From Global to Regional Coverage: Hyperspherical Prototype Discovery for Storage-Efficient Multimodal Retrieval

Anonymous ACL submission

Abstract

Multi-vector visual document retrieval models encode pages into hundreds of patch-level embeddings, enabling fine-grained interaction but incurring substantial storage overhead. Existing train-free compression approaches rely on token pruning or heuristic merging, which often fail to preserve the broad semantic coverage of the original representations under constrained budgets. To address this, we propose CovMerge, a train-free compression framework that formulates token reduction as a semantic coverage problem on the hypersphere. Given frozen patch embeddings, CovMerge utilizes Farthest Point Sampling (FPS) to establish coverage-aware prototype initializations, followed by global spherical K-Means clustering. We introduce two complementary strategies: Global Coverage, which maximizes page-level semantic diversity directly, and Regional Coverage, which leverages document layout structures to allocate initial prototypes proportionally across distinct visual regions before global refinement. Operating purely as post-processing without retraining, CovMerge supports flexible deployment budgets. On the ViDoRe V1 benchmark, CovMerge retains up to 98.1% and 98.6% of the original nDCG@5 for ColPali and ColQwen2, respectively, while using only 10% of the original storage, establishing a new state-of-the-art for storage-efficient multimodal retrieval.

1 Introduction

Visual Document Retrieval (VDR) matches queries against layout-rich documents using late-interaction Vision-Language Models (VLMs) like ColPali (Faysse et al., 2025) and ColQwen2 (Xu et al., 2025). These models represent pages as dense, patch-level multi-vector embeddings, preserving fine-grained spatial and semantic details that single-vector encoders discard (Radford et al., 2021), thereby achieving state-of-the-art retrieval accuracy.

The storage bottleneck. This representational richness introduces severe memory overhead. For instance, ColPali produces 1,024 patch embeddings per page; indexing just 4,000 pages requires roughly 1 GB. Scaling to enterprise corpora creates prohibitive storage bottlenecks, making index compression essential for large-scale deployment.

Limitations of existing approaches. Current token reduction methods typically rely on pruning (Yan et al., 2025) or merging (Bolya et al., 2022). *Pruning* discards patches with low attention scores, risking the irreversible loss of "background" patches that might perfectly match fine-grained queries. *Merging* aggregates similar patches but suffers from poor initialization, causing *semantic collapse* where distinct minority details (e.g., small formulas) are swallowed by dominant visual features. To improve merging, some methods (Ma et al., 2025) resort to costly ratio-specific fine-tuning. Furthermore, recent train-free hybrid pipelines (Yan et al., 2026a) still inherit the query-agnostic flaws of the initial pruning stage and rigidly depend on predefined compression targets, limiting deployment flexibility.

Our approach. To address these challenges, we formulate index compression as a *budgeted semantic coverage* problem and propose **CovMerge**, a purely train-free post-processing framework for frozen VLM embeddings. To effectively mitigate semantic collapse (as illustrated in Figure 1), CovMerge utilizes Farthest Point Sampling (FPS) (Eldar et al., 1997) to discover diverse, coverage-aware prototype initializations. These are subsequently refined via global spherical K-Means clustering (McQueen, 1967; Dhillon and Modha, 2001) to align perfectly with the cosine similarity used in MaxSim retrieval.

Under this framework, we investigate two complementary strategies: **1) Global Coverage** applies FPS directly to all page patches, maximizing overall semantic diversity and establishing a

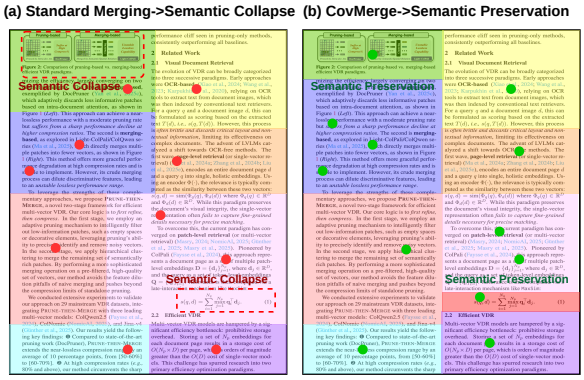

Figure 1: Motivation for budgeted semantic coverage. **(a) Standard Merging:** Similarity-driven merging leads to *semantic collapse*, where minority layout elements (red dashed boxes) are swallowed by dominant text features. **(b) CovMerge (Ours):** By discovering coverage-aware seeds via FPS, CovMerge allocates dedicated prototypes (green dots) to these distinct elements, achieving *semantic preservation* at high compression ratios.

robust, parser-free baseline. **2) Regional Coverage** utilizes a lightweight layout detector (DocLayout-YOLO) to allocate prototype budgets proportionally across semantic regions. Region-wise FPS seeds are generated first, followed by a *single* global spherical K-Means refinement, elegantly injecting layout priors without imposing brittle boundaries on the final merged tokens.

Contributions. Our main contributions are summarized as follows:

- We propose **CovMerge**, a simple yet highly effective, train-free multimodal index compression framework formulated as a budgeted semantic coverage problem. It features two variants: **Global Coverage**, which establishes a strong baseline via FPS-initialized clustering, and **Regional Coverage**, which leverages lightweight layout parsing to structurally guide prototype allocation.
- We demonstrate that our semantic merging process strictly preserves the MaxSim scoring property at query time, enabling elegant, plug-and-play integration into existing late-interaction VDR pipelines without any architecture modifications.
- Extensive experiments on the ViDoRe V1 benchmark validate CovMerge's cross-architecture effectiveness. At a mere 10% storage budget, it retains up to 98.1% and 98.6% of the original nDCG@5 performance for ColPali and ColQwen2, respectively. This establishes a new state-of-the-art for train-free, storage-efficient multimodal retrieval.

2 Related Work

2.1 Multi-Vector Visual Document Retrieval

The late-interaction paradigm, popularized by ColBERT (Khattab and Zaharia, 2020), represents queries and documents as sets of token-level embeddings, matching them via a MaxSim operation to achieve superior recall over single-vector dense retrieval. ColPali (Faysse et al., 2025) successfully extended this to layout-rich Visual Document Retrieval (VDR) by encoding document pages as grids of patch embeddings via PaliGemma (Beyer et al., 2024). ColQwen2 (Xu et al., 2025) further advanced VDR using a Qwen2-VL backbone with dynamic-resolution encoding(Wang et al., 2024). While establishing a new state-of-the-art on benchmarks like ViDoRe (Faysse et al., 2025), these architectures mandate storing hundreds of vectors per page, creating severe deployment bottlenecks that necessitate effective index compression.

2.2 Compression via Token Pruning and Merging

Existing token reduction methods typically operate via pruning or merging (Rao et al., 2021).

Pruning-based methods discard seemingly uninformative patches. DocPruner (Yan et al., 2025) relies on self-attention scores to filter tokens, while Look-in-the-Middle (Liu et al., 2026) utilizes intermediate-layer attention maps. Although effective at moderate compression rates, pruning suffers from a fundamental flaw: it is inherently query-agnostic. A patch deemed "unimportant" by self-attention might be the exact MaxSim target for a highly specific query.

Merging-based methods address this by aggregating tokens to preserve broader context. While ToMe (Bolya et al., 2022) introduced bipartite matching during the forward pass, Light-ColPali (Ma et al., 2025) explored post-processing strategies, finding that agglomerative semantic clustering outperforms simple spatial pooling. However, clustering all patches jointly often leads to *centroid collapse*, where semantically distinct layout regions (e.g., a tiny formula next to a large paragraph) are erroneously fused. Furthermore, Light-ColPali's optimal performance requires ratio-specific fine-

tuning, severely limiting deployment flexibility. Recent hybrid pipelines like Prune-then-Merge (Yan et al., 2026a) extend the compression range but still inherit the query-agnostic limitations of the initial pruning stage.

