DEPTH-AS-BUDGET: PATCH-ADAPTIVE RESIDUAL QUANTIZATION FOR VISUAL DOCUMENT RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Late-interaction visual document retrieval (VDR) enables fine-grained patch matching, but its high storage and scoring costs make index compression essential at scale. The key challenge is to compress the index before future queries are known, even though these queries determine which patches matter for matching. Then, we propose **PARQ** (Patch-Adaptive Residual Quantization), which keeps every patch available for matching and distributes a fixed compression budget using document-side reconstruction gains. PARQ adaptively allocates residual codes based on reconstruction-error reduction. This allocation requires no query supervision or encoder fine-tuning. Grouped lookup-table scoring enables retrieval over variable-depth codes without reconstructing dense embeddings. On ViDoRe v1 and v2, PARQ retains 90.3% and 89.2% of the nDCG@5 of uncompressed ColPali and ColQwen2, respectively, at $64\times$ code-payload compression, and 97.9% for both at $26\times$. On a 1M-patch index, total query latency is 6.09 ms versus 39.11 ms for uncompressed ColPali.

1 INTRODUCTION

Retrieval-augmented generation (RAG) grounds language model outputs in retrieved external knowledge (Lewis et al., 2020; Borgeaud et al., 2022; Hu et al., 2023). For visually rich documents, text-only pipelines may miss evidence in layouts, tables, and figures (Xu et al., 2021; Wang et al., 2023). Visual document retrieval (VDR) addresses this limitation by using page images to capture such evidence (Faysse et al., 2025; Yu et al., 2025). Late-interaction VDR represents each page as a collection of visual patch embeddings and scores it by matching each query token to its highest-scoring patch before aggregating these token-wise maxima (Khattab & Zaharia, 2020; Faysse et al., 2025). This fine-grained interaction enables matching across text, tables, and figures, but it also makes the index expensive to store and score (Ma et al., 2025; Santhanam et al., 2022a). Compression is therefore essential at scale.

Existing compression approaches reduce storage by merging or pruning patches (Ma et al., 2025; Yan et al., 2025), lower scoring cost via single-vector candidate selection followed by multi-vector reranking (Dhulipala et al., 2024; Kim et al., 2025), or compress token embeddings using fixed-width residual codes (Santhanam et al., 2022b; Scheerer et al., 2025). These approaches make good progress in compression but discard the patch information which may be needed by unknown future queries. Note that the patches that matter for late-interaction scoring are query-dependent: different queries may select different MaxSim winners from the same page. This creates a fundamental mismatch between query-dependent patch importance and query-independent compression decisions. Therefore, one problem remains unresolved: *without query-specific information, how can we allocate a given compression budget across all patches?*

To this end, we introduce **PARQ** (Patch-Adaptive Residual Quantization) for allocating variable precision from document embeddings. To guide allocation without access to queries, we evaluate how accurately each patch embedding can be reconstructed from its compressed code. We define the reconstruction residual as the difference between the original patch embedding and its reconstruction, from which we derive an upper bound on MaxSim score distortion for any future query. Then, PARQ fits residual codebooks to unlabeled embeddings from a frozen encoder and adaptively allocates residual depth by marginal reconstruction-error reduction. The allocation requires no fu-

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.

ture queries, historical query logs, or relevance labels. At query time, the depths remain fixed, and grouped lookup tables (LUTs) enable direct MaxSim scoring without dense-vector reconstruction.

Our contributions are summarized as follows:

- We propose a document-only precision-allocation method that retains every visual patch and distributes a code budget according to marginal reconstruction gains, without query supervision or encoder fine-tuning.
- We establish a connection between patch reconstruction residuals and MaxSim score distortion, and demonstrate that the distribution of reconstruction errors—particularly in poorly reconstructed patches—matters for retrieval fidelity.
- We develop grouped-LUT scoring for variable-depth residual codes, enabling direct late-interaction retrieval without dense-vector reconstruction, and evaluate trade-offs between retrieval quality and execution cost on two visual backbones.

2 Related Work

Late-interaction compression and retrieval. ColBERT establishes token-level late interaction, and ColBERTv2 develops compact representations for this setting (Khattab & Zaharia, 2020; Santhanam et al., 2022b). PLAID, MUVERA, and WARP improve multi-vector retrieval through indexing, alternative representations, or execution strategies (Santhanam et al., 2022a; Dhulipala et al., 2024; Scheerer et al., 2025). Visual retrievers operate on image patches, with pruning, pooling, and hybrid representations reducing storage or computation (Faysse et al., 2025; Ma et al., 2025; Yan et al., 2025; Kim et al., 2025). Retaining the patch set is a design constraint of PARQ; our contribution concerns the precision assigned within that set and its execution. Our experiments use exhaustive compressed-index scoring and do not establish an end-to-end comparison with candidate-generation engines such as PLAID.

Quantization and adaptive coding. Product quantization, optimized product quantization, and residual or additive quantization provide established compact representations (Jégou et al., 2011; Ge et al., 2014; Juang & Gray, 1982; Chen et al., 2010; Babenko & Lempitsky, 2014). Neural codebooks further improve residual quantization (Huijben et al., 2024), and variable-rate residual coding already adapts codebook counts to input content, as in audio VRVQ (Chae et al., 2024). PARQ examines the retrieval consequences of allocating residual depth over frozen visual patches. Its setting combines an offline decision made without queries, a fixed retained patch set, and online MaxSim execution over variable-length codes. We evaluate how a document-only allocation signal performs under these constraints.

3 Method

PARQ builds a compressed index from document embeddings before queries arrive. We keep the patch set fixed, allocate its precision through residual prefixes, and execute the resulting codes when a query is available (Figure 1). The formulation uses a page-level code budget; Section 3.5 distinguishes code payload from physical storage.

3.1 Late Interaction and Residual Prefixes

Given a text query q, let $Q = \{q_i\}_{i=1}^{N_q}$ denote its query-token embeddings, where each $q_i \in \mathbb{R}^D$. For a document page d, let $X_d = \{x_j\}_{j=1}^{N_d}$ denote its patch embeddings, where each $x_j \in \mathbb{R}^D$. Following late-interaction retrieval (Faysse et al., 2025), the relevance score between q and d is computed as

$$s(q,d) = \sum_{i=1}^{N_q} \max_{1 \le j \le N_d} \langle q_i, x_j \rangle. \tag{1}$$

The storage and scoring cost of (1) scales with the number of document patches, since the full floating-point patch matrix must be stored and queried at retrieval time. A natural strategy is therefore to replace dense patch vectors with compact discrete codes.

