GENERATIVE LATE-INTERACTION EMBEDDINGS FOR VISUAL DOCUMENT RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Late-interaction retrieval is the state-of-the-art for visual document search, but storage costly, where compression techniques retain a subset or local average of the $N \approx 1{,}000$ vectors per page. Under aggressive memory budgets, these methods degrade sharply, while alternatives require retraining the encoder. Investigating this degradation across three encoders revealed two geometric properties: vectors lie on the unit sphere and concentrate within a 5-6 dimensional manifold. Consequently, a) standard k-means centroids fall inside the sphere, underestimating MaxSim scores and normalizing them to the surface fixes this ($+0.093$ nDCG@5). b) the low-dimensional manifold allows full vector regeneration from a few. To this end, we introduce Generative Late-Interaction Embeddings (**GLIE**): $k \ll N$ vectors per page learned from the normalized centroids to serve as both a lightweight index and a basis for regenerating a page's full embedding set. At query time, search runs exclusively on these k vectors, and a decoder expands only the top candidates back to all N vectors for exact re-scoring. At four vectors per page on ViDoRe v1, **GLIE** retains nearly 80% of the uncompressed system's nDCG@5, against 70% for the best prior post-hoc method. These results use a 415K-parameter network fitted in within three GPU-minutes on just a few thousands training pages. At a matched training budget, fine-tuning the encoder does not reach even the training-free stage of **GLIE**, and the full system beats it at every budget. These patterns are true for a second encoder and ViDoRe v2. By reconstructing evidence on demand rather than sampling it, **GLIE** opens a new axis for storage-efficient retrieval, with the decoder as its main design surface.

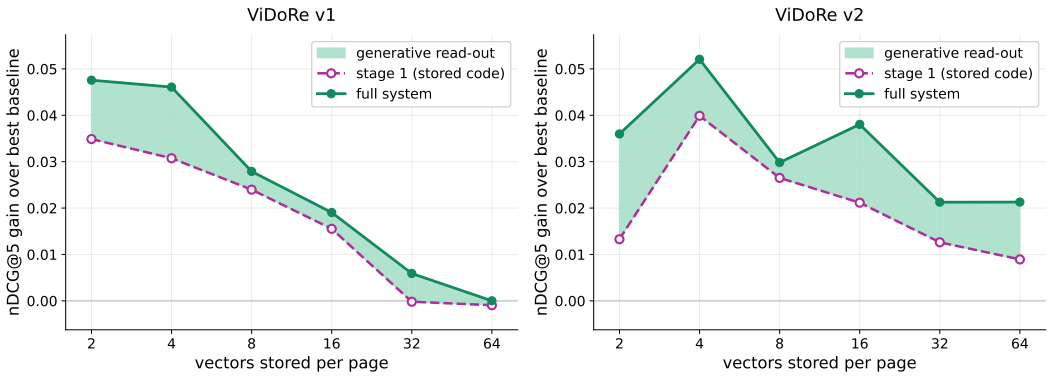
1 INTRODUCTION

Retrieval over visual documents has converged on late interaction. Instead of collapsing a page into one vector, models such as ColPali (Faysse et al., 2025) embed every image patch separately and score a query against a page with the MaxSim operator, the sum over query tokens of the maximum inner product with any page vector. This design is what makes visual retrieval work. It preserves local evidence such as a table cell or a phrase in a figure caption, and multi-vector representations are provably more expressive than any single vector of comparable size (Jayaram, 2026).

The price is storage. ColPali stores 1,031 patch vectors of dimension 128 per page, about 258 KB in bfloat16, so one million pages cost a quarter terabyte of embeddings before any index structure is added. The cost is paid at rest, in memory during search, and in transfer at query time, and the community now names index footprint among the paradigm's central open problems (Clavié et al., 2026). The dominant remedies compress the stored set directly: pooling merges similar vectors (Clavié et al., 2024), pruning keeps a salient subset (Liu et al., 2026), merging studies combine both (Ma et al., 2025; Yan et al., 2026), and quantization stores vectors as cluster IDs and quantized residuals (Santhanam et al., 2022a). These methods share a premise, that the compressed representation is a subset or local average of the encoder's vectors, and they also share a limit. None reports operating below roughly sixteen vectors per page, and methods that reach smaller budgets retrain the encoder (MacAvaney et al., 2025; Veneroso et al., 2025; Xiao et al., 2026), which invalidates every embedding already computed.

This paper starts from a different question: *what is the stored object, geometrically?* Measuring ColPali page embeddings across all ten evaluation corpora, we find that the 1,031 vectors of a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) (b)

Figure 1: Margin over the strongest training-free baseline at each budget on ViDoRe v1 and v2.

page concentrate near a manifold of intrinsic dimension five to six, living on the unit sphere of the 128-dimensional embedding space. Two further encoders give the same answer, one of them in a 3,072-dimensional space.

If a page is really a low-dimensional surface on the sphere, then a compressed representation should describe those degrees of freedom instead of subsampling the patches. We introduce **GLIE**, which builds such a code post hoc from a frozen encoder. Per-page cluster centroids are projected back onto the unit sphere, where the encoder actually places its outputs, a free correction worth up to $+0.093$ nDCG@5 on its own. A zero-initialized network then adjusts those centroids while reading the full token set, so the code starts exactly at normalized clustering and training improves it. At query time, cheap MaxSim over the projected vectors ranks every page, and then, for the top-L candidates only, a shared decoder regenerates the full set of embeddings for exact rescoring.

This design shift has three structural consequences. First, the code lies on the unit sphere by construction. Second, correctness is structural: the code cannot start worse than normalized clustering, and because the decoder emits each stored vector unchanged among its outputs, regeneration can add evidence but cannot destroy it. Third, nothing in the codec conditions on a corpus, so one fit serves every collection.

Because what **GLIE** targets is the gap between a compressed code and its own uncompressed system, a stronger encoder raises the ceiling it is closing on, and a better decoder closes more of that gap.

Contributions:

Geometry of the stored object. We measure page token clouds as a manifold of intrinsic dimension five to six on the unit sphere.

Spherical anchoring. We identify a systematic MaxSim underestimate in standard k-means clustering and remove it at zero cost by re-projecting centroids to the unit sphere.

Generative Late-Interaction Embeddings (GLIE). We replace extractive subset sampling with a codec that stores k vectors per page and regenerates all N from them on demand, together with the asymmetric pipeline it enables: initial retrieval runs on the stored vectors alone, and only a top-L shortlist is expanded and rescored exactly.

2 Related Work

Late-interaction retrieval. ColBERT introduced token-level document representations scored with MaxSim (Khattab & Zaharia, 2020), ColBERTv2 compressed the index with centroid-plus-residual storage (Santhanam et al., 2022b), and ColPali transferred the paradigm to visual documents with the ViDoRe benchmark, on which late interaction dominates single-vector alternatives (Faysse et al., 2025). Video-ColBERT extends it to video and names storage as its principal drawback (Reddy et al., 2025). Jayaram (2026) proves multi-vector embeddings strictly more expressive than single vectors of comparable dimension, which cautions against collapsing pages into one vector. The gap in this

Table 1: Positioning against prior compression for late interaction. The frozen-backbone column sets our experimental scope: a method that updates the encoder cannot be adopted without re-encoding the corpus, so Section 4 compares against the frozen rows and reproduces only Light-ColPali, at a matched training budget.