Unlike these methods, **CovMerge** avoids query-agnostic token dropping by formulating compression as a budgeted semantic coverage problem, utilizing Farthest Point Sampling and spherical clustering to ensure broad, distinct semantic support.

2.3 Layout-Aware Processing

Document Layout Analysis models, such as the highly efficient DocLayout-YOLO (Zhao et al., 2024), identify structural elements (paragraphs, tables, figures) with minimal overhead. The value of layout in VDR has been explored by architecture-level methods like ColParse (Yan et al., 2026b), which segments pages via heavy parsers (MinerU2.5(Niu et al., 2025)) for independent encoding. However, ColParse is a complex, encoding-level pipeline that cannot compress existing pre-computed indices. In contrast, our **Regional Coverage** strategy elegantly leverages lightweight layout detection purely as a structural prior to allocate prototype initialization budgets, ensuring layout awareness in a fully train-free, post-processing paradigm.

3 Preliminaries

3.1 Multi-Vector Visual Document Retrieval

Let $\mathcal{C} = \{d_1, d_2, \ldots, d_{|\mathcal{C}|}\}$ denote a corpus of document pages, where each page d is represented as a raster image. Given a textual query q, the task of Visual Document Retrieval (VDR) is to rank pages in $\mathcal{C}$ by relevance to q.

In the multi-vector paradigm (Faysse et al., 2025), a frozen Vision-Language Model (VLM) encoder Φ maps each document image into a set of N_p patch-level embeddings $\mathcal{D} = \Phi(d) = \{d_j\}_{j=1}^{N_p}$ (where $d_j \in \mathbb{R}^k$), and maps the query into a set of N_q token-level embeddings $\mathcal{Q} = \Phi(q) = \{q_i\}_{i=1}^{N_q}$ (where $q_i \in \mathbb{R}^k$). All embeddings are L_2-normalized (Liu et al., 2015) to lie on the unit hypersphere. For ColPali, $N_p = 1024$; for ColQwen2, N_p is dynamic and depends on the image resolution.

3.2 Late Interaction Scoring (MaxSim)

Relevance between a query q and a document d is computed via the MaxSim operator (Khattab and Zaharia, 2020):

$$s(q, d) = \sum_{i=1}^{N_q} \max_{j=1}^{N_p} q_i^\top d_j \tag{1}$$

Each query token finds its single best-matching document patch; overall relevance is the sum of these per-token maxima. This fine-grained token-to-patch alignment allows specific query terms to be matched to precise spatial regions within the document, yielding superior retrieval accuracy on layout-rich pages compared to single-vector dot-product retrieval.

3.3 Compression Goal

Storing the full multi-vector index for a corpus requires $|\mathcal{C}| \times N_p \times k \times b$ bits, where b is the bit precision (typically 16 for bfloat16). The goal of token compression is to produce a reduced representation $\mathcal{D}' = \{d'_l\}_{l=1}^{N'_p}$ with $N'_p \ll N_p$, such that the MaxSim score under $\mathcal{D}'$ closely approximates the original: $s(q, \mathcal{D}') \approx s(q, \mathcal{D})$ for all q. We define the *retained storage percentage* as $\rho = (N'_p/N_p) \times 100$ and evaluate methods at target retention rates of 5%, 10%, and 20%.

4 Methodology

Figure 2 summarizes CovMerge. Given a document page image d and frozen patch embeddings $\mathbf{D} = \{\mathbf{d}_j\}_{j=1}^{N_p}$, CovMerge outputs $\mathbf{D}' = \{\mathbf{d}'_\ell\}_{\ell=1}^{K}$ with $K = \lfloor \rho N_p \rfloor$ for a user-specified retention ratio ρ. All operations are offline post-processing; the query encoder and MaxSim scorer remain unchanged.

To accommodate diverse computational constraints and document types, CovMerge operates in two complementary modes: **1) Global Coverage**, a minimalist baseline that bypasses layout parsing and directly initializes prototypes globally; and **2) Regional Coverage**, which introduces Stages 1 to 3 below to structurally guide prototype allocation using a lightweight layout prior.

4.1 Stage 1: Document Layout Detection (Regional Coverage)

(This stage is specific to Regional Coverage; Global Coverage skips directly to Stage 4.)

We use DocLayout-YOLO to detect a set of axis-aligned bounding boxes with semantic labels from the document page image. Each detected region is represented as a tuple $r =$

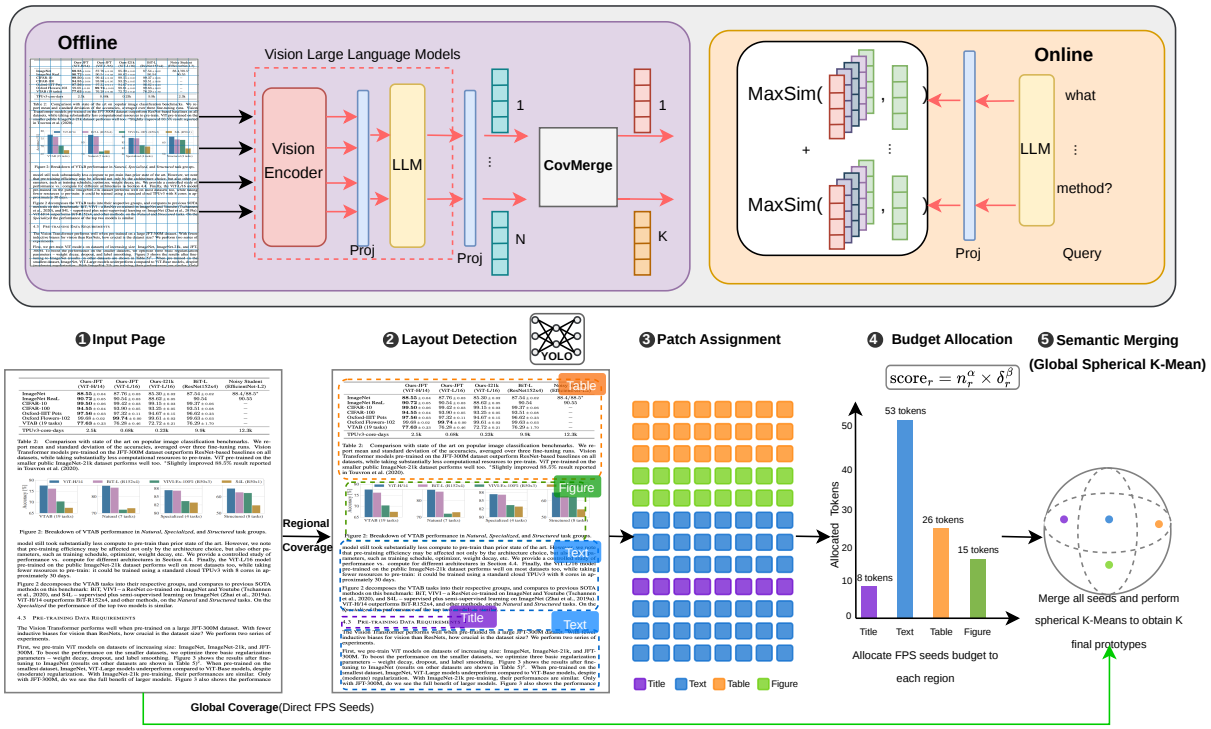

Figure 2: Overview of CovMerge. **Top:** The overall retrieval architecture, integrating CovMerge into the visual encoding pipeline during offline processing, while preserving MaxSim scoring at query time (online). **Bottom:** Detailed workflows of the CovMerge module. **Regional Coverage** (main path): DocLayout-YOLO detects regions; patch assignment and budget allocation determine how many FPS seeds each region contributes. **Global Coverage** (bottom bypass): FPS selects K page-level seeds directly, bypassing layout parsing. In both variants, a *single* global spherical K-Means refines all K output prototypes into a global initialization. Both variants are train-free.