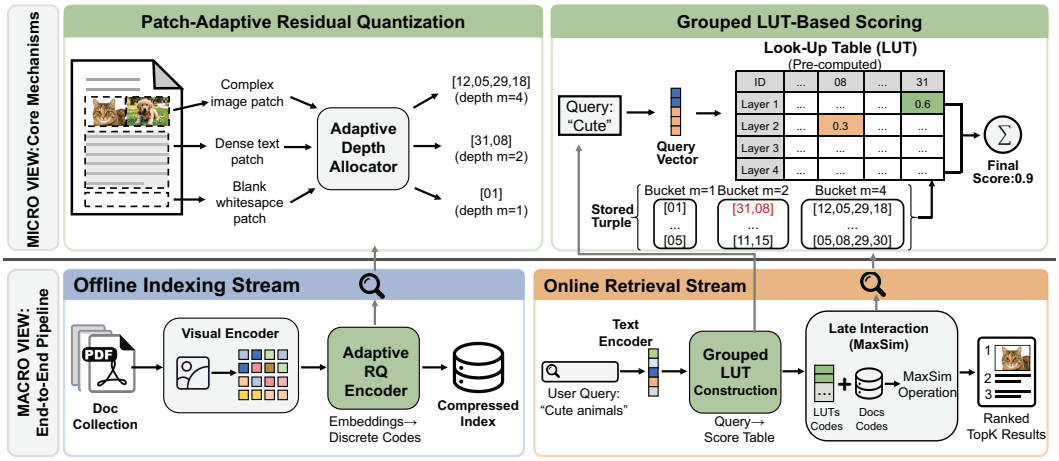


Figure 1: **PARQ framework.** Top: patch-adaptive residual coding and grouped LUT scoring. Bottom: offline index construction and online retrieval. Depth is allocated from document embeddings by marginal reconstruction gain under a shared code budget and remains fixed for future queries. All patch locations remain candidates; illustrated depths are schematic.

In standard RQ (Juang & Gray, 1982; Chen et al., 2010), a patch embedding x_j is approximated using a fixed number M of codebooks. Let $\{C_t\}_{t=1}^{M}$ denote stage-wise codebooks, where each $C_t \in \mathbb{R}^{K \times D}$ contains K centroids. The fixed-depth reconstruction is

$$\hat{x}_j^{(M)} = \sum_{t=1}^{M} C_t[b_{j,t}], \tag{2}$$

where $b_{j,t} \in \{1, \ldots, K\}$ is the selected centroid index at stage t. This formulation is attractive because it supports LUT scoring and compact storage. However, it allocates the same number of stages to all patches, regardless of the approximation benefit each patch would receive from an additional stage.

3.2 Allocating Precision with Every Patch Retained

To make precision allocatable, we assign a depth to each patch while keeping all N_d patch locations eligible in MaxSim. Let $M_{\max}$ denote the maximum residual depth, and let

$$m_j \in \{m_{\min}, \ldots, M_{\max}\} \tag{3}$$

be the assigned depth of patch x_j, where $m_{\min} \geq 1$ is a small minimum depth shared by all patches. We learn a family of codebooks $\{C_t\}_{t=1}^{M_{\max}}$, and reconstruct patch x_j using only the first m_j stages:

$$\hat{x}_j^{(m_j)} = \sum_{t=1}^{m_j} C_t[b_{j,t}]. \tag{4}$$

Fixed-depth RQ is the special case $m_j = M$ for all j. Each additional stage spends one codeword on refining an existing candidate patch. We choose the prefix lengths by minimizing a document-only distortion surrogate under a shared code budget:

$$\begin{aligned} \min_{\{m_j\}} \quad & \sum_{j=1}^{N_d} L_j(m_j) \\ \text{s.t.} \quad & \sum_{j=1}^{N_d} m_j \log_2 K \leq B_{\text{code}}, \\ & m_j \in \{m_{\min}, \ldots, M_{\max}\}. \end{aligned} \tag{5}$$

where B_{code} is shared by the N_d patches of page d and must satisfy $B_{\text{code}} \geq N_d m_{\min} \log_2 K$. Thus, the displayed problem allocates capacity within a page; it does not optimize the distribution of budgets across pages. In the page-major layout, bucket membership identifies depth without explicit per-patch page IDs, spatial offsets, or depth fields. Shared boundary structures are included in the storage accounting.

3.3 A Query-Agnostic Rule for Allocating Stages

The allocator must make its decisions without future query embeddings. We use the squared reconstruction error, which is computable from each patch and its residual prefixes:

$$L_j(m) = \|x_j - \hat{x}_j^{(m)}\|_2^2 \tag{6}$$

The relation of this surrogate to retrieval follows from the stability of the maximum operator. Write $e_j = x_j - \hat{x}_j^{(m_j)}$ and let $\hat{s}(q, d)$ be the MaxSim score over the approximated patches. For each query token,

$$\begin{aligned} &\left| \max_j \langle q_i, x_j \rangle - \max_j \langle q_i, \hat{x}_j^{(m_j)} \rangle \right| \\ &\quad \leq \max_j |\langle q_i, e_j \rangle| \leq \|q_i\|_2 \max_j \|e_j\|_2. \end{aligned} \tag{7}$$

Summing over query tokens and bounding the largest residual by the root of the total squared error gives

$$|s(q, d) - \hat{s}(q, d)| \leq \left(\sum_i \|q_i\|_2 \right) \sqrt{\sum_j L_j(m_j)}. \tag{8}$$

The inequality applies to any subsequent query, including cases in which quantization changes the winning patch. Its document-dependent factor is available at indexing time, giving a query-agnostic rationale for reducing reconstruction error.

The value of the next stage. We define the marginal benefit of extending patch j from depth $m-1$ to m as

$$\Delta_{j,m} = L_j(m-1) - L_j(m). \tag{9}$$

This quantity measures how much an additional codeword improves approximation. A large current residual need not imply a large next-stage gain, and neither is a query-relevance score. This distinction motivates the comparison with norm-based allocation in Section 4.2.