Method	Frozen backbone	Effective ≤ 16 vec	New budget (no re-encode)	Generative read-out	Adapt cost per budget
Token pooling / merging	✓	✗	✓	✗	0
Light-ColPali (finetuned)	✗	✗	✗	✗	$\sim$72 GPU-h
MetaEmbed	✗	✓	✓	✗	$\sim$192 GPU-h*
GLIE, training-free stage	✓	✓	✓	✗	0
GLIE, full	✓	✓	✓	✓	$\sim$3 GPU-min

*Amortized across five jointly trained budgets (960 GPU-h total).

line is that expressiveness is bought with storage, and no account exists of when that storage can be compressed safely.

Post-hoc reduction of the stored set. Token pooling merges document vectors by hierarchical clustering, retaining about 97% of quality at pool factor 4 (Clavié et al., 2024). For visual documents, Light-ColPali finds that merging post-projector embeddings retains 97.8% of nDCG@5 at merging factor 9 and 93.6% at factor 49, roughly a $41\times$ smaller index, its most aggressive published point being about twenty-one vectors per page (Ma et al., 2025). Anchor pruning (Liu et al., 2026) and prune-then-merge (Yan et al., 2026) extend the near-lossless range. All store a subset or local average of the encoder's vectors, and none reports an operating point below roughly sixteen vectors per page, despite selecting, averaging, and pruning by different criteria. Section 3.1 characterizes the geometry behind that shared limit, and our method targets the budgets below it.

Small budgets via retraining. ConstBERT learns a projection to a constant number of document vectors (MacAvaney et al., 2025), CRISP trains clusterability into the representation (Veneroso et al., 2025), and MetaEmbed trains nested meta-tokens serving budgets 1 to 64 at test time (Xiao et al., 2026). Each trains the encoder, so adopting one means giving up the public checkpoint and re-encoding the corpus. The training is substantial on its own: MetaEmbed reports 32 H100 GPUs for 30 hours. Budgets are also fixed at training time, and within that set a deployed index can be truncated freely, but reaching a budget outside it requires training again. Our setting is complementary: a frozen public checkpoint, cached embeddings, and a per-budget fit measured in GPU-minutes.

Orthogonal axes. PLAID (Santhanam et al., 2022a) and EMVB (Nardini et al., 2024) compress each vector and prune candidates at query time while leaving the per-page vector count intact, Matryoshka learning shrinks the dimension axis (Kusupati et al., 2022; Xiang et al., 2026), and MUVERA sketches multi-vector scoring for candidate generation while keeping full vectors at rest (Dhulipala et al., 2024). These compose with vector-count compression.

Training-free reduction is cheap and elastic but stops at a shared floor. Retraining crosses the floor but gives up the frozen checkpoint. The orthogonal axes shrink bits, dimensions, or scoring cost. What none of the three asks is what the stored set actually is: a low-dimensional manifold on the sphere. Section 3 measures that geometry and builds the code it implies. Table 1 summarizes.

3 Generative Late-Interaction Embeddings

3.1 What the stored object is

Let $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ be the token vectors a frozen encoder produces for a page, with $N = 1{,}031$ and $D = 128$ for ColPali, and $Q = \{q_1, \ldots, q_m\}$ the vectors for a query. The late-interaction score is

$$\mathrm{MaxSim}(Q, X) \;=\; \sum_{j=1}^{m} \max_{i} \langle q_j, x_i \rangle. \tag{1}$$

Table 2: Per-page geometry (medians) for three encoders.

Statistic	ColPali	ColQwen2	Nemotron v2
Ambient dimension D	128	128	3,072
Token norm	1.000	1.000	1.000
TwoNN intrinsic dimension	4.9	5.1	6.1
per-corpus range	4.7–5.1	4.9–5.5	5.6–8.1

A compressed representation at budget k replaces X with a set C of k vectors, scored by the same operator. Two measured properties of X constrain what C should be. Both are computed on the corpora and encoders of Section 4.1, over 6,729 pages.

1) Low intrinsic dimension. The TwoNN estimator (Facco et al., 2017) gives a median of 4.9 against an ambient dimension of 128 on ColPali. Per-corpus medians span 4.7 to 5.1, from scientific figures to government reports. A second encoder gives 5.1, and a third, Nemotron v2 (Moreira et al., 2026), gives 6.1 at an ambient dimension of 3,072 (Table 2). Ambient dimension varies by 24× across the three and intrinsic dimension varies by one. The number is a property of the token cloud and not of the estimator. On the same ColPali pages, a Gaussian fitted to each page's own covariance, which has the same linear spectrum and no curvature, reads 32.2, and uniform noise of the same size reads 61.4.

2) Unit norm. All three encoders L2-normalize their output, so X lies exactly on the sphere $\mathbb{S}^{D-1}$.

A code that respects both properties is small and spherical. Clustering respects the first and not the second. A k-means centroid is a Euclidean mean of unit vectors, meaning it lies strictly inside the sphere. Because its norm is less than one, it systematically understates the inner products in equation 1 for all query directions.

Proposition 1 (The centroid norm records its cluster's spread)**.** *Let $x_1, \ldots, x_n$ be unit vectors with mean c. Then*

$$\frac{1}{n}\sum_{i=1}^{n}\|x_i - c\|^2 \;=\; 1 - \|c\|^2,$$

and consequently the k-means objective on the sphere equals $\sum_j n_j\,(1 - \|c_j\|^2)$ over clusters j with sizes n_j.

Proof. Expand $\|x_i - c\|^2 = 1 - 2\langle x_i, c\rangle + \|c\|^2$ and average, using $\frac{1}{n}\sum_i \langle x_i, c\rangle = \|c\|^2$. □

Three consequences follow. Firstly, projecting the centroids back onto the unit sphere costs nothing and should improve MaxSim by correcting this systematic underestimation. Secondly, the improvement should shrink as k grows, because tighter clusters have means closer to the sphere. Thirdly, the projection moves the code outside the original data, so a centroid's score is no longer capped by the best true vector. A code that could only underestimate can now overestimate as well.

3.2 What a learned code must add

Normalized k-means is the training-free stage of **GLIE**, and it falls short in two ways that define the rest of the codec. First, it is blind beyond the mean. Each centroid summarizes its cluster by an average, so scoring-relevant structure inside the cluster is unrecoverable from the code no matter how the centroids are post-processed. Second, it is a sample. It stores k points and can only ever score with those k points, so a query token pointing at a region the sample misses loses its evidence. A code trained as a *description* of the page can instead be expanded back into the fine token cloud when a candidate matters, re-materializing evidence that no k-vector sample can hold. This is why regeneration is useful at all.