$(\mathrm{id}_r, \mathbf{b}_r, \ell_r, c_r)$, where $\mathbf{b}_r = (x_0, y_0, x_1, y_1)$ is the bounding box in page pixel coordinates, $\ell_r \in \{\texttt{title}, \texttt{text}, \texttt{table}, \texttt{figure}, \texttt{caption}, \ldots\}$ is the layout category, and $c_r \in (0, 1]$ is the detector confidence score.

Regions with zero area or zero confidence are discarded. Each surviving bounding box is expanded by a small page-relative margin (default 3%) to account for detector boundary imprecision, and then clamped to the page boundary. The set of regions is capped at a maximum of 64 (ordered by confidence $\times$ area) to prevent pathological cases with tiny boxes from exhausting the budget.

Fallback. When the detector returns no valid regions, Regional Coverage falls back to Global Coverage. Patches not contained by any detected box are assigned to a background bucket `__background__`.

4.2 Stage 2: Patch-to-Region Assignment (Regional Coverage)

(This stage is specific to Regional Coverage; Global Coverage skips directly to Stage 4.)

ColPali/ColQwen2 produce patch embeddings arranged on a regular spatial grid. For ColPali v1.3 the grid is fixed at 32×32; for ColQwen2 v1.0 the grid dimensions (H, W) are read from the processor's `image_grid_thw` field and vary per image after spatial merging.

The center pixel coordinate of patch (i, j) in the grid is $x_{ij} = (j + 0.5)W_{\text{page}}/W$ and $y_{ij} = (i + 0.5)H_{\text{page}}/H$, where W_{page} and H_{page} are the original image dimensions in pixels. Each patch is assigned to the *smallest* detected region that contains its center (favoring specific layout boxes over broad enclosing ones). Patches whose center falls inside no detected region are assigned to the background bucket. This yields a partition $\{P_r\}$ where $P_r \subseteq \{1, \ldots, N_p\}$ indexes the patches belonging to region r.

4.3 Stage 3: Budget Allocation (Regional Coverage Only)

Global Coverage skips this stage. For Regional Coverage, we determine how many FPS seeds each region contributes before global refinement. We first reserve a minimum of one seed per non-background region. The remaining budget K' is

distributed proportionally based on a per-region score:

$$s_r = n_r^{\alpha} \cdot (\delta_r + \epsilon)^{\beta}, \tag{2}$$

where $n_r = |P_r|$ is the number of patches assigned to region r, and δ_r represents the intra-region embedding diversity. It is mathematically defined as the mean squared deviation of the region's patch embeddings from their centroid $\mu_r = \frac{1}{n_r}\sum_{d_j \in P_r} d_j$:

$$\delta_r = \frac{1}{n_r} \sum_{d_j \in P_r} |d_j - \mu_r|_2^2. \tag{3}$$

The constant $\epsilon{=}10^{-4}$ prevents degenerate scores for singleton regions (where $\delta_r = 0$). Default exponents are $\alpha{=}1$ and $\beta{=}1$. Integer seed counts are computed via largest-remainder rounding. To prevent over-allocation, the maximum budget for any region is strictly capped at its actual patch count n_r. Any excess seeds resulting from this cap are iteratively reallocated to the nearest unsaturated regions to ensure the full budget K' is strictly exhausted.

4.4 Stage 4: Coverage-Aware Prototype Discovery and Refinement

CovMerge represents each output slot as a prototype on the unit hypersphere. Both variants share the same global refinement step; they differ only in how prototypes are initialized.

Global Coverage. We select K farthest-point sampling (FPS) seeds from all L2-normalized page embeddings, then run spherical K-Means for $T{=}12$ Lloyd iterations on the full patch set, using the FPS seeds as initial centroids. *This variant serves as a highly robust, universally applicable baseline that requires zero external dependencies.* Each cluster is represented by the L2-normalized mean of its members. When $K{=}1$, we return the medoid patch.

Regional Coverage. For each region r with allocated seed count b_r, we run FPS *inside* P_r to obtain b_r diverse seeds. Seeds from all regions are concatenated (total K) to form a *layout-conditioned initialization*. We then run the *same* global spherical K-Means on all page patches—not independent per-region clustering. This design lets prototypes move across region boundaries during refinement while still biasing initialization toward layout-aware coverage.

Why global refinement? Per-region clustering hard-separates outputs and is sensitive to detector errors (missed boxes, overlapping labels). Global refinement preserves a single K-partition of the page while letting layout influence *where* prototypes start, which we find more robust in practice (Table 1).

4.5 Complexity and Overhead

Layout detection with DocLayout-YOLO adds approximately 30 ms per page on CPU and is fully parallelizable across the corpus. The patch assignment is $\mathcal{O}(N_p M)$ where $M \leq 64$ is the number of regions. Budget allocation is $\mathcal{O}(M)$. Spherical K-Means runs in $\mathcal{O}(N_p \cdot K \cdot T)$ with $T{=}12$ Lloyd iterations over all page patches, independent of whether seeds come from whole-page FPS or region-wise FPS. The total indexing overhead relative to the ColPali encoding pass is negligible.

5 Experiments

5.1 Setup

Backbones. We evaluate on **ColPali-v1.3** (PaliGemma-3B, 32×32 grid, $d{=}128$) and **ColQwen2-v1.0** (Qwen2-VL-2B, dynamic-resolution grid, $d{=}128$), the two dominant backbones on ViDoRe-v1.

Benchmark and Metrics. We report nDCG@5 across all 10 test splits of the ViDoRe benchmark, comprising ArxivQA, DocVQA, InfoVQA, TabFQuAD, TAT-DQA, ShiftProject, and four SyntheticDocQA tasks. While our primary evaluation is conducted at a tight storage budget of $\rho{=}10\%$, we additionally test across a wider range of compression ratios $\rho \in \{5\%, 20\%\}$ against six competitive baselines to evaluate scaling robustness (see Figure 3).

Baselines. We evaluate both variants of our approach (CovMerge-Global and CovMerge-Regional) against the uncompressed *dense* index and six representative train-free compression baselines, categorized as follows: (i) **Naive approaches:** Random Pruning and Token Pooling; (ii) **Pruning-based:** Adaptive DocPruner (Yan et al., 2025) and SAP; (iii) **Merging-based:** Train-free Light-ColPali (Ma et al., 2025); (iv) **Hybrid:** Prune-then-Merge (Yan et al., 2026a).

Implementation. Layout detection uses DocLayout-YOLO. CovMerge (Global) uses Global Coverage; CovMerge (Regional) uses

Regional Coverage with default $(\alpha, \beta)=(1,1)$ unless ablated. All methods share the same frozen encoders and MaxSim scoring.

5.2 Evaluation Metrics

We report standard retrieval metrics evaluated at rank cutoff K. Recall@K measures whether the relevant document appears within the top-K retrieved results. nDCG@K (Normalized Discounted Cumulative Gain) accounts for ranking position (Järvelin and Kekäläinen, 2002):

$$\text{nDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}, \tag{4}$$

where $\text{DCG@K} = \sum_{i=1}^{K} \text{rel}_i / \log_2(i+1)$, $\text{rel}_i \in \{0,1\}$ is the binary relevance of the i-th retrieved document, and IDCG@K is the ideal maximum DCG. We use nDCG@5 as our primary metric, consistent with the ViDoRe benchmark.

5.3 Main Results

Table 1 reports the per-dataset nDCG@5 at a tight 10% storage budget. Notably, our **CovMerge** framework establishes a new state-of-the-art across both backbones. On the stronger ColQwen2 backbone, **CovMerge (Global)** outperforms the strongest baseline, Prune-then-Merge, by a solid margin (+1.8 points on average, 86.8 vs. 85.0). Furthermore, when compared exclusively to the prior train-free pure merging method (Light-ColPali), CovMerge demonstrates massive improvements (+4.4 and +5.0 points), validating that coverage-aware initialization and L2-normalized refinement are vastly superior to naive hierarchical clustering.