Each patch first receives $m_{\min}$ stages. We precompute stage-wise gains and repeatedly extend the patch with the largest currently available $\Delta_{j,m}$. Only the next stage of a patch's prefix is eligible. Every extension costs $\log_2 K$ bits, so selecting the largest gain is equivalent to selecting the largest gain per code bit. Allocation continues until the code budget is exhausted or all maximum depths are reached. The output for page d is $\{(m_j, b_{j,1:m_j})\}_{j=1}^{N_d}$; these depths remain fixed for all subsequent queries.

Conditional optimality. If each patch has non-negative, non-increasing marginal gains,

$$\Delta_{j,m_{\min}+1} \geq \cdots \geq \Delta_{j,M_{\max}} \geq 0, \tag{10}$$

greedy allocation is globally optimal for the page-level surrogate problem (Appendix C). Empirical checks find mostly diminishing gains and greedy–exact objective gaps below 0.3% on sampled pages (Appendix C.2).

3.4 Executing the Allocated Index with Grouped LUTs

Once a query arrives, its embeddings are available for scoring the already allocated index. The challenge is to execute unequal prefix lengths efficiently. We first compute query–centroid similarities in a lookup table:

$$T_{i,t}[k] = \langle q_i, C_t[k] \rangle, \tag{11}$$

where $i \in \{1, \ldots, N_q\}$, $t \in \{1, \ldots, M_{\max}\}$, and $k \in \{1, \ldots, K\}$.

Using (4), the approximate interaction between query token q_i and patch x_j becomes

$$\langle q_i, \hat{x}_j^{(m_j)} \rangle = \sum_{t=1}^{m_j} T_{i,t}[b_{j,t}], \tag{12}$$

which leads to the variable-depth late-interaction score

$$\hat{s}(q,d) = \sum_{i=1}^{N_q} \max_{1 \le j \le N_d} \sum_{t=1}^{m_j} T_{i,t}[b_{j,t}]. \tag{13}$$

Naively evaluating (13) introduces irregular control flow because different patches have different depths. To avoid this issue, we group patches by their assigned depth:

$$\mathcal{P}_m = \{j \mid m_j = m\}, m \in \{m_{\min}, \dots, M_{\max}\}. \tag{14}$$

We then evaluate patches within each bucket using regular batched accumulation over the first m stages, and take the maximum across buckets. Concretely,

$$\hat{s}(q,d) = \sum_{i=1}^{N_q} \max_{m} \max_{j \in \mathcal{P}_m} \sum_{t=1}^{m} T_{i,t}[b_{j,t}]. \tag{15}$$

Empty buckets are skipped. Grouping changes the execution order, not the MaxSim score of the quantized embeddings: Equation (15) is equal to Equation (13). The approximation arises from quantization itself. Within each bucket, code lengths are fixed, enabling batched lookups and additions without reconstructing patch vectors.

3.5 Code Budget, Storage, and Scoring Cost

For dimension D and b bits per dense coordinate, a patch requires Db bits before compression. With K a power of two, each residual code ID requires $\log_2 K$ bits. The ideal packed code payload for page d is

$$\mathrm{Bits}_{\mathrm{code}}(d) = \sum_{j=1}^{N_d} m_j \log_2 K = N_d \bar{m} \log_2 K, \tag{16}$$

where $\bar{m}$ is the average depth. Setting $\bar{m} = M$ matches the fixed-depth code budget, giving

$$\mathrm{Comp.}_{\mathrm{code}} = \frac{Db}{\bar{m} \log_2 K} = \frac{Db}{M \log_2 K}. \tag{17}$$

The nominal $26\times$ and $64\times$ settings denote code-payload ratios ($25.6\times$ rounded to $26\times$). Including all storage overhead, the measured 84K-page indexes achieve $25.51\times$/$63.86\times$ physical compression (Appendix B.2.1).

For a shared codebook family and N indexed patches, building the query LUT costs $O(N_q M_{\max} K D)$, and code accumulation costs $O(N_q \sum_{j=1}^{N} m_j)$. At matched average depth this accumulation count equals that of fixed-depth RQ, although more codebooks and bucket processing can add overhead. The computation remains linear in the number of indexed patches; Section 4.5 reports measured latency.

4 Experiments

We evaluate retrieval quality, the connection between reconstruction and retrieval fidelity, depth allocation, and execution cost. PARQ constructs its codebooks and depths from document embeddings alone. A separate query-weighted variant serves only as a diagnostic.

4.1 Experimental Setup

We evaluate retrieval quality on five ViDoRe v1 subsets and four ViDoRe v2 subsets (Faysse et al., 2025; Macé et al., 2025), listed in Appendix A. Results are the unweighted mean nDCG@5 across these nine subsets. All reported timing measurements use a single L20 (48GB) GPU.

Table 1: **Detailed Results on ViDoRe v1 and v2 (ColPali Backbone).** We compare adaptive-depth PARQ with dense retrieval, token-compression baselines, vector-quantization baselines, and fixed-depth RQ. **Comp.** denotes the code-payload compression ratio; implementation details are provided in Appendix B.2. **Avg.** reports the average nDCG@5 across all listed ViDoRe v1 and v2 subsets. **Reten.** indicates the retention rate relative to the dense baseline.

