali to other multimodal data types beyond visual documents (e.g., audio, video) or applying it to new domains with unique retrieval challenges would demonstrate its broader applicability and robustness.
- **Hardware Acceleration:** Exploring hardware-specific optimizations, such as leveraging specialized AI accelerators or custom hardware designs, could further boost the performance of HPC-ColPali, particularly for the binary encoding and Hamming distance computations.

REFERENCES

BAIDU (2024). Colqwen2.5 model. Hugging Face, 2024. Accessed: May 15, 2024.

Chen, Q. et al. (2021). Spann: Highly-efficient billion-scale approximate nearest neighbor search. In *NeurIPS*.

Facebook Research (2024). Faiss wiki. GitHub. Accessed: May 15, 2024.

Faysse, M. et al. (2024). Colpali: Efficient document retrieval with vision language models. *arXiv preprint arXiv:2407.01449*.

Gao, L. et al. (2023). Precise zero-shot dense retrieval without relevance labels. In *ACL*.

Ge, T. et al. (2013). Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755.

Gong, A. and Shi, T. (2020). Binary embeddings for faster retrieval: An overview. ScienceDirect, 2020. Accessed: May 15, 2024.

Goyal, N. et al. (2020). Power-bert: Accelerating bert inference via progressive layer dropping. In *arXiv preprint arXiv:2002.07881*.

Guo, R. et al. (2022). Accelerating large-scale inference with anisotropic vector quantization. In *ICML*.

Han, S. et al. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*.

Izacard, G. et al. (2021). Distilling knowledge from reader to retriever for question answering. In *ICLR*.

Izacard, G. and Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. In *ACL*.

Ji, Z., Lee, N., et al. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Jégou, H., Douze, M., and Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

Katz, D. et al. (2024). Gpt-4 passes the bar exam. *arXiv:2303.17012*.

Khattab, O., Sanctuary, H., and Potts, C. (2021). Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online and Punta Cana, Dominican Republic.

Koreeda, Y. and Manning, D. (2021). Contractnli: A dataset for document-level natural language inference for contracts. In *EMNLP*.

Lewis, P. et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Lin, J. et al. (2021). Batchneg: Efficient and effective training of retrieval models. In *SIGIR*.

Malkov, Y. and Yashunin, D. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Manning, C., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Muennighoff, N. et al. (2024). Generative representational instruction tuning. *arXiv:2402.09906*.

Nguyen, T. et al. (2016). Ms marco: A human generated machine reading comprehension dataset. In *arXiv preprint arXiv:1611.09268*.

Norouzi, M. et al. (2014). Fast exact search in hamming space with multi-index hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1107–1119.

Rao, Y. et al. (2021). Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*.

Sanathanam, K. et al. (2021). Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *NAACL*.

Tang, L. et al. (2023). Dynamicvit: Dynamic vision transformer without tuning and training. *arXiv preprint arXiv:2304.01186*.

Thakur, N., Reimers, N., et al. (2021). Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. In *arXiv preprint arXiv:2104.08663*.

Zhan, J., Mao, J., et al. (2021). Optimizing dense retrieval model training with hard negatives. In *SIGIR*.

Zhang, T. et al. (2020). Bertscore: Evaluating text generation with bert. In *ICLR*.

Zilliz (2023). Introducing colpali: The next-gen multimodal document retrieval model. Zilliz Blog, 2023. Accessed: May 15, 2024.