Regarding our two variants, **CovMerge (Regional)** excels on specific datasets where layout priors are crucial (e.g., Shift on ColQwen2: 86.0 vs. 84.6; Shift on ColPali: 74.6 vs. 72.7). Meanwhile, the Global variant remains the strongest on average for ColQwen2 (retaining 98.6% of the dense performance). On ColPali, both variants perform comparably well, retaining 98.0% and 98.1% of the upper bound, respectively. Notably, while Prune-then-Merge manages to tie our performance on the smaller ColPali model, it clearly falls behind on the advanced ColQwen2 architecture, where CovMerge proves its superior scalability.

5.4 Effect of Storage Ratio

To evaluate scalability, Figure 3 plots the mean nDCG@5 across varying retention ratios $\rho \in \{5\%, 10\%, 20\%\}$. CovMerge degrades gracefully

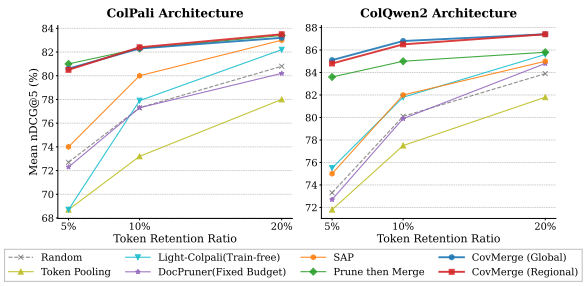

Figure 3: Overall retrieval performance (Mean nDCG@5) across varying token retention ratios (ρ) on ColPali v1.3 and ColQwen2. Both CovMerge (Global) and CovMerge (Regional) consistently outperform the competitive baselines, maintaining near-lossless accuracy even under an extreme 5% storage budget.

across all settings. Remarkably, even under an extreme 5% storage budget, both variants maintain above 80% mean nDCG@5 on ColPali and 84% on ColQwen2, significantly outperforming existing baselines. At a more relaxed 20% budget, CovMerge approaches the uncompressed dense upper bound ($\geq$87.3% on ColQwen2).

5.5 Comparison with Fine-Tuned Methods

While train-free compression is highly desirable for deployment, fine-tuning after merging can theoretically recover additional accuracy. However, this comes at the cost of substantial training overhead and architectural inflexibility (i.e., requiring a separate checkpoint per compression ratio). To evaluate this efficiency-performance trade-off, we benchmark our train-free CovMerge against the fully fine-tuned Light-ColPali.

Experimental Setup. Following the original Light-ColPali study, we restrict this evaluation to their exact 6-dataset subset (*InfoVQA, DocVQA, ArxivQA, TabFQuAD, TAT-DQA, and Shift*). Although baseline compression intervals slightly differ ($x4/x9/x25$ vs. our $x5/x10/x20$), they serve as functionally equivalent tiers for mild, moderate, and aggressive budgets.

Results. As shown in Table 2, our train-free CovMerge achieves retention rates remarkably close to, and sometimes exceeding, the fine-tuned baseline. On ColQwen2, CovMerge trails by a marginal $< 1.4\%$, bypassing the heavy retraining pipeline entirely. More impressively, on ColPali under mild compression, both CovMerge variants (**99.3%** and **99.5%**) outperform the fine-tuned model (**98.8%**). Even under aggressive compression, CovMerge-whole (**95.4%**) surpasses the

Table 1: Performance comparison on the ViDoRe benchmark. Models are evaluated using nDCG@5 (%). The best performance among the compression methods is highlighted in **bold**, while the uncompressed dense baseline is provided as an upper bound. The subscripts in the Average column indicate the performance retention relative to the dense baseline under x10 compression ratio.

Method	ViDoRe Datasets										
	Arxiv	**Doc**	**Info**	**TabF**	**TAT**	**Shift**	**AI**	**Energy**	**Gov**	**Health**	**Avg.**
ColPali v1.3 (Base: PaliGemma-3B)											
Dense (Upper Bound)	*82.6*	*53.8*	*84.6*	*87.4*	*69.5*	*77.0*	*96.6*	*95.0*	*95.7*	*97.9*	$84.0_{100\%}$
Random Pruning	77.4	41.2	77.4	82.5	47.1	58.6	86.6	83.3	83.7	89.4	$72.7_{86.5\%}$
Token Pooling	68.3	31.0	74.3	79.7	47.3	47.8	87.6	79.9	84.7	86.3	$68.7_{81.8\%}$
Light-Colpali(Train-free)	80.5	46.9	81.2	86.6	60.1	57.7	92.0	90.9	90.5	92.4	$77.9_{92.7\%}$
DocPruner(Fixed Budget)	78.9	46.2	80.2	86.1	54.5	63.5	92.5	89.6	90.6	91.5	$77.3_{92.0\%}$
SAP	80.0	52.0	81.0	85.0	62.0	64.0	93.0	91.0	**95.0**	93.0	$79.6_{94.8\%}$
Prune-then-Merge	81.9	51.5	84.0	**87.2**	**66.7**	74.0	97.1	**94.1**	92.5	94.4	$82.3_{98.0\%}$
CovMerge(Global)	**82.2**	**51.9**	**84.2**	87.0	66.6	72.7	**97.7**	93.4	92.3	95.4	$82.3_{98.0\%}$
CovMerge(Regional)	81.8	**51.9**	84.1	87.0	65.5	**74.6**	97.1	92.5	92.8	**96.9**	$\mathbf{82.4_{98.1\%}}$
ColQwen v1.0 (Base: Qwen2-VL-2B)											
Dense (Upper Bound)	*87.8*	*54.3*	*91.3*	*88.7*	*81.0*	*89.1*	*98.8*	*96.3*	*94.8*	*98.4*	$88.0_{100\%}$
Random Pruning	84.0	45.7	82.1	84.2	63.3	72.8	93.4	88.6	90.9	95.6	$80.1_{91.0\%}$
Token Pooling	78.4	40.1	81.1	82.8	63.8	63.8	92.7	90.0	91.4	90.6	$77.5_{88.1\%}$
Light-Colpali(Train-free)	85.7	49.7	84.5	87.7	69.3	69.5	95.8	93.8	89.8	92.6	$81.8_{93.0\%}$
DocPruner(Fixed Budget)	82.9	43.4	84.4	83.9	64.3	71.0	95.7	90.0	92.7	93.2	$80.2_{91.1\%}$
SAP	79.0	52.0	84.0	84.0	69.0	81.0	93.0	90.0	91.0	97.0	$82.0_{93.2\%}$
Prune-then-Merge	86.0	52.5	86.9	88.7	73.9	82.7	96.5	93.5	92.3	96.7	$85.0_{96.6\%}$
CovMerge(Global)	**86.8**	**52.9**	88.5	**89.4**	77.1	84.6	98.1	95.9	**95.9**	**98.4**	$\mathbf{86.8_{98.6\%}}$
CovMerge(Regional)	86.2	52.1	**88.5**	88.3	**77.3**	**86.0**	**98.3**	**96.3**	94.2	97.5	$86.5_{98.3\%}$

Table 2: Performance retention analysis on the 6-dataset subset of ViDoRe. The values represent the percentage (%) of the original uncompressed nDCG@5 retained. Results are grouped by compression severity: Mild (∼x4–x5), Moderate (∼x9–x10), and Aggressive (∼x20–x25).

Architecture	Method	Retention (%)		
		Mild	**Moderate**	**Aggressive**
ColQwen2	Light-ColPali (Trained)	**99.5**	**98.7**	**97.4**
	CovMerge(Global) (*Train-free*)	98.8	97.4	95.4
	CovMerge(Regional) (*Train-free*)	98.8	97.2	94.5
ColPali	Light-ColPali (Trained)	98.8	**98.1**	95.1
	CovMerge(Global) (*Train-free*)	99.3	97.7	**95.4**
	CovMerge(Regional) (*Train-free*)	**99.5**	97.8	95.3

trained baseline (**95.1%**). This strongly validates our budgeted semantic coverage formulation: advanced prototype discovery can effectively eliminate the need for costly ratio-specific fine-tuning.