GLIE discharges these requirements with three components, each anchored to normalized k-means: (a) the code is initialized at it exactly, (b) is refined against the MaxSim operator itself while reading the full token set, and (c) is decoded under a guarantee that regeneration never lowers a stored score. The encoder is frozen throughout and everything happens post hoc on cached embeddings. Figure 2 shows the fitting procedure and Figure 3 the query path.

3.3 SPHERICAL ANCHORING

Per page, k-means centroids of the token set X are computed and projected onto the unit sphere, $u_j = c_j/\|c_j\|$. By Proposition 1 the projection removes a systematic MaxSim underestimate at zero cost. Empirically it is worth $+0.031$ to $+0.093$ nDCG@5 on the full benchmark, shrinking as k grows exactly as the proposition predicts (Table 7). Normalized clustering is therefore the training-free stage of **GLIE** and the floor every learned component must clear, as well as the practical one-line recommendation for any dot-product late-interaction system that clusters: normalize your centroids.

3.4 ZERO-INITIALIZED REFINEMENT THAT READS THE PAGE

Defining $U = \{u_1, \ldots, u_k\}$ for the anchors, a shared cross-attention module refines them against the full token set, with U as queries and X as keys and values,

$$C \;=\; \mathrm{normalize}\big(U + \pi_\theta(\mathrm{Attn}(U,\, X,\, X))\big), \tag{2}$$

where $\mathrm{Attn}(Q, K, V)$ is multi-head attention, $\mathrm{normalize}$ projects each row back to the unit sphere as in Section 3.1, and the output projection π_θ is initialized at zero, so at initialization $\pi_\theta(\cdot) = 0$ and $C = U$ exactly. Intuitively, the code begins exactly at normalized clustering and training can only move it where the objective improves, so the strongest training-free solution is the floor rather than a competitor. The correction reads X directly, so the code can encode scoring-relevant structure that centroids average away, which no post-processing of the centroids alone could recover. The refiner and decoder together hold 415K parameters against the 3B backbone. The refiner runs once per page at indexing time, while the decoder runs only at query time, on the shortlist.

3.5 ANCHORED GENERATIVE READ-OUT

A shared decoder g_ψ expands the k stored vectors back into N unit vectors for reranking. Its structure encodes three guarantees. First, count-proportional slots: cluster j owns n_j output slots, so the decoder reproduces the page's actual cluster structure rather than a fixed learned layout. Second, an exact anchor: slot 0 of each cluster emits its refined unit vector C_j verbatim, $\hat{x}_{j0} = C_j$, so the decoded set contains the code set, and because MaxSim is a maximum,

$$\mathrm{MaxSim}(Q, g_\psi(C)) \;\geq\; \mathrm{MaxSim}(Q, C) \quad \text{per query token.}$$

Intuitively, regeneration can add evidence but structurally cannot destroy it, with no tuning. Third, bounded displacement. A shared MLP predicts a displacement $d_{j\ell}$ and a scalar $a_{j\ell}$ from C_j, the original centroid norm $\|c_j\|$, the cluster count n_j, and fixed features identifying slot ℓ. Each non-anchor child, $\ell = 1, \ldots, n_j - 1$, is

$$\hat{x}_{j\ell} = \mathrm{normalize}(C_j + \alpha\, \sigma(a_{j\ell})\, \mathrm{normalize}[d_{j\ell} - \langle C_j, d_{j\ell}\rangle C_j]\,), \tag{3}$$

where σ is the sigmoid function and $\alpha = 0.75$. The inner term removes the radial component and normalizes the tangent direction; the outer normalization returns the child to the sphere. A child can therefore sit at most $\arctan(0.75) \approx 37^\circ$ from its anchor, so a cluster's children fill a patch around it rather than scattering across the sphere. Fixed sine and cosine slot features require no per-slot parameters, allowing one decoder to serve clusters of any size. Appendix A.1 specifies the architecture.

Prior compression selects or averages among the encoder's vectors at indexing time, and that fixed set is all a query can ever be scored against. The generative read-out instead re-materializes the fine structure only when a candidate is worth the cost.

3.6 TWO-STAGE INFERENCE

GLIE (Figure 3) combines the code and the read-out:

1. **Stage 1 (retrieve):** score every page against the query by MaxSim over its k stored vectors, at the cost of a pooled baseline.
2. **Stage 2 (regenerate and rerank):** expand the top-L candidates ($L = 20$) back to N vectors with g_ψ and rescore them by full MaxSim. Non-candidates keep their first-stage order.

The asymmetry is the point: cheap over everything, expensive over almost nothing.

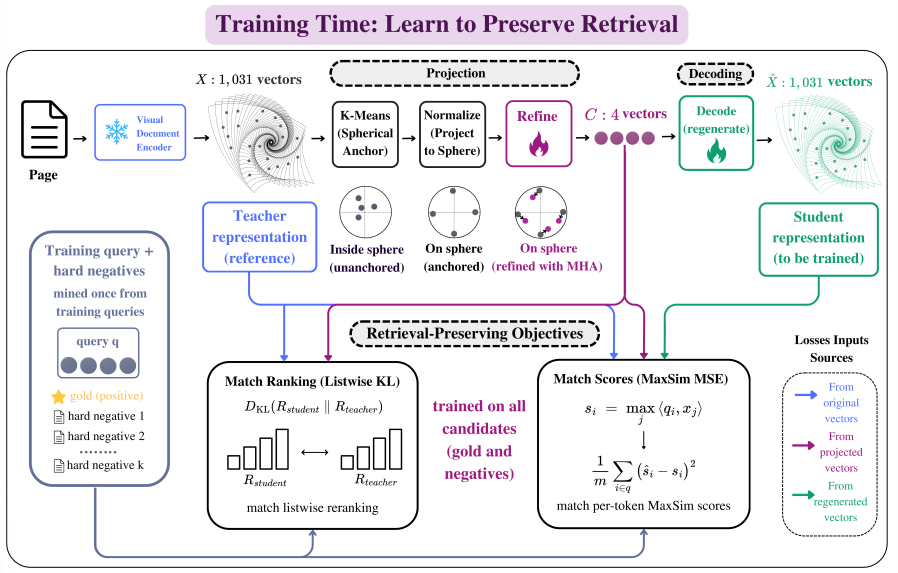


Figure 2: **Training.** The encoder stays frozen. The projection is initialized at normalized k-means and a refiner adjusts it from there. Three losses train the two objects: a MaxSim term and a listwise term on the stored code, and a MaxSim term on the regenerated vectors.