Method	Comp.	ViDoRe v1					ViDoRe v2				Avg.	Reten.
		Arxiv	Info	TabF	Shift	Health	ESG	Econ.	Insur.	Biomed.		
ColPali	1×	78.74	82.45	86.65	76.14	94.66	46.10	56.70	54.70	56.40	**70.28**	–
Token Compression Baselines												
DocPruner	6×	73.01	69.58	79.24	49.27	83.67	38.18	46.12	45.01	47.65	59.08	84.1%
	11×	61.03	57.27	64.38	32.12	75.68	32.25	38.28	37.66	40.37	48.78	69.4%
Light-ColPali	26×	76.85	80.31	85.74	70.24	93.95	43.91	53.47	51.58	54.04	67.79	96.4%
	64×	72.90	78.20	83.30	62.40	88.70	39.68	47.97	46.90	48.86	63.21	89.9%
Vector Quantization-based Compression												
BQ	16×	77.22	82.00	86.01	72.58	93.71	45.35	55.60	53.66	55.38	69.06	98.3%
OPQ	26×	75.10	78.20	84.40	67.80	91.20	43.05	52.71	50.86	52.63	66.22	94.2%
	64×	65.80	67.30	78.60	54.20	82.80	36.42	43.85	42.60	45.09	57.41	81.7%
Residual Depth Allocation: High-Capacity Budget, matched to fixed-depth ($M = 8, \text{nbits} = 10$)												
Fixed-depth	26×	76.64	80.06	86.92	69.56	94.10	44.92	55.05	53.27	54.61	68.35	97.2%
Random Adaptive	26×	74.80	78.10	85.50	67.20	92.40	43.75	53.54	51.77	53.01	66.67	94.9%
Norm Adaptive	26×	75.60	78.90	86.00	68.00	92.90	44.12	54.16	52.23	53.53	67.27	95.7%
Ours	**26×**	**77.20**	**80.90**	**87.20**	**70.40**	**94.30**	**45.30**	**55.53**	**53.72**	**54.99**	**68.84**	**97.9%**
Residual Depth Allocation: High-Compression Budget, matched to fixed-depth ($M = 4, \text{nbits} = 8$)												
Fixed-depth	64×	64.83	64.20	81.66	54.14	82.70	35.99	43.37	42.73	44.84	57.16	81.3%
Random Adaptive	64×	58.20	57.90	76.80	47.60	77.50	32.25	38.97	38.24	40.50	52.00	74.0%
Norm Adaptive	64×	62.30	61.80	78.90	50.80	80.10	34.12	41.37	40.59	43.11	54.79	78.0%
Ours	**64×**	**72.30**	**75.20**	**85.40**	**63.10**	**87.20**	**40.43**	**49.48**	**48.20**	**50.01**	**63.48**	**90.3%**

Comparisons. The uncompressed ColPali and ColQwen2 indexes provide reference scores. External compression baselines are binary quantization (BQ), optimized product quantization (OPQ) (Ge et al., 2014), DocPruner (Yan et al., 2025), and Light-ColPali/Light-ColQwen2 (Ma et al., 2025). Our principal allocation comparison is with **Fixed-depth RQ**, the uniform-depth special case. **Random Adaptive** and **Norm Adaptive** use random and norm-based depth assignment under matched code budgets; **PARQ** uses marginal residual-error reduction. All four RQ allocation variants keep every patch; their comparison examines precision assignment within that retained set. External baselines place the resulting quality in context.

Encoders and index construction. We use the reported ColPali and ColQwen2 backbones, both with $D = 128$ patch embeddings. PARQ operates on frozen embeddings and fits codebooks without queries or relevance labels, using up to 100k patches sampled from each target subset. The quantization comparisons share this domain-specific fitting protocol; they do not assume one universal codebook across domains. Appendix D.4 evaluates codebook transfer. The main settings use 80 and 32 code bits per patch on average, corresponding to $26\times$ and $64\times$ code-payload compression. Implementation and scoring protocols are provided in Appendix B.

4.2 Retrieval under Allocated Precision

Different allocations, the same retained patch set. Tables 1 and 2 compare allocation rules under matched code budgets. At $64\times$, fixed-depth RQ scores 57.16 on ColPali and 63.05 on ColQwen2. Random Adaptive scores 52.00/57.83, and Norm Adaptive scores 54.79/60.60. PARQ reaches 63.48/68.82, improving over fixed depth by 6.32/5.77 points. Thus, the observed gains depend on the allocation criterion; variable code length alone does not improve retrieval.

Refinement becomes more consequential at the tighter budget. At $26\times$, fixed-depth RQ already reaches 68.35/75.05, and PARQ improves this to 68.84/75.50, gains of 0.49/0.45 points. PARQ retains 97.9% of the uncompressed score on both backbones at this budget, versus 90.3%/89.2%

Table 2: **Detailed Results on ViDoRe v1 and v2 (ColQwen2 Backbone).**

Method	Comp.	ViDoRe v1					ViDoRe v2				Avg.	Reten.
		Arxiv	Info	TabF	Shift	Health	ESG	Econ.	Insur.	Biomed.		
ColQwen2	1×	88.90	91.53	89.63	85.96	97.29	57.40	59.80	60.30	63.60	**77.16**	–
Token Compression Baselines												
DocPruner	31×	39.09	22.98	32.42	20.29	37.29	22.41	21.44	22.47	26.50	27.21	35.3%
Light-ColQwen2	26×	85.92	87.50	88.07	77.95	96.79	54.25	56.05	56.85	60.92	73.81	95.7%
	64×	80.90	82.40	85.20	69.80	91.20	49.47	50.58	51.57	55.96	68.56	88.9%
Vector Quantization-based Compression												
BQ	16×	87.36	91.40	87.99	85.13	96.62	56.52	58.78	59.28	62.57	76.18	98.7%
OPQ	26×	84.90	87.40	86.80	79.30	94.10	53.94	55.84	56.31	59.95	73.17	94.8%
	64×	73.80	76.20	80.40	61.50	86.10	46.32	46.97	47.64	51.76	63.41	82.2%
Residual Depth Allocation: High-Capacity Budget, matched to fixed-depth ($M = 8, \text{nbits} = 10$)												
Fixed-depth	26×	87.23	89.37	88.43	81.86	95.37	55.51	57.68	58.27	61.74	75.05	97.3%
Random Adaptive	26×	85.80	87.70	87.20	79.90	94.00	54.57	56.52	56.98	60.43	73.68	95.5%
Norm Adaptive	26×	86.40	88.30	87.70	80.70	94.50	54.95	57.00	57.46	60.92	74.21	96.2%
Ours	**26×**	**87.80**	**89.80**	**88.90**	**82.50**	**95.70**	**55.89**	**58.16**	**58.68**	**62.09**	**75.50**	**97.9%**
Residual Depth Allocation: High-Compression Budget, matched to fixed-depth ($M = 4, \text{nbits} = 8$)												
Fixed-depth	64×	72.10	73.80	84.20	59.70	86.10	45.69	46.62	47.51	51.69	63.05	81.7%
Random Adaptive	64×	65.80	66.20	80.10	52.80	81.70	41.79	42.12	43.04	46.94	57.83	75.0%
Norm Adaptive	64×	69.10	70.40	82.00	56.40	84.20	43.93	44.44	45.48	49.42	60.60	78.5%
Ours	**64×**	**79.20**	**81.60**	**87.10**	**68.30**	**90.80**	**50.41**	**52.02**	**53.06**	**56.85**	**68.82**	**89.2%**