5.6 Sensitivity Analysis and Algorithmic Robustness

Prune-then-Merge mechanisms rely heavily on a Stage-1 hard thresholding gate ($\tau = \mu + k\sigma$) to discard tokens. However, empirical analysis reveals a high sensitivity to this hyperparameter. As quantified in Table 3, shifting k from the optimal -0.25 to 0.0 triggers a severe performance degradation, dropping the average nDCG@5 by **6.2** points on ColPali and **5.5** points on ColQwen2.

In contrast, CovMerge completely circumvents this vulnerability. Operating as a purely train-free post-processing step, CovMerge requires zero thresholding parameter sweeps. Users simply specify the continuous target retention ratio. Under the same 10% storage budget, both CovMerge variants consistently deliver stable, near-lossless SOTA performance without hyperparameter tuning, highlighting their robustness for practical deployment.

5.7 Ablation Studies

We conduct rigorous ablation studies across three design axes to validate the core components of our framework: prototype initialization strategy, budget allocation parameters, and the choice of the foundational clustering algorithm. All ablations are evaluated at a 10% target retention ratio.

Initialization and Distance Metrics. Table 4 isolates the effects of initialization and geometric

Table 3: Sensitivity analysis on the ViDoRe benchmark at a 10% storage budget. Prune-then-Merge is highly sensitive to its thresholding parameter k, whereas CovMerge delivers stable performance by simply specifying the target compression ratio.

Architecture	Method & Setting	Avg. nDCG@5	Status
ColQwen2	Prune-then-Merge ($k = -0.25$)	**85.0**	Optimal tuned
	Prune-then-Merge ($k = 0.0$)	79.5	Drop (-5.5)
	CovMerge (Global)	86.8	**Robust (No tuning)**
	CovMerge (Regional)	86.5	**Robust (No tuning)**
ColPali	Prune-then-Merge ($k = -0.25$)	**82.3**	Optimal tuned
	Prune-then-Merge ($k = 0.0$)	76.1	Drop (-6.2)
	CovMerge (Global)	82.3	**Robust (No tuning)**
	CovMerge (Regional)	**82.4**	**Robust (No tuning)**

Table 4: Ablation study of initialization and distance metrics for the global K-Means refinement (Mean nDCG@5, %).

Initialization	Metric	ColPali	ColQwen2
Random	Euclidean	81.6	86.0
Random	Spherical	81.8	86.0
K-means++	Spherical	82.1	86.3
FPS (Ours)	**Spherical**	**82.4**	**86.8**

space for the global K-Means refinement. Transitioning from Euclidean to Spherical space yields minor improvements, aligning the clustering geometry with the downstream MaxSim dot-product scoring. However, the most significant gain stems from the initialization strategy. Farthest Point Sampling (FPS) consistently outperforms Random and K-means++ initialization, confirming that ensuring broad semantic coverage from the outset is crucial for preserving minority elements under tight storage budgets.

Table 5: Sensitivity to budget allocation exponents (α, β) in Eq. (2). Results indicate mean nDCG@5 at 10% storage.

α (Size)	β (Diversity)	ColPali	ColQwen2
0.65	0.85	82.2	86.5
1.00	1.00	**82.4**	86.5
0.85	0.65	82.1	**86.8**

Budget Allocation Parameters. For the Regional variant, Table 5 evaluates the sensitivity to the exponents (α, β) governing the allocation of FPS seeds based on region size and semantic diversity (Eq. 2). Modifying these hyperparameters causes a maximum variance of only 0.3 nDCG@5 points from our default setting of $(\alpha, \beta) = (1, 1)$. This demonstrates that the specific exponent values are less critical than the underlying heuristic: regions with larger patch counts or higher internal variance inherently require more prototypes.

Table 6: Comparison of clustering families (HAC vs. K-Means) across architectures. To ensure a rigorous baseline, HAC is also implemented with L2-normalized representations.

Architecture	Method	Regional	Global
ColPali	HAC (L2-Norm)	80.5	81.3
	CovMerge (Ours)	**82.4**	**82.3**
ColQwen2	HAC (L2-Norm)	83.8	85.4
	CovMerge (Ours)	**86.5**	**86.8**

Foundational Clustering Algorithm. Standard token merging methods often employ Hierarchical Agglomerative Clustering (HAC). In Table 6, we contrast HAC with our CovMerge paradigm (FPS + global Spherical K-Means). To establish a highly competitive baseline, we enhance HAC by enforcing L2-normalization on its inputs and merged centroids, ensuring it also operates on the unit hypersphere. Despite this enhancement, CovMerge dominates the HAC baseline by a substantial margin of 1.0 to 2.7 absolute nDCG@5 points across both architectures. This confirms that iterative global refinement from coverage-aware seeds is fundamentally superior to greedy agglomerative merging, which often suffers from irreversible suboptimal early merges (semantic collapse).

6 Conclusion

We presented CovMerge, a train-free framework for compressing ColPali-family document indices via budgeted semantic coverage on the hypersphere. FPS initialization combined with global spherical K-Means yields a formidable Global Coverage baseline. Regional Coverage further injects layout structure through budgeted, region-wise seed allocation prior to the same global refinement step. Evaluated on the ViDoRe benchmark at a 10% storage budget, CovMerge retains up to 98.1% and 98.6% of the dense nDCG@5 performance for ColPali and ColQwen2, respectively. Notably, it establishes a new state-of-the-art for train-free compression and rivals heavier pruning-then-merging pipelines, all without requiring ratio-specific fine-tuning. Future work includes exploring learned layout priors, joint quantization of prototypes, and extensions to multi-page document reasoning.

Limitations

While CovMerge establishes a robust train-free framework for multimodal index compression, we acknowledge several limitations that provide avenues for future research:

Dependency on External Parsers (Regional Variant). Our Global Coverage strategy operates independently of any external tools. However, the Regional Coverage variant relies on an off-the-shelf layout detector (DocLayout-YOLO) to inject structural priors. Although lightweight ($\sim$30 ms per page on CPU), this introduces an additional preprocessing step that breaks the end-to-end nature of the VLM encoding pipeline. Furthermore, the effectiveness of the layout-guided seed initialization is inherently upper-bounded by the accuracy of the underlying detector; missed bounding boxes or misclassified regions in highly unconventional documents could lead to sub-optimal budget allocation, though our global refinement step partially mitigates this.

Inability to Adapt the Embedding Space. CovMerge is designed as a strictly train-free, post-processing method operating on frozen patch embeddings. While this plug-and-play nature ensures zero training overhead and immediate deployment across varying compression budgets, it inherently lacks the capacity to alter or project the original embedding space. In extremely aggressive compression scenarios (e.g., $< 5\%$ retention), methods that fine-tune projection layers (e.g., trained LightColPali) might theoretically disentangle overlapping semantics better than purely spatial prototype refinement.

Heuristic Nature of Budget Allocation. In the Regional Coverage variant, the allocation of FPS seeds is governed by hyperparameters (α and β) that weigh region size and semantic diversity. Although our empirical sensitivity analysis (Section 5.7) demonstrates that CovMerge is highly robust to variations in these exponents across standard benchmarks, they remain dataset-agnostic heuristics. Highly specialized domains with extreme layout distributions (e.g., dense mathematical manuscripts versus sparse infographics) might benefit from dynamically learned allocation priors rather than static exponential scaling.