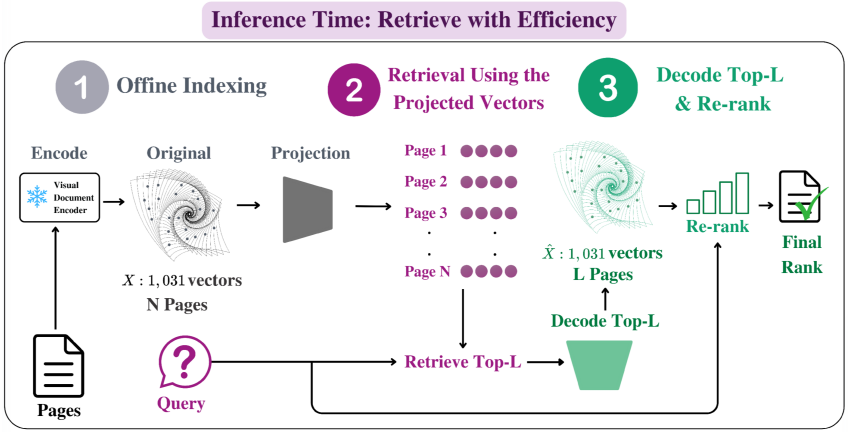


Figure 3: **Inference.** Every page is scored on its k projected vectors, then only the top-L shortlist is decoded back to full length and rescored exactly.

3.7 Training

Two objects are trained, the code and the regenerated set, and both must score a query the way the full page does (i.e., for every query token, the best match in the trained object should equal the best match in the page). Therefore, we use the squared difference of the two per-token MaxSim values,

$$\mathcal{L}_{\text{tok}} = \frac{1}{m}\sum_{j=1}^{m}\Big(\max_{l}\langle q_j, \hat{x}_l\rangle - \max_{i}\langle q_j, x_i\rangle\Big)^2. \tag{4}$$

Here x_i ranges over the page's N encoder vectors and $\hat{x}_l$ over the object being trained, the k code vectors or the N regenerated vectors. This term is applied to both objects. The code receives one more term. Because it ranks every page in the corpus, its order of candidates should match the teacher's, so we add a listwise KL over each query's candidate list $\mathcal{P}$ (the relevant page and its mined hard negatives),

$$\mathcal{L}_{\text{list}} = \text{KL}\Big(\text{softmax}_{c\in\mathcal{P}}\big(s_c/\tau\big)\ \Big\|\ \text{softmax}_{c\in\mathcal{P}}\big(\hat{s}_c/\tau\big)\Big), \tag{5}$$

Table 3: ViDoRe v1 (10 subsets), nDCG@5 on all queries of each subset, macro-averaged across subsets.

Method	k=2	k=4	k=8	k=16	k=32	k=64
Raw k-means	0.464	0.512	0.594	0.662	0.737	0.779
Token pooling	0.553	0.584	0.624	0.657	0.702	0.747
Cluster merging	0.447	0.471	0.531	0.613	0.690	0.763
GLIE	**0.610**	**0.655**	**0.711**	**0.755**	**0.790**	**0.809**
shortlist oracle	0.769	0.782	0.807	0.821	0.830	0.834
% of uncompressed (0.836)	73%	78%	85%	90%	94%	97%

Table 4: ViDoRe v2 (4 subsets), nDCG@5 on all queries of each subset, macro-averaged across subsets.

Method	k=2	k=4	k=8	k=16	k=32	k=64
Raw k-means	0.184	0.201	0.275	0.318	0.376	0.405
Token pooling	0.228	0.241	0.244	0.276	0.307	0.370
Cluster merging	0.188	0.181	0.190	0.220	0.285	0.363
GLIE	**0.274**	**0.327**	**0.388**	**0.435**	**0.467**	**0.495**
shortlist oracle	0.416	0.454	0.483	0.506	0.513	0.521
% of uncompressed (0.517)	53%	63%	75%	84%	90%	96%

where s_c and $\hat{s}_c$ are the MaxSim scores of candidate c under the encoder and under the code, and τ is a temperature. That is the whole objective: three terms in total.

Reconstruction error is absent on purpose. It is minimized by placing every child near its cluster mean, which is exactly the collapse that loses the extreme points MaxSim reads.

4 Experiments

4.1 Setup

Protocol. We evaluate on all ten ViDoRe v1 subsets (Faysse et al., 2025) under the benchmark's standard protocol. The codec is fitted once on 5,000 pages of the public ColPali training collection, and applied frozen to every test subset. The corpus of each subset is its set of unique pages, de-duplicated by image identity, with query-less pages kept as distractors. We report single-relevant nDCG@5, and every learned number is a mean over three training seeds. Frozen ColPali v1.3 is used, which gives 1,031 patch vectors of dimension 128 per page. Budgets $k \in \{2, 4, 8, 16, 32, 64\}$, i.e. $516\times$ down to $16\times$ fewer stored vectors. At $k = 4$ a page costs 1,040 bytes against 257.8 KB uncompressed, so one million pages shrink from 258 GB to 1.0 GB. Section 4.2 repeats the sweep on ViDoRe v2 (Macé et al., 2025) and Section 4.4 on ColQwen2.

Baselines. All baselines are post hoc on the same frozen encoder at identical stored budget: raw per-page k-means, semantic cluster merging (the training-free stage of Light-ColPali (Ma et al., 2025), with its fine-tuning removed), and sequential token pooling (Clavié et al., 2024).Margins are always reported against the strongest training-free method chosen per subset and per budget, normalized k-means included. We additionally reproduce Light-ColPali's fine-tuning stage at matched training budget and source (Section 4.3). Architecture and evaluation details are in Appendices A.1 and A.2.

4.2 Main Results: ViDoRe v1 and v2

Aggressive-budget results. GLIE reaches 78% of uncompressed retrieval quality storing 1.0 KB per page and 90% at 4 KB, and beats almost every prior baseline on all subsets of both ViDoRe v1 and v2 at every budget (Table 3, 4). The best-powered subset is also among the strongest. TAT-DQA, with 1,663 queries, shows margins $+0.035$ to $+0.053$ at $k \leq 8$ over its own training-free stage.

Margin profile across budgets. The margin is positive at every budget but not uniform, and its profile differs between the two benchmarks (Figure 1). On v1 it is $+0.048$ and $+0.046$ at $k = 2$

Table 5: Matched training budget over the ten ViDoRe v1 subsets.

Method	k=2	k=4	k=8	k=16	k=32	k=64
Light-ColPali (LoRA fine-tune, ours)	0.523	0.544	0.586	0.632	0.661	0.701
Normalized k-means (free)	0.552	0.605	0.684	0.736	0.784	0.809
GLIE (frozen backbone)	**0.610**	**0.655**	**0.711**	**0.755**	**0.790**	**0.809**

Table 6: Full sweep repeated on ColQwen2 over the ten ViDoRe v1 subsets.