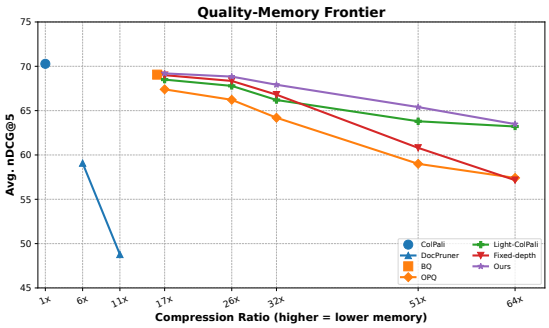
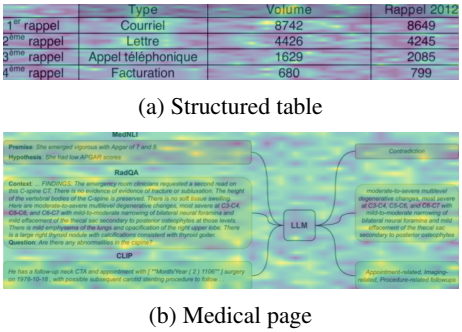


Figure 2: **Quality–Memory Frontier.** Average nDCG@5 under different compression ratios.



(a) Structured table

(b) Medical page

Figure 3: **Spatially non-uniform allocation.** Depth allocation on two document types. Brighter regions receive deeper residual codes.

at 64×. The stronger improvement at 64× is consistent with greater sensitivity to allocation when capacity is scarce. However, the two settings change both stage count and codebook size: ($M = 8, \text{nbits} = 10$) versus ($M = 4, \text{nbits} = 8$). Separate depth-range and codebook-size ablations at matched payload appear in Appendix D.3.

Comparison with other compression methods. At 64×, OPQ obtains 57.41/63.41, below PARQ's 63.48/68.82. Light-ColPali and Light-ColQwen2 remain competitive at 63.21/68.56: PARQ improves on these scores by 0.27/0.26 points. The published Light-ColPali/ColQwen2 recipe fine-tunes the encoder with merged embeddings using supervised LoRA training (Ma et al., 2025). PARQ obtains these gains by fitting codebooks to frozen patch embeddings, avoiding the additional encoder fine-tuning and its compute requirements. BQ retains 98.3%/98.7% at 16×, a less compressed operating point. DocPruner is reported at different achieved ratios and therefore does not form an exactly matched-budget comparison. Figure 2 compares average nDCG@5 across compression ratios. Appendix B.5 reports the OPQ parameter search.

Table 3: **Matched-distortion diagnostics.** ColPali, 32 code bits per patch on average ($64\times$). Controls A–C match average residual distortion while varying error in the highest-residual 10% of patches. Fixed-depth RQ and PARQ provide baseline comparisons at the same code budget.

Allocation	Residual ↓	Top-10% residual ↓	MaxSim error ↓	Rank flips ↓	nDCG@5 ↑
Fixed-depth	0.162	0.510	8.4%	18.0%	57.16
PARQ	0.137	0.370	6.1%	11.0%	63.48
A: reduced tail error	0.138	0.290	6.0%	10.5%	63.60
B: constrained random swaps	0.138	0.410	6.6%	12.5%	62.70
C: concentrated tail error	0.138	0.550	7.4%	15.0%	61.40

4.3 Does Reconstruction Fidelity Track Retrieval?

At ColPali $64\times$, PARQ reduces relative MaxSim error from 8.4% to 6.1% versus fixed depth and preserves 78.6% rather than 71.8% of dense winners. Residual distortion, score error, and ranking quality vary consistently across the tested allocations. A held-out-query winner-weighted diagnostic adds 0.64 nDCG@5 points over PARQ. Appendix F reports the complete metrics, correlations, and diagnostic construction.

Controlling average reconstruction error. To distinguish average fidelity from the distribution of errors, we construct three ColPali allocations at $64\times$ with identical code budgets and approximately matched distortion. Starting from PARQ, within-page stage exchanges preserve code prefixes and keep each page's reconstruction loss within 1% of its reference. A reduces error in the highest-residual 10% of patches, B samples eligible exchanges randomly, and C concentrates error in this tail. Queries are used only after allocation, for evaluation. Appendix F.3 gives the complete construction protocol.

Evaluation metrics. Overall and top-10% residuals use a shared, fixed normalization across allocations. MaxSim error denotes relative score error. Rank flips measure the fraction of strict dense document-pair orderings reversed by compression within each query's fixed dense Top-20 document set, averaged over queries with comparable pairs. Compressed ties do not count as reversals. nDCG@5 is evaluated over the full retrieval collection, independently of this diagnostic candidate set.

Residual concentration matters. Table 3 shows that A–C share an average residual of 0.138 but differ in retrieval quality. Compared with C, A reduces top-10% residual from 0.550 to 0.290, lowers rank flips by 4.5 percentage points, and improves nDCG@5 by 2.20 points. B lies between them on all retrieval metrics. Average reconstruction error alone therefore does not characterize retrieval fidelity in these controls.

PARQ likewise has smaller tail residuals than fixed depth (0.370 versus 0.510), alongside lower MaxSim error and fewer rank flips. This pattern is consistent with reduced residual concentration contributing to its retrieval gains. A exceeds PARQ by 0.12 nDCG@5 points, leaving room for further refinement; the evidence supports a reconstruction-based surrogate without establishing ranking optimality.

4.4 How Is Precision Distributed across Patches?

Depth distributions over 288,680 patches (Appendix D.2) show that PARQ uses a range of depths within each budget. At average depth 4, 67.46% of patches receive depth at most 4 and 4.67% receive depth at least 8. The distribution shows how a fixed average code length supports both broad patch retention and selective refinement.