Ethics Statement

This work uses the publicly released ViDoRe benchmark (Faysse et al., 2025) and pretrained ColPali (Faysse et al., 2025) / ColQwen2 (Xu et al., 2025) weights. We do not introduce new datasets, human annotation, or human evaluation. CovMerge operates exclusively on frozen patch embeddings via post-processing and does not modify the encoder, so it does not introduce additional fairness, privacy, or harmful-content risk beyond that already inherent in the underlying vision–language models. By design, compression selects which patch statistics are retained; in principle, an adversary could attempt to exploit this selection to bias retrieval, though we are not aware of a concrete attack vector. We will release code, configuration, and analysis scripts upon acceptance to support reproducibility.

References

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, and 1 others. 2024. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

Inderjit S Dhillon and Dharmendra S Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1):143–175.

Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. 1997. The farthest point strategy for progressive image sampling. *IEEE transactions on image processing*, 6(9):1305–1315.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. In *International Conference on Learning Representations*, volume 2025, pages 61424–61449.

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

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Wei Liu, Andrew Rabinovich, and Alexander C Berg. 2015. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*.

Zhuchenyang Liu, Ziyu Hu, Yao Zhang, and Yu Xiao. 2026. Look in the middle: Structural anchor pruning for scalable visual rag indexing. *arXiv preprint arXiv:2601.20107*.

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 *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19568–19580.

James B McQueen. 1967. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297.

Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, and 1 others. 2025. Mineru2. 5: A decoupled vision-language model for efficient high-resolution document parsing. In *The 64th Annual Meeting of the Association for Computational Linguistics–Industry Track*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

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, Mingdong Ou, Yi Cao, Xin Zou, Jiahao Huo, Shuliang Liu, James Kwok, and Xuming Hu. 2026a. Sculpting the vector space: Towards efficient multi-vector visual document retrieval via prune-then-merge framework. *arXiv preprint arXiv:2602.19549*.

Yibo Yan, Mingdong Ou, Yi Cao, Xin Zou, Shuliang Liu, Jiahao Huo, Yu Huang, James Kwok, and Xuming Hu. 2026b. Beyond the grid: Layout-informed multi-vector retrieval with parsed visual document representations. *arXiv preprint arXiv:2603.01666*.

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*.

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: ViDoRe-v1 Benchmark Details

Datasets. We evaluate our methods on all 10 datasets from the ViDoRe-v1 benchmark (Faysse et al., 2025), covering a broad spectrum of visually rich document retrieval scenarios. As summarized in Table 7, the benchmark includes two complementary categories.

Academic Tasks contain naturally occurring documents with highly diverse layouts, including scanned academic papers (ArxivQA), infographic-style visual reasoning documents (InfoVQA), financial tables and reports (TabFQuAD), scientific charts and figures (TAT-QA), and web-style tabular documents (ShiftProject). These datasets emphasize fine-grained alignment between textual queries and localized visual regions such as tables, captions, formulas, and dense figure structures.

Practical Tasks consist of industry-oriented synthetic documents spanning domains such as energy, government, healthcare, and artificial intelligence. Compared to academic datasets, these documents typically exhibit cleaner layouts but stronger domain-specific terminology and template regularity, providing a complementary evaluation setting for retrieval robustness under structured enterprise-style documents.

Overall, ViDoRe-v1 presents substantial variation in document layout complexity, semantic density, OCR characteristics, and region granularity, making it a challenging benchmark for studying storage-efficient multi-vector retrieval.

Appendix B: Detailed Pseudocode for CovMerge

We present the formal algorithmic pipeline for both `CovMerge(Global)` and `CovMerge(Regional)` in Algorithm 1. The framework accommodates arbitrary target storage budgets completely train-free, operating purely as an indexing post-processing step.

Dataset	Language	# Queries	# Documents	Description
			Academic Tasks	
DocVQA	English	500	500	Scanned documents from UCSF Industry
InfoVQA	English	500	500	Infographics scrapped from the web
TAT-DQA	English	1663	277	High-quality financial reports
arXiVQA	English	500	500	Scientific Figures from arXiv
TabFQuAD	French	280	70	Tables scrapped from the web
			Practical Tasks	
Energy	English	100	977	Documents about energy
Government	English	100	972	Administrative documents
Healthcare	English	100	965	Medical documents
AI	English	100	968	Scientific documents related to AI
Shift Proj.	French	100	1000	Environmental reports

Table 7: Overview of the 10 datasets from the ViDoRe benchmark used in our evaluation.

Appendix C: Intermediate Visualization of CovMerge(region)

To further illustrate how layout structural priors steer prototype discovery, we provide step-by-step feature map visualizations. To provide an intuitive understanding of the `CovMerge(Regional)` pipeline, we visualize both the prerequisite layout detection phase and the subsequent adaptive prototype budget allocation mechanism.

Step 1: Layout Detection via DocLayout-YOLO. Our region-guided merging strategy relies on robust semantic parsing of the document page. Figure 4 demonstrates the effectiveness of the off-the-shelf DocLayout-YOLO engine across diverse distributions within the ViDoRe benchmark. Whether parsing complex multi-column reports with interleaved charts from the *ShiftProject* dataset (left) or dense archival forms from *DocVQA* (right), the layout engine accurately segments the page into semantically coherent bounding boxes (e.g., text blocks, figures, tables, and titles). This precise spatial segmentation forms the foundation for our diversity-aware merging.

Step 2: Adaptive Prototype Budget Allocation. Building upon the detected regions, we visualize the exact distribution of prototypes under the `CovMerge(region)` strategy. As formalized in Equation 2, the budget assigned to each detected region is not naively proportional to its physical area. Instead, it is determined by a carefully calibrated balance between the region's patch count (n_r) and its intra-region embedding diversity (δ_r).

Figure 5 illustrates this budget allocation on two representative document pages using the ColPali architecture at a 10% storage ratio (yielding exactly 102 tokens per page).

- **Page (A) - Table-Dense Document:** The diversity-aware nature of our scoring function (s_r) is clearly demonstrated here. Although *Table #0* might not occupy the largest physical area, it exhibits extremely high structural variance and semantic density (δ_r). Consequently, the algorithm automatically prioritizes it, allocating the maximum budget of 16 prototypes to preserve its intricate tabular details.

- **Page (B) - Figure & Title Document:** The allocation naturally concentrates the budget on core semantic elements such as *Figures* (11–12 prototypes) and essential *Text Segments* (11–14 prototypes). In contrast, redundant or uninformative layout noise, classified as *Background / Header*, is heavily penalized by the diversity term and receives minimal budget (e.g., 5–7 prototypes).

These intermediate visualizations empirically validate that `CovMerge(region)` acts as an intelligent, semantics-driven filter. By seamlessly integrating layout priors with embedding variance, it strategically concentrates the constrained budget on information-dense areas, ensuring high retrieval performance even under aggressive compression.

Appendix D: Full Quantitative Results

For completeness, we report the absolute nDCG@5 scores across all 10 individual datasets for all compression scales (5% and 20%).