Method	k=2	k=4	k=8	k=16	k=32	k=64
Raw k-means	0.524	0.623	0.706	0.762	0.806	0.841
Token pooling	0.619	0.657	0.702	0.740	0.774	0.812
Cluster merging	0.529	0.520	0.553	0.643	0.743	0.809
GLIE	**0.677**	**0.733**	**0.781**	**0.824**	**0.849**	**0.868**
% of uncompressed (0.883)	77%	83%	88%	93%	96%	98%
Subsets improved	10/10	10/10	10/10	10/10	10/10	10/10

and 4, drops to $+0.028$ at $k = 8$, decreases to +0.019 at $k = 16$, and decays to noise by $k = 32$, a mean of $+0.041$ at $k \leq 8$ against $+0.008$ above. The stored code alone turns slightly negative at $k = 64$, where normalized clustering already sits 0.027 from the ceiling and a refiner fitted elsewhere has nothing left to correct. Four of the ten v1 subsets have ceilings of 0.94 to 0.98, so part of that high-budget decay is saturation rather than method failure. On v2, which saturates nowhere, the margin never decays and the band the read-out contributes stays open to $k = 64$. Finally, at $k = 2$, with fewer anchors than the page's intrinsic dimension, the generative read-out still adds $+0.012$ on v1 and $+0.022$ on v2.

4.3 A Small Codec Beats Fine-Tuning at a Matched Budget

Fine-tuning the encoder on 4,000 pages does not reach free normalized clustering at any budget in our hands (Table 5), while the same source and budget spent on a frozen-backbone codec beats it at all six, by $+0.087$ to $+0.129$. The gap is not evidence against Light-ColPali, whose authors train on 130K queries for about 72 GPU-hours per budget. Instead, it measures what a small training budget buys: 13.3M LoRA parameters over 1.5 GPU-hours perturb a 3B backbone too little to help and enough to hurt, whereas 415K parameters over under three GPU-minutes reshape the stored code directly. Appendix A.3 gives the adapter, merge, and training settings of the reproduction.

4.4 The Recipe Transfers to a Second Encoder

On ColQwen2 the codec retains 83% of uncompressed quality at $k = 4$ (Table 6). Against spherical anchoring alone, which already reaches 0.702 at $k = 4$ on this encoder, it wins at every budget, by $+0.048$ at $k = 2$ and $+0.031$ at $k = 4$. ColQwen2 is the stronger encoder, so clustering already sits closer to its ceiling at every budget and the room a learned code has is correspondingly smaller, with margins of $+0.040$ at $k \leq 4$ against $+0.009$ at $k \geq 16$.

4.5 What limits the cascade

Component decomposition. Table 7 assigns the gains cleanly. Spherical anchoring is the largest single factor at every budget, from $+0.093$ at $k = 4$ to $+0.030$ at $k = 64$, and it is free. The learned code adds $+0.046$ to $+0.016$ up to $k = 16$ and nothing beyond it. The generative read-out is the aggressive-budget specialist on v1, $+0.012$ and $+0.015$ at $k = 2$ and 4 against at most $+0.006$ beyond. On ViDoRe v2 it does not decay, because that benchmark saturates nowhere and the headroom survives to $k = 64$ (Figure 1).

Error analysis. The oracle row splits the residual error (Figure 4). At $k = 4$ on v1 the gap from **GLIE** to the oracle, 0.655 to 0.782, is decode fidelity, recoverable by better decoding, while the gap from the oracle to the ceiling, 0.782 to 0.836, is shortlist recall, recoverable only with a larger

Table 7: What each component contributes, ViDoRe v1, three-seed means. Each row adds one component to the row above, so the three increments sum to the full-system gain over raw k-means.

Component	k=2	k=4	k=8	k=16	k=32	k=64
Raw k-means	0.464	0.512	0.594	0.662	0.737	0.779
+ spherical anchoring *(training-free)*	0.552	0.605	0.684	0.736	0.784	0.809
+ learned code (stage 1)	0.598	0.640	0.708	0.752	0.784	0.808
+ generative read-out (stage 2)	**0.610**	**0.655**	**0.711**	**0.755**	**0.790**	**0.809**

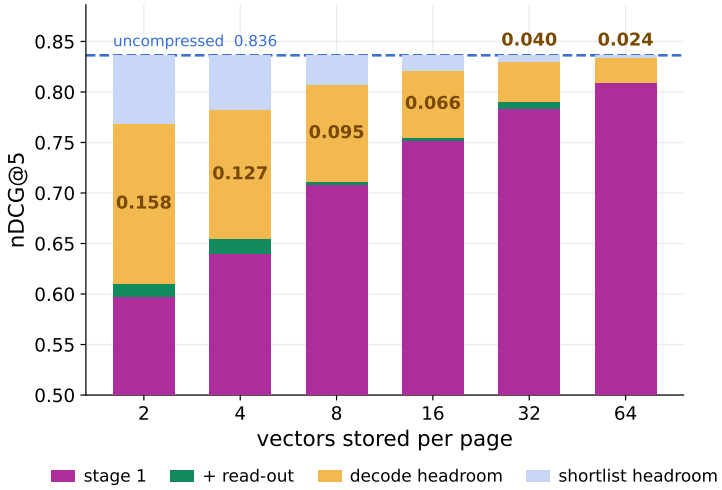


Figure 4: Decomposition of the gap to the uncompressed ceiling at each budget on ViDoRe v1.

L. Widening the shortlist confirms the split (Appendix A.4): from $L = 5$ to $L = 100$ the oracle rises from 0.704 to 0.824 while **GLIE** moves only from 0.650 to 0.663, so the shortlist is not the binding constraint and $L = 20$ is already past the point of diminishing returns. The generative read-out contributes $+0.015$ at $k = 4$, a third of the total margin at near-zero marginal cost, and a further $+0.13$ sits in the same shortlist waiting on a better decoder. On v2 that decode-fidelity term stays large at every budget, which is why the read-out keeps earning there. Decoding is therefore the lever with measured headroom. Simply scaling capacity does not reach it (Appendix A.4), but swapping decoder architecture already moves the score a lot (Appendix A.5), making decoder design a promising direction for closing the remaining gap.

Hyperparameters. Decoder capacity from 184K to 13M parameters moves nDCG@5 by at most 0.008 with no monotone trend, and widening the shortlist beyond $L = 20$ is worth $+0.003$ while the oracle over the same shortlists climbs by $+0.04$ (Appendix A.4). Training data saturates just as early: refitting on 1,250, 2,500, and 5,000 source pages leaves the $k = 4$ margin at $+0.048$, $+0.052$, and $+0.049$, so about a thousand pages, a hundredth of what the encoder trains on, is enough. What the codec learns is a property of the encoder's geometry, which Table 2 showed is corpus-independent.

5 Conclusion

Late-interaction retrieval owes its accuracy to storing token-level vectors and its cost to the same decision. We argued the way out is geometric. The low-dimensional geometry motivates a compact representation with a few anchors and one shared decoder. Better decoders are the immediate move on this axis, and the code itself is bounded by what k vectors can hold.

Three next steps look most promising. Extending the sweep to further late-interaction encoders would test how far the recipe carries beyond the two we evaluate. Composing **GLIE** with quantized storage, since the two act on different axes of the footprint. And the largest gains should wait where token sets are largest and most redundant, which points to video late interaction (Reddy et al., 2025).