Figure 3 shows representative depth allocation on a table and a medical page. Depth varies within tables and text regions, illustrating heterogeneous allocation. These maps characterize coding capacity rather than query-specific winners. Together with the distributions, they show how PARQ departs from uniform precision; the matched-budget comparisons in Section 4.2 measure the resulting retrieval quality.

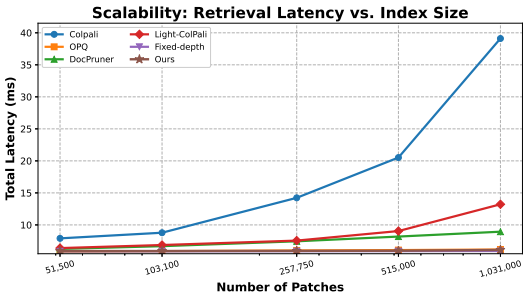


Figure 4: **Scalability Analysis.** Retrieval latency under increasing index sizes.

Table 4: **Latency breakdown.** Per-query latency in milliseconds on a 1M-patch index.

Method	Enc.	LUT	Bkt.	Score	Total
ColPali	5.70	–	–	33.41	39.11
DocPruner	5.70	–	–	3.24	8.94
Light-ColPali	5.70	–	–	7.52	13.22
BQ	5.70	–	–	29.34	35.04
OPQ	5.70	0.03	–	0.43	6.16
Fixed-depth	5.70	0.04	–	0.18	5.92
Random Adaptive	5.70	0.05	0.08	0.31	6.14
Norm Adaptive	5.70	0.05	0.08	0.29	6.12
Ours	5.70	0.05	0.08	0.26	**6.09**

4.5 What Does Variable-Depth Execution Cost?

Table 4 reports total per-query latency of 39.11 ms for uncompressed ColPali, 13.22 ms for Light-ColPali, and 6.09 ms for PARQ on a 1M-patch index. Fixed-depth RQ takes 5.92 ms, so PARQ adds 0.17 ms in this setting. Random and norm-based allocation incur similar total latency without the retrieval gains.

Query encoding contributes 5.70 ms to every row and dominates the compressed methods' totals. Excluding encoding, fixed-depth RQ takes 0.22 ms and PARQ takes 0.39 ms; the scoring components alone are 0.18 and 0.26 ms. Thus, the small absolute increase in total latency does not imply negligible relative scoring overhead. Figure 4 shows the measured range, while the code-accumulation cost still grows with index size. These measurements quantify the cost of supporting variable depth with grouped execution.

5 Conclusion

PARQ addresses precision allocation in visual document indexes built without knowledge of future queries. It keeps every patch available for matching, assigns residual-code prefixes by marginal reconstruction gain, and executes the fixed allocation through grouped LUT scoring. Across the evaluated backbones, marginal-gain allocation improves retrieval over fixed depth, while random and norm-based allocation reduce quality at the same code budgets. The gains are largest in the $64\times$ setting. These findings support precision allocation as a useful design choice for retaining a full patch set under tight compression, with no encoder fine-tuning or query supervision.

AI Use Statement

AI tools assisted with manuscript preparation, including language polishing and structural editing

REFERENCES

Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022.

Yunkee Chae, Woosung Choi, Yuhta Takida, Junghyun Koo, Yukara Ikemiya, Zhi Zhong, Kin Wai Cheuk, Marco A. Martínez-Ramírez, Kyogu Lee, Wei-Hsiang Liao, and Yuki Mitsufuji. VRVQ: Variable bitrate residual vector quantization for audio compression. In *Workshop on Machine Learning and Compression, NeurIPS 2024*, 2024. URL https://openreview.net/forum?id=EuPaZLX657.

Yongjian Chen, Tao Guan, and Cheng Wang. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273, 2010. ISSN 1424-8220. doi: 10.3390/s101211259. URL https://www.mdpi.com/1424-8220/10/12/11259.

Laxman Dhulipala, Majid Hadian, Rajesh Jayaram, Jason Lee, and Vahab Mirrokni. Muvera: Multi-vector retrieval via fixed dimensional encoding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 101042–101073. Curran Associates, Inc., 2024. doi: 10.52202/079017-3204. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b71cfefae46909178603b5bc6c11d3ae-Paper-Conference.pdf.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Colpali: Efficient document retrieval with vision language models. In *ICLR*, 2025. URL https://openreview.net/forum?id=ogjBpZ8uSi.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2014. doi: 10.1109/TPAMI.2013.240.

Ziniu Hu, Ahmet Iscen, Chen Sun, Zirui Wang, Kai-Wei Chang, Yizhou Sun, Cordelia Schmid, David A. Ross, and Alireza Fathi. Reveal: Retrieval-augmented visual-language pre-training with multi-source multimodal knowledge memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23369–23379, June 2023.

Iris A. M. Huijben, Matthijs Douze, Matthew Muckley, Ruud J. G. van Sloun, and Jakob Verbeek. Residual quantization with implicit neural codebooks, 2024. URL https://arxiv.org/abs/2401.14732.

Biing-Hwang Juang and A. Gray. Multiple stage vector quantization for speech coding. In *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pp. 597–600, May 1982. doi: 10.1109/ICASSP.1982.1171604.

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, Jan 2011. ISSN 1939-3539. doi: 10.1109/TPAMI.2010.57.

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*, SIGIR '20, pp. 39–48, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

Juyeon Kim, Geon Lee, Dongwon Choi, Taeuk Kim, and Kijung Shin. Hybrid-vector retrieval for visually rich documents: Combining single-vector efficiency and multi-vector accuracy. *arXiv preprint arXiv:2510.22215*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

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, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1003. URL https://aclanthology.org/2025.findings-acl.1003/.

Quentin Macé, António Loison, and Manuel Faysse. Vidore benchmark v2: Raising the bar for visual retrieval, 2025. URL https://arxiv.org/abs/2505.17166.

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 & Knowledge Management*, CIKM '22, pp. 1747–1756, New York, NY, USA, 2022a. Association for Computing Machinery. ISBN 9781450392365. doi: 10.1145/3511808.3557325. URL https://doi.org/10.1145/3511808.3557325.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, Seattle, United States, July 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.272. URL https://aclanthology.org/2022.naacl-main.272/.