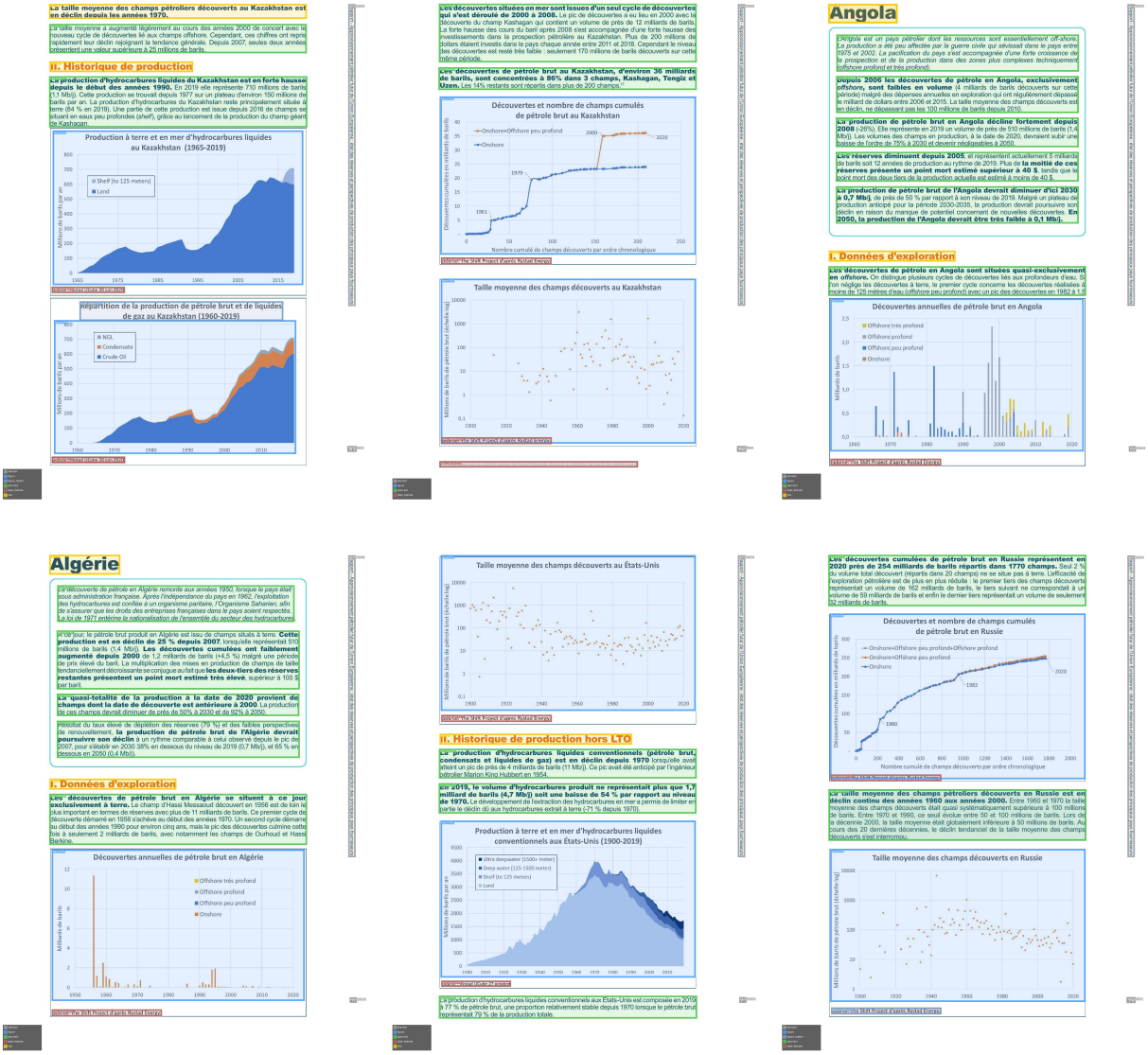

Figure 4: Qualitative examples of region detection using DocLayout-YOLO. The model successfully identifies and bounds distinct semantic regions across complex document layouts, including reports from ShiftProject and dense pages from DocVQA.

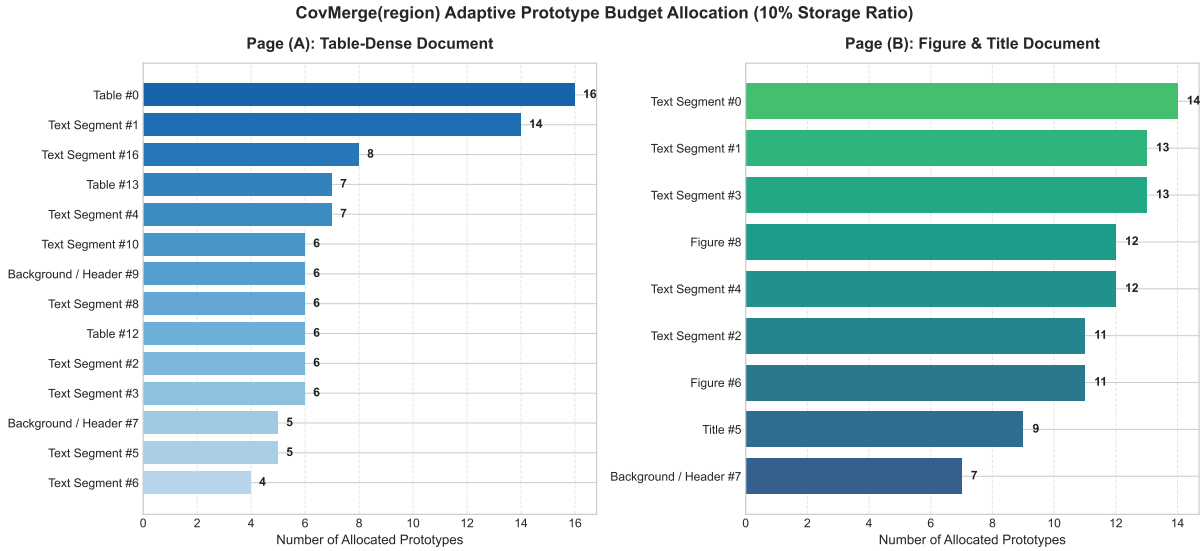

Figure 5: Intermediate visualization of the adaptive prototype budget allocation for `CovMerge(region)` under a 10% storage ratio (102 tokens per page) using the ColPali architecture. The allocation successfully prioritizes semantically rich and highly diverse regions (e.g., Tables, Figures) while suppressing uninformative backgrounds.

Table 8: Performance comparison on the ViDoRe benchmark under an aggressive **5% storage budget** (x20 compression ratio). Models are evaluated using nDCG@5 (%). The best performance among the compression methods is highlighted in **bold**, while the uncompressed dense baseline is provided as an upper bound. The subscripts in the Average column indicate the performance retention relative to the dense baseline.

Method	**ViDoRe Datasets**										
	Arxiv	**Doc**	**Info**	**TabF**	**TAT**	**Shift**	**AI**	**Energy**	**Gov**	**Health**	**Avg.**
ColPali v1.3 (Base: PaliGemma-3B)											
Dense (Upper Bound)	*82.6*	*53.8*	*84.6*	*87.4*	*69.5*	*77.0*	*96.6*	*95.0*	*95.7*	*97.9*	*84.0*$_{100\%}$
Random Pruning	77.4	41.2	77.4	82.5	47.1	58.6	86.6	83.3	83.7	89.4	72.7$_{86.5\%}$
Token Pooling	68.3	31.0	74.3	79.7	47.3	47.8	87.6	79.9	84.7	86.3	68.7$_{81.8\%}$
Light-Colpali(Train-free)	78.8	42.0	75.4	85.5	47.9	36.6	83.2	80.9	81.2	75.5	68.7$_{81.8\%}$
DocPruner(Fixed Budget)	76.6	41.0	77.9	80.1	47.8	55.6	87.3	82.7	84.6	89.3	72.3$_{86.1\%}$
SAP	77.0	46.0	75.0	82.0	51.0	57.0	90.0	85.0	86.0	90.0	74.0$_{88.1\%}$
Prune-then-Merge	81.2	49.1	83.3	**86.8**	62.3	**71.3**	**96.4**	**92.8**	91.8	**94.4**	**80.9**$_{\mathbf{96.3\%}}$
CovMerge(Global)	81.0	**51.5**	83.6	86.3	62.3	69.4	95.8	92.5	90.0	93.1	80.6$_{96.0\%}$
CovMerge(Regional)	**81.3**	50.6	**83.8**	86.1	**63.0**	68.8	94.9	91.5	**91.9**	93.4	80.5$_{95.8\%}$
ColQwen v1.0 (Base: Qwen2-VL-2B)											
Dense (Upper Bound)	*87.8*	*54.3*	*91.3*	*88.7*	*81.0*	*89.1*	*98.8*	*96.3*	*94.8*	*98.4*	*88.0*$_{100\%}$
Random Pruning	80.5	37.5	75.7	80.3	52.8	59.3	89.0	83.4	83.7	90.2	73.3$_{83.3\%}$
Token Pooling	73.1	34.4	76.6	77.7	56.0	56.6	86.9	85.8	85.0	85.9	71.8$_{81.6\%}$
Light-Colpali(Train-free)	81.5	45.5	75.5	84.2	59.4	55.1	92.3	85.5	88.0	88.0	75.5$_{85.8\%}$
DocPruner(Fixed Budget)	79.3	37.8	77.7	76.4	53.0	63.1	87.3	82.1	83.0	87.4	72.7$_{82.6\%}$
SAP	74.0	47.0	78.0	80.0	56.0	69.0	87.0	86.0	87.0	91.0	75.0$_{85.2\%}$
Prune-then-Merge	83.9	51.2	**87.1**	87.9	70.6	79.7	95.1	92.5	92.0	95.7	83.6$_{95.0\%}$
CovMerge(Global)	85.8	**52.3**	87.0	**88.5**	**74.6**	**80.9**	97.5	**94.0**	92.5	**97.4**	**85.1**$_{\mathbf{96.7\%}}$
CovMerge(Regional)	**86.0**	51.4	86.8	88.0	74.2	78.5	**97.6**	93.0	**95.0**	**97.4**	84.8$_{96.4\%}$