AI USE STATEMENT

We used generative AI tools to find and screen related work, to discuss experimental design and analyses during the research, to write and debug experiment code, and to draft and polish sections of the text. All AI-assisted code was reviewed by the authors and verified against independent runs, all numbers were produced by our own executions on our own hardware, all citations were manually verified against their sources, and every claim and interpretation was decided by the authors. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

All experiments except the Light-ColPali reproduction use frozen, publicly released checkpoints (ColPali v1.3 and ColQwen2 from HuggingFace) and the public ViDoRe v1 and v2 benchmarks and ColPali training collection, on a single A100 80GB. The evaluation protocol, including corpus construction and the all-queries setting, is given in Section 4.1 and Appendix A.2; architecture, optimization, loss weights, and model selection in Appendix A.1; the Light-ColPali reproduction in Appendix A.3; and every cell behind the macro averages in Appendix A.6. Learned numbers are means over training seeds 0, 1, and 2. The complete code, including the geometry measurements, the codec, and all baselines, is included as supplementary material and will be publicly released after publication.

REFERENCES

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

Benjamin Clavié, Xianming Li, Antoine Chaffin, Omar Khattab, Tom Aarsen, Manuel Faysse, and Jing Li. Lir: The first workshop on late interaction and multi vector retrieval@ ecir 2026. In *European Conference on Information Retrieval*, pp. 158–168. Springer, 2026.

Laxman Dhulipala, Majid Hadian, Rajesh Jayaram, Jason Lee, and Vahab Mirrokni. Muvera: Multi-vector retrieval via fixed dimensional encoding. *Advances in Neural Information Processing Systems*, 37:101042–101073, 2024.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):12140, 2017.

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

Rajesh Jayaram. Multi-vector embeddings are provably more expressive than single vector embeddings. *arXiv preprint arXiv:2606.23475*, 2026.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 39–48, 2020.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

Zhuchenyang Liu, Ziyu Hu, Yao Zhang, and Yu Xiao. Structural anchor pruning: Training-free multi-vector compression for visual document retrieval. arXiv:2601.20107, 2026.

Yubo Ma, Jinsong Li, Yuhang Zang, Xiaobao Wu, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Jiaqi Wang, Yixin Cao, and Aixin Sun. Towards storage-efficient visual document retrieval: An empirical study on reducing patch-level embeddings. In *Findings of the Association for*

Computational Linguistics: ACL 2025, pp. 19568–19580, Vienna, Austria, 2025. Association for Computational Linguistics. arXiv:2506.04997.

Sean MacAvaney, Antonio Mallia, and Nicola Tonellotto. Efficient constant-space multi-vector retrieval. In *European Conference on Information Retrieval*, pp. 237–245. Springer, 2025.

Quentin Macé, António Loison, and Manuel Faysse. Vidore benchmark v2: Raising the bar for visual retrieval. *arXiv preprint arXiv:2505.17166*, 2025.

Gabriel de Souza P Moreira, Ronay Ak, Mengyao Xu, Oliver Holworthy, Benedikt Schifferer, Zhiding Yu, Yauhen Babakhin, Radek Osmulski, Jiarui Cai, Ryan Chesler, et al. Nemotron colembed v2: Top-performing late interaction embedding models for visual document retrieval. *arXiv preprint arXiv:2602.03992*, 2026.

Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. Efficient multi-vector dense retrieval with bit vectors. In *European Conference on Information Retrieval*, pp. 3–17. Springer, 2024.

Arun Reddy, Alexander Martin, Eugene Yang, Andrew Yates, Kate Sanders, Kenton Murray, Reno Kriz, Celso M De Melo, Benjamin Van Durme, and Rama Chellappa. Video-colbert: Contextualized late interaction for text-to-video retrieval. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19691–19701. IEEE, 2025.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. PLAID: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1747–1756, 2022a. arXiv:2205.09707.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, 2022b.

João Veneroso, Rajesh Jayaram, Jinmeng Rao, Gustavo Hernández Ábrego, Majid Hadian, and Daniel Cer. Crisp: Clustering multi-vector representations for denoising and pruning. *arXiv preprint arXiv:2505.11471*, 2025.

Haowen Xiang, Yibo Yan, Jiahao Huo, Yu Huang, Yi Cao, Mingdong Ou, and Xuming Hu. Mmmatryoshka: Towards budget-elastic visual document retrieval via a 2d multimodal matryoshka training framework. *arXiv preprint arXiv:2606.07654*, 2026.

Zilin Xiao, Qi Ma, Mengting Gu, Chun-cheng Chen, Xintao Chen, Vicente Ordonez, and Vijai Mohan. Metaembed: Scaling multimodal retrieval at test-time with flexible late interaction. In *International Conference on Learning Representations*, volume 2026, pp. 23619–23638, 2026.

Yibo Yan, Mingdong Ou, Yi Cao, Xin Zou, Jiahao Huo, Shuliang Liu, James Kwok, and Xuming Hu. Sculpting the vector space: Towards efficient multi-vector visual document retrieval via prune-then-merge framework. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24883–24925, 2026.

A Appendix

A.1 Implementation details

Architecture. The refiner is one four-head cross-attention layer with a zero-initialized output projection. The decoder sums 256-dimensional affine projections of C_j, $[\|c_j\|, \log(1+n_j)/\log(1+N)]$, and nine slot features: s, $\sin(\pi f s)$, and $\cos(\pi f s)$ for $f \in \{1, 2, 4, 8\}$, where $s = \ell/\max(n_j - 1, 1)$. The sum passes through LayerNorm and a $256 \to 512 \to 256$ MLP with GELU after each linear layer, followed by separate affine heads of sizes D and 1 for $d_{j\ell}$ and $a_{j\ell}$. The refiner and decoder together hold 415K parameters. Per-page k-means uses two restarts.

Optimization. We train with AdamW at learning rate 2×10^{-4} and weight decay 10^{-4}, cosine decay to 5% of the initial rate over 100 epochs, and gradient clipping at 1.0. A batch is eight queries, each paired with its relevant page and seven hard negatives drawn from the fifty highest-scoring non-relevant pages under the frozen code. Loss weights are 1.0 for the MaxSim term on the regenerated set and 0.5 for the MaxSim and listwise terms on the code, with a listwise temperature of 0.07.

Model selection and inference. Two checkpoints are kept from each run, the best single-stage code and the best cascade on that holdout, and each is reported on the metric it is deployed under. The shortlist is $L = 20$ and pages outside it keep their first-stage order. Stored vectors are bfloat16 with one fp16 norm and one uint16 count per cluster. One budget fits in 2.7 minutes on a single NVIDIA A100 80GB, and seeds are 0, 1, and 2.