Jan Luca Scheerer, Matei Zaharia, Christopher Potts, Gustavo Alonso, and Omar Khattab. Warp: An efficient engine for multi-vector retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pp. 2504–2512, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3729904. URL https://doi.org/10.1145/3726302.3729904.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. Vrdu: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 5184–5193, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2579–2591, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.201.

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

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. VisRAG: Vision-based retrieval-augmented generation on multi-modality documents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zG459X3Xge.

A Dataset Details

We evaluate five ViDoRe v1 subsets (Arxiv, Info, TabF, Shift, and Health) and four ViDoRe v2 subsets (ESGReports, Economics, Insurance, and Biomedical). They cover scientific pages, infographics, tables, layout variation, health material, and domain-specific reports. Scores are the unweighted mean nDCG@5 across these nine subsets. Methods use the same query–document splits. RQ allocation variants share frozen patch embeddings; comparisons with other method families additionally depend on their checkpoints and preprocessing.

B Implementation and Evaluation Details

B.1 Codebook Training Protocol

For each target subset, we randomly sample up to 100k unlabeled 128-dimensional patch embeddings from the indexed collection using the frozen ColPali or ColQwen2 encoder. OPQ, BQ, fixed-depth RQ, and PARQ share this domain-specific fitting protocol, without queries or relevance labels. Appendix D.4 instead fits codebooks on one source domain and transfers them without retraining.

B.2 Compression and Physical Storage

Each dense patch uses $128 \times 16 = 2048$ bits in `bfloat16`. In Equation (17), $(M, \mathrm{nbits}) = (8, 10)$ and $(4, 8)$ give 80-bit and 32-bit payloads: $25.6\times$ (reported as $26\times$) and $64\times$. PARQ matches these budgets with average depths 8 and 4. BQ uses one bit per dimension, giving 128 bits per patch and $16\times$ compression; OPQ configurations are listed below.

B.2.1 Physical Index Size

For collection $\mathcal{D}$, let $\mathcal{I}_{\mathcal{D}}$ comprise all serialized index files, including patch codes and the required codebooks. Physical storage is measured directly:

$$B_{\mathrm{physical}} = 8\ \mathrm{size}_{\mathrm{bytes}}(\mathcal{I}_{\mathcal{D}}). \tag{18}$$

A family of $M_{\max}$ codebooks uses $M_{\max} K D b_c$ bits for b_c-bit coordinates; each required family is counted once. Page/bucket boundary information, if stored, is part of the serialized layout, not an additional per-patch PARQ field. Alignment or padding is counted only when such bytes are actually stored; neither is required by the quantization formulation. Any such bytes are already included in the measured total and are not added again as estimated overhead. Table 5 reports an 84K-page index, ten times the evaluation collection at the same patch density. The zero row denotes fields that are not stored, as explained in Appendix B.3.

Table 5: **Measured storage for an 84K-page index.** MB is decimal. Physical compression uses the measured total across index files; nominal settings refer to code payload. Component rows reproduce the reported breakdown; auxiliary bytes are not separately itemized.

Component	$26\times$ setting	$64\times$ setting
Patch code IDs (MB)	864.60	345.80
Shared codebooks (MB)	3.15	0.79
Explicit per-patch page/depth/offset fields (MB)	0	0
Total serialized index (MB)	**867.75**	**346.59**
Physical compression	$\mathbf{25.51}\times$	$\mathbf{63.86}\times$

Codebooks add 0.36%/0.23% relative to patch-code storage at the two budgets and are amortized across the collection. This measurement concerns storage; timing uses the patch counts specified in the latency experiments.

B.3 Indexing and Baseline Settings

The index is page-major, with patches grouped by depth within each page. Page association follows the page-major organization, depth follows bucket membership, and MaxSim requires no spatial patch position. Thus, no explicit page ID, spatial offset, or depth field is stored per patch. Any page-boundary information belongs to the common page-major index structure, not an additional per-patch overhead introduced by PARQ. All methods use exhaustive scoring over their retained patch sets on the same corpus.

DocPruner uses $k = 0$ for ColPali $6\times$, $k = 0.25$ for ColPali $11\times$, and $k = -0.03$ for ColQwen2 $31\times$. Its achieved ratio is measured after adaptive pruning. These operating points are more aggressive than retaining 40–50% of patches (2–$2.5\times$ token-storage compression). Light-ColPali/ColQwen2 use pooling factors 26 and 64. We use official checkpoints or released configurations where available, otherwise reproducing the released settings on frozen embeddings.

Encoder-training cost. The original Light-ColPali/ColQwen2 study uses over 130k training queries and five epochs of LoRA fine-tuning (Ma et al., 2025). Its Table 3 reports 9.0 hours on eight A100 GPUs for Light-ColQwen2-2B (72 GPU-hours). This is a published offline training cost, separate from our query-latency measurements. PARQ uses up to 100k frozen patch embeddings for unsupervised codebook fitting and requires no encoder backpropagation or labeled query–document pairs.

B.4 Latency Protocol

Timing uses the same indexed corpus, batch setting, and candidate pool across methods and excludes disk I/O and data loading. Table 4 separates query encoding, LUT construction, bucketing, and scoring. Dense/token-compression methods use dot-product scoring; quantized methods use LUT scoring, with grouped execution for PARQ. No approximate candidate generation or reranking is used.

B.5 OPQ Tuning

OPQ code length is $B = M_{\text{sub}} n_{\text{bits}}$. Table 6 gives the complete ColPali search under the shared fitting protocol; bold entries are the configurations used in the main comparison.

Table 6: **OPQ configuration search on ColPali.** Every configuration uses either 80 or 32 code bits per patch. Bold entries identify the selected configuration within each budget.

Compression	M_{sub}	n_{bits}	nDCG@5	Retention
$26\times$	8	10	**66.22**	**94.2%**
$26\times$	16	5	65.88	93.7%
$64\times$	4	8	**57.41**	**81.7%**
$64\times$	8	4	56.94	81.0%
$64\times$	16	2	48.86	69.5%

C Greedy Allocation: Guarantee and Empirical Check

C.1 Conditional Optimality

After assigning minimum depth, let $T = M_{\max} - m_{\min}$ and $y_j = m_j - m_{\min} \in \{0, \dots, T\}$. The remaining number of equal-cost stages is

$$R = \left\lfloor \frac{B_{\text{code}} - N_d m_{\min} \log_2 K}{\log_2 K} \right\rfloor. \tag{19}$$

The utility of y_j extra stages is $G_j(y_j) = L_j(m_{\min}) - L_j(m_{\min} + y_j)$. Maximizing $\sum_j G_j(y_j)$ under $\sum_j y_j \le R$ is equivalent to the surrogate problem in Equation (5). Write $\delta_{j,t} = G_j(t) - G_j(t-1)$; only a patch's next unselected stage is eligible.