Table 9: Performance comparison on the ViDoRe benchmark under a mild **20% storage budget** (x5 compression ratio). Models are evaluated using nDCG@5 (%). The best performance among the compression methods is highlighted in **bold**, while the uncompressed dense baseline is provided as an upper bound. The subscripts in the Average column indicate the performance retention relative to the dense baseline.

Method	**ViDoRe Datasets**										
	Arxiv	**Doc**	**Info**	**TabF**	**TAT**	**Shift**	**AI**	**Energy**	**Gov**	**Health**	**Avg.**
ColPali v1.3 (Base: PaliGemma-3B)											
Dense (Upper Bound)	*82.6*	*53.8*	*84.6*	*87.4*	*69.5*	*77.0*	*96.6*	*95.0*	*95.7*	*97.9*	*84.0*$_{100\%}$
Random Pruning	81.2	49.7	81.5	85.3	61.7	73.9	94.3	92.8	93.2	94.5	80.8$_{96.2\%}$
Token Pooling	75.8	43.5	79.8	85.5	58.6	66.3	92.8	91.7	93.1	93.4	78.0$_{92.9\%}$
Light-Colpali(Train-free)	81.4	50.5	83.5	**88.5**	65.5	73.0	96.3	93.2	93.9	96.1	82.2$_{97.9\%}$
DocPruner(Fixed Budget)	80.4	48.5	83.2	86.6	61.8	71.0	93.9	91.6	91.2	94.2	80.2$_{95.5\%}$
SAP	**82.0**	**57.0**	83.0	86.0	67.0	74.0	95.0	93.0	**95.0**	95.0	83.0$_{98.8\%}$
Prune-then-Merge	**82.0**	52.8	**84.6**	87.6	**68.9**	**77.1**	**97.4**	93.5	94.1	96.2	83.4$_{99.3\%}$
CovMerge(Global)	**82.0**	52.0	84.5	87.7	68.3	**77.1**	**97.4**	93.8	93.0	96.3	83.2$_{99.0\%}$
CovMerge(Regional)	**82.0**	52.7	**84.6**	87.6	68.3	**77.1**	97.1	**94.2**	94.3	**97.4**	**83.5$_{99.4\%}$**
ColQwen v1.0 (Base: Qwen2-VL-2B)											
Dense (Upper Bound)	*87.8*	*54.3*	*91.3*	*88.7*	*81.0*	*89.1*	*98.8*	*96.3*	*94.8*	*98.4*	*88.0*$_{100\%}$
Random Pruning	85.6	50.4	86.3	87.0	72.2	78.8	97.4	91.9	92.6	96.7	83.9$_{95.3\%}$
Token Pooling	81.9	47.2	84.7	85.6	69.3	74.6	96.1	92.0	93.0	94.0	81.8$_{92.9\%}$
Light-Colpali(Train-free)	86.1	52.5	88.3	87.9	75.3	82.0	97.2	94.8	93.6	97.9	85.6$_{97.3\%}$
DocPruner(Fixed Budget)	85.3	51.2	85.2	87.8	72.5	83.2	96.9	94.5	**94.7**	96.5	84.8$_{96.4\%}$
SAP	82.0	**56.0**	88.0	88.0	76.0	83.0	96.0	92.0	94.0	97.0	85.0$_{96.6\%}$
Prune-then-Merge	85.7	53.3	87.7	88.5	75.4	84.8	97.3	94.0	93.4	97.5	85.8$_{97.5\%}$
CovMerge(Global)	86.8	54.3	89.9	**88.7**	**78.7**	**87.8**	**99.3**	95.0	**94.7**	**98.6**	**87.4$_{99.3\%}$**
CovMerge(Regional)	**86.9**	54.2	**90.6**	88.4	**78.7**	87.4	**99.3**	**95.2**	**94.7**	98.1	**87.4$_{99.3\%}$**

Algorithm 1 CovMerge: Hyperspherical Semantic Coverage and Token Merging

Require: Document page patch embeddings $\mathbf{X} = \{x_i\}_{i=1}^{N_p} \subset \mathbb{R}^D$; Target prototype budget M; Scaling parameters α, β (for regional mode); Layout engine (e.g., DocLayout-YOLO) yielding regions $\mathcal{R} = \{R_1, R_2, \ldots, R_K\}$ with $R_k \subset \mathbf{X}$.

Ensure: Compressed page prototypes $\mathbf{P} = \{p_j\}_{j=1}^{M} \subset \mathbb{R}^D$.

0: **Initialize** seed prototype set $\mathbf{S} \leftarrow \emptyset$
0: **if** *Mode* is `CovMerge(Global)` **then**
0: $\mathbf{S} \leftarrow$ FarthestPointSampling($\mathbf{X}$, budget $= M$) {Maximize page-wide diversity}
0: **else**{*Mode* is `CovMerge(Regional)`}
0: **for** each region $R_k \in \mathcal{R}$ **do**
0: Calculate patch count $n_k = |R_k|$
0: Calculate intra-region diversity $\text{div}_k = 1 - \frac{1}{n_k^2} \sum_{x_i, x_j \in R_k} \text{cos_sim}(x_i, x_j)$
0: Compute allocation score $\text{score}_k = n_k^{\alpha} \times \text{div}_k^{\beta}$
0: **end for**
0: Normalize scores to allocate integer budgets $\{M_1, M_2, \ldots, M_K\}$ such that $\sum_k M_k = M$
0: **for** each region $R_k \in \mathcal{R}$ **do**
0: $\mathbf{S}_k \leftarrow$ FarthestPointSampling(R_k, budget $= M_k$) {Layout-guided seeds}
0: $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{S}_k$
0: **end for**
0: **end if**
{_____ Global Spherical Refinement _____}
0: $\mathbf{P}^{(0)} \leftarrow \mathbf{S}$ {Initialize centroids with coverage-aware seeds}
0: $t \leftarrow 0$
0: **repeat**
0: **Partition step:** Assign each patch $x_i \in \mathbf{X}$ to the nearest centroid on hypersphere:
$\mathcal{C}_j^{(t)} = \{x_i \in \mathbf{X} \mid j = \arg\max_l \text{cos_sim}(x_i, p_l^{(t)})\}$
0: **Update step:** Project the mean vector of each cluster back to the unit sphere:
$p_j^{(t+1)} \leftarrow \text{Normalize}\left(\sum_{x_i \in \mathcal{C}_j^{(t)}} x_i\right) = \frac{\sum x_i}{\|\sum x_i\|_2}$
0: $t \leftarrow t + 1$
0: **until** centroids $\mathbf{P}$ converge or $t \geq t_{\max}$
0: **return** Final prototypes $\mathbf{P} = \mathbf{P}^{(t)}$ =0