A.2 Evaluation protocol details

No learned component ever sees a ViDoRe test query or page, so we evaluate on all queries of each subset: 3,943 queries in total, of which 1,663 are on TAT-DQA. The corpus of each subset is its set of unique pages, de-duplicated by image identity, with query-less pages kept as distractors. Our pipeline reproduces the official leaderboard average for ColPali v1.3. Margins are reported against the strongest training-free baseline chosen per subset and per budget. Learned rows are means over three training seeds with seed standard error at most 0.003 on every cell and margin.

A.3 Light-ColPali reproduction

Light-ColPali (Ma et al., 2025) merges post-projector embeddings by clustering and fine-tunes the encoder through the merge. We reproduce the fine-tuning half on the same source as **GLIE**. The merge follows the authors: per-page cluster assignment is computed without gradient, and the cluster means are renormalized and trained end to end. Because `vidore/colpali-v1.3` is itself a PEFT checkpoint, we merge and unload its own adapter before attaching a fresh LoRA (rank 32, $\alpha = 32$, dropout 0.05). We verify at initialization that the untrained merge scores exactly as stock ColPali plus merging before any update. Training uses the pairwise loss of the ColPali training recipe at batch 8 with gradient accumulation 4, five epochs over 4,000 query-page pairs and 10% warmup. One adapter is trained per budget, 13.3M trainable parameters and about 1.5 GPU-hours each on one A100, and evaluated zero-shot on all queries of the ten ViDoRe v1 subsets under the same protocol as **GLIE**. The published method trains on 130K queries for about 72 GPU-hours per budget and reports results only at $k \gtrsim 16$, so Table 5 is a statement about what a matched small budget buys and not about the published method.

A.4 Hyperparameter sensitivity

Hyperparameters are not where the result comes from. Varying decoder width from 128 to 1024 and depth from one block to three, a seventy-fold range from 184K to 13M parameters, moves nDCG@5 by at most 0.008 with no monotone trend (Table 8). The 415K configuration we report is an arbitrary point inside a flat band rather than a tuned one. The shortlist behaves differently and is informative for a different reason (Table 9). Widening L from 5 to 20 is worth $+0.010$, while widening it from 20 to 100 is worth $+0.003$ even though the oracle over those same shortlists climbs from 0.704 to 0.824. The candidates are present and the decoder does not exploit them.

Table 8: Decoder capacity.

width × depth	params	$k = 4$	$k = 16$
128×1	184K	0.653	0.752
128×3	316K	0.660	0.756
256×1	415K	0.659	0.753
512×1	1.3M	0.655	0.753
512×3	3.4M	0.659	**0.759**
1024×1	4.6M	0.659	0.758
1024×3	13.0M	**0.661**	0.754

Table 9: Shortlist size at $k = 4$

L	**GLIE**	oracle
5	0.650	0.704
10	0.658	0.750
20	0.660	0.783
50	**0.663**	0.813
100	**0.663**	**0.824**

A.5 Decoder architecture comparison

Table 10 compares the retrieval performance of our decoder with five other decoder architectures, described below.

Table 10: Decoder architecture comparison.

Decoder	$k = 2$	$k = 4$	$k = 8$	$k = 16$
Ours	**0.630**	**0.677**	**0.721**	**0.760**
Cross-Attention + MLP	0.621	0.662	0.716	0.752
Dot-Product	0.599	0.668	0.713	0.750
Cross-Attention + Local Mixing	0.616	0.649	0.699	0.757
Cross-Attention	0.491	0.525	0.519	0.518
MLP	0.523	0.528	0.515	0.523

Cross-Attention + MLP. Uses single-head cosine cross-attention with learned query and key projections to combine code vectors, followed by an MLP-predicted residual.

Dot-Product. Forms mixtures of code vectors using direct normalized dot-product attention and adds a combination with fixed weights of code and learned positional vectors.

Cross-Attention + Local Mixing. Combines cross-attention over code vectors with local multi-head self-attention across output slots and an MLP-predicted residual.

Cross-Attention. Generates output vectors through a single standard multi-head cross-attention layer, using learned slot queries and code vectors as keys and values.

MLP. Generates each output vector using a shared two-layer MLP conditioned on the flattened code vectors and a learned slot embedding.

The results split cleanly by whether the decoder stays anchored to the stored code vectors. The two unanchored designs (Cross-Attention and MLP) collapse to about 0.52 regardless of budget, showing that regeneration without structural grounding in the code fails entirely. Among the four anchored designs, architecture choices already move nDCG@5 by up to 0.010 at $k = 4$, yet all four remain roughly 0.1 below the oracle over the same shortlist. Since scaling capacity from 184K to 13M parameters yields at most 0.008 (Table 8), the remaining headroom is an architecture problem rather than a capacity one, making decoder design a promising direction for future work.

A.6 Per-subset results

Tables 11, 12, and 13 give every cell behind the macro averages of Tables 3 and 4.

Table 11: ViDoRe v1 per subset, part 1 of 2. nDCG@5 over all queries, three-seed means for the learned rows. Best method per column in bold; the uncompressed row is the ceiling and does not compete.

Method	k=2	k=4	k=8	k=16	k=32	k=64
ArxivQA						
Raw k-means	0.451	0.561	0.669	0.710	0.760	0.794
Token pooling	0.527	0.572	0.641	0.671	0.698	0.731
Cluster merging	0.495	0.537	0.646	0.723	0.779	0.800
GLIE, training-free stage	0.584	0.666	0.722	0.742	0.786	0.806
GLIE, stage 1	0.614	0.673	0.727	0.754	0.789	0.806
GLIE, full	**0.619**	**0.699**	**0.745**	**0.767**	**0.798**	**0.809**
Uncompressed	0.827	0.827	0.827	0.827	0.827	0.827
DocVQA						
Raw k-means	0.118	0.183	0.255	0.335	0.400	0.460
Token pooling	0.191	0.220	0.250	0.307	0.343	0.387
Cluster merging	0.140	0.183	0.269	0.343	0.411	0.470
GLIE, training-free stage	0.204	0.271	0.344	0.413	0.466	**0.500**
GLIE, stage 1	0.232	0.275	0.343	0.411	0.445	0.493
GLIE, full	**0.256**	**0.295**	**0.359**	**0.417**	**0.471**	0.492
Uncompressed	0.525	0.525	0.525	0.525	0.525	0.525
InfoVQA						
Raw k-means	0.613	0.642	0.669	0.721	0.786	0.815
Token pooling	0.684	0.705	0.721	0.747	0.765	0.791
Cluster merging	0.620	0.603	0.641	0.686	0.744	0.792
GLIE, training-free stage	0.678	0.717	0.747	0.786	**0.818**	**0.832**
GLIE, stage 1	0.692	0.716	0.755	0.791	0.812	0.830
GLIE, full	**0.700**	**0.725**	**0.760**	**0.795**	0.813	0.831
Uncompressed	0.842	0.842	0.842	0.842	0.842	0.842
Shift Project						
Raw k-means	0.297	0.335	0.412	0.473	0.544	0.602
Token pooling	0.392	0.403	0.429	0.485	0.540	0.609
Cluster merging	0.273	0.256	0.273	0.358	0.387	0.566
GLIE, training-free stage	0.386	0.407	0.547	0.584	0.646	**0.693**
GLIE, stage 1	**0.406**	0.449	**0.565**	**0.599**	0.652	0.686
GLIE, full	0.389	**0.453**	0.552	0.597	**0.654**	0.691
Uncompressed	0.768	0.768	0.768	0.768	0.768	0.768
Synthetic AI						
Raw k-means	0.592	0.641	0.716	0.807	0.916	0.931
Token pooling	0.742	0.749	0.792	0.824	0.882	0.924
Cluster merging	0.548	0.561	0.579	0.717	0.836	0.926
GLIE, training-free stage	0.706	0.756	0.810	0.908	0.938	0.960
GLIE, stage 1	**0.783**	0.811	0.880	0.928	0.939	0.961
GLIE, full	0.773	**0.813**	**0.885**	**0.936**	**0.945**	**0.969**
Uncompressed	0.984	0.984	0.984	0.984	0.984	0.984