Theorem 1. *If every patch satisfies $\delta_{j,1} \geq \cdots \geq \delta_{j,T} \geq 0$ and all stages have equal storage cost, selecting the largest eligible marginal gain at each step maximizes the budgeted surrogate utility.*

Proof. Let $R' = \min\{R, N_d T\}$. Non-negative gains allow an optimum with exactly R' stages. Relaxing prefix constraints gives the upper bound obtained by selecting the R' largest gains. Sort all patch–stage pairs by non-increasing gain, breaking ties so that a patch's earlier stages precede its later ones. Diminishing gains make the first R' pairs prefix-feasible, so they attain this bound.

At each greedy step, any unavailable stage has an unselected predecessor with at least as large a gain. Hence a largest remaining gain is always eligible. Greedy selects R' such gains and attains the same upper bound, including under equal-gain ties. □

C.2 Empirical Check

We measure monotonicity of adjacent gains and complete patch chains, then compare greedy with exact dynamic programming on sampled pages (Table 7). Violations are small and objective gaps are below 0.3%, although the sufficient condition does not hold for every patch. The theorem remains conditional and concerns reconstruction utility rather than ranking.

Table 7: **Empirical behavior of the greedy allocator.** The objective gap compares greedy with exact dynamic programming on sampled pages; it is not an nDCG@5 gap.

Metric	ColPali	ColQwen2
Monotone adjacent stage pairs	94.7%	93.9%
Fully monotone patch chains	81.6%	79.8%
Mean violation / mean gain	1.9%	2.1%
Greedy–exact objective gap	0.18%	0.23%

D Codebook and Allocation Analysis

D.1 Training Data Size

Figure 5 varies the codebook-training sample at three budgets. Scores improve most between 1k and 50k patches and largely saturate at 50k–100k; the effect is strongest at $64\times$.

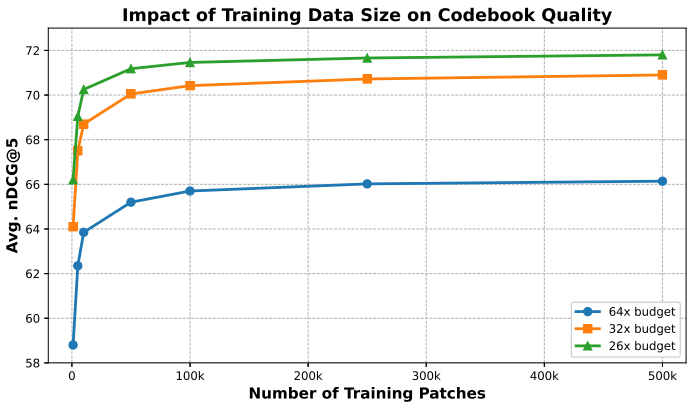


Figure 5: **Codebook training size.** Average nDCG@5 under three code-payload budgets.

D.2 Depth Distribution

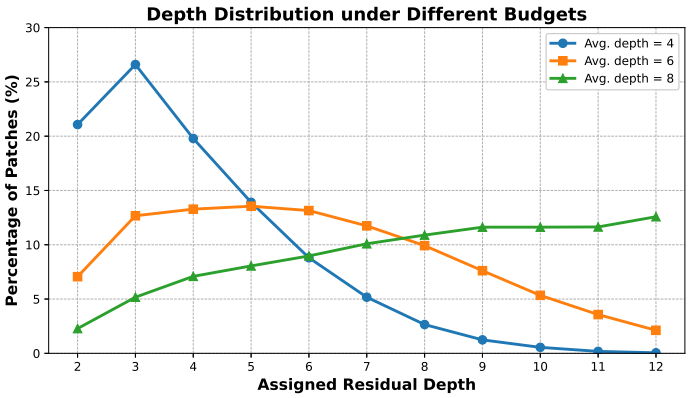


Figure 6: **Depth distribution.** Assigned depths over 288,680 patches at three target average depths, with allowed depths from 2 to 12.

At average depth 4, 67.46% of patches receive depth at most 4 and 4.67% at least 8. At average depth 6, 18.64% receive depth at least 9; at average depth 8, depths 8–12 account for 58.35%. Standard deviations are 1.77, 2.58, and 2.84, respectively.

D.3 Sensitivity to Range, Codebook Size, and Budget

Table 8 holds average payload at 32 bits per patch for the depth-range and codebook-size sweeps; average depth adjusts when bits per stage change. The final group varies the budget. The default $[2, 12]$ range and 8-bit stages perform best among the tested settings. Allocation-rule ablations are reported in Table 1.

Table 8: **Sensitivity of PARQ on ColPali.** Scores are mean nDCG@5. The first two groups hold average code payload fixed at 32 bits per patch; the final group varies the budget.

Factor	Setting	nDCG@5
Depth range	$[2, 8]$	62.91
	$[2, 12]$	**63.48**
	$[2, 16]$	63.34
Bits per stage	4	61.92
	8	**63.48**
	10	62.87
Code-payload compression	26×	68.84
	32×	67.92
	51×	65.40
	64×	63.48

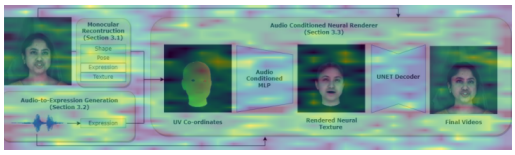
D.4 Cross-Domain Generalization

Table 9 fits codebooks on one source subset and transfers them to all targets without retraining, keeping the allocation rule fixed. Transfer preserves quality on related domains but degrades under distribution shift: Arxiv and TabF fitting perform best on their respective source domains, while ESGReports fitting transfers better to report-style ViDoRe