Table 12: ViDoRe v1 per subset, part 2 of 2. nDCG@5 over all queries, three-seed means for the learned rows. Best method per column in bold; the uncompressed row is the ceiling and does not compete.

Method	k=2	k=4	k=8	k=16	k=32	k=64
Synthetic Energy						
Raw k-means	0.551	0.613	0.666	0.791	0.890	0.920
Token pooling	0.691	0.710	0.732	0.735	0.812	0.873
Cluster merging	0.562	0.571	0.653	0.720	0.811	0.897
GLIE, training-free stage	0.662	0.704	0.814	0.871	**0.924**	**0.929**
GLIE, stage 1	0.740	0.757	**0.836**	**0.884**	0.922	**0.929**
GLIE, full	**0.787**	**0.794**	0.829	0.878	0.917	0.925
Uncompressed	0.938	0.938	0.938	0.938	0.938	0.938
Synthetic Gov.						
Raw k-means	0.514	0.585	0.691	0.741	0.852	0.905
Token pooling	0.639	0.713	0.747	0.791	0.873	0.899
Cluster merging	0.465	0.526	0.568	0.671	0.777	0.833
GLIE, training-free stage	0.624	0.705	0.782	0.825	0.912	**0.940**
GLIE, stage 1	0.735	0.769	0.805	0.868	0.915	0.929
GLIE, full	**0.736**	**0.784**	**0.822**	**0.879**	**0.919**	0.925
Uncompressed	0.950	0.950	0.950	0.950	0.950	0.950
Synthetic Health						
Raw k-means	0.683	0.701	0.804	0.856	0.890	0.938
Token pooling	0.771	0.802	0.860	0.869	0.904	0.922
Cluster merging	0.621	0.650	0.719	0.759	0.835	0.908
GLIE, training-free stage	0.757	0.780	0.868	0.901	0.932	0.955
GLIE, stage 1	0.823	0.845	**0.914**	**0.920**	**0.938**	**0.962**
GLIE, full	**0.840**	**0.862**	0.905	0.914	0.937	0.960
Uncompressed	0.976	0.976	0.976	0.976	0.976	0.976
TabFQuAD						
Raw k-means	0.601	0.617	0.723	0.792	0.832	0.844
Token pooling	0.606	0.652	0.700	0.734	0.760	0.801
Cluster merging	0.582	0.650	0.739	0.811	0.833	0.850
GLIE, training-free stage	0.621	0.691	0.768	0.822	0.841	0.852
GLIE, stage 1	0.629	0.709	0.785	0.823	0.841	**0.856**
GLIE, full	**0.653**	**0.721**	**0.790**	**0.825**	**0.847**	0.854
Uncompressed	0.852	0.852	0.852	0.852	0.852	0.852
TAT-DQA						
Raw k-means	0.222	0.240	0.333	0.397	0.501	0.578
Token pooling	0.283	0.319	0.369	0.403	0.448	0.536
Cluster merging	0.166	0.173	0.226	0.345	0.483	0.585
GLIE, training-free stage	0.297	0.356	0.434	0.507	0.577	0.626
GLIE, stage 1	0.322	0.392	0.466	0.537	0.586	0.633
GLIE, full	**0.350**	**0.405**	**0.469**	**0.542**	**0.597**	**0.638**
Uncompressed	0.700	0.700	0.700	0.700	0.700	0.700

Table 13: ViDoRe v2 per subset. Graded nDCG@5 over all queries. Best method per column in bold; the uncompressed row is the ceiling and does not compete.

Method	k=2	k=4	k=8	k=16	k=32	k=64
Biomedical lectures						
Raw k-means	0.187	0.243	0.321	0.378	0.446	0.508
Token pooling	0.296	0.322	0.347	0.388	0.422	0.460
Cluster merging	0.251	0.280	0.310	0.361	0.459	0.512
GLIE, training-free stage	0.285	0.356	0.419	0.461	0.510	0.548
GLIE, stage 1	0.314	0.385	0.435	0.472	0.518	0.550
GLIE, full	**0.338**	**0.401**	**0.459**	**0.494**	**0.530**	**0.562**
Uncompressed	0.564	0.564	0.564	0.564	0.564	0.564
Economics reports						
Raw k-means	0.350	0.323	0.331	0.361	0.378	0.372
Token pooling	0.369	0.343	0.323	0.345	0.357	0.392
Cluster merging	0.335	0.323	0.278	0.233	0.255	0.285
GLIE, training-free stage	0.308	0.344	0.377	0.405	0.403	0.415
GLIE, stage 1	0.362	0.398	**0.411**	0.429	**0.429**	0.438
GLIE, full	**0.403**	**0.403**	0.409	**0.434**	0.425	**0.451**
Uncompressed	0.468	0.468	0.468	0.468	0.468	0.468
ESG reports (human)						
Raw k-means	0.082	0.082	0.219	0.263	0.347	0.375
Token pooling	0.132	0.181	0.179	0.208	0.235	0.347
Cluster merging	0.098	0.057	0.068	0.143	0.250	0.364
GLIE, training-free stage	0.157	0.188	0.332	0.364	0.430	0.467
GLIE, stage 1	0.172	0.226	**0.368**	0.410	**0.443**	0.480
GLIE, full	**0.177**	**0.235**	0.352	**0.415**	**0.443**	**0.483**
Uncompressed	0.523	0.523	0.523	0.523	0.523	0.523
ESG reports						
Raw k-means	0.119	0.156	0.227	0.272	0.334	0.364
Token pooling	0.114	0.116	0.128	0.164	0.214	0.282
Cluster merging	0.069	0.065	0.102	0.142	0.178	0.293
GLIE, training-free stage	0.132	0.214	0.304	0.359	0.442	0.467
GLIE, stage 1	0.158	0.251	0.324	0.363	0.446	0.463
GLIE, full	**0.180**	**0.270**	**0.331**	**0.398**	**0.472**	**0.486**
Uncompressed	0.514	0.514	0.514	0.514	0.514	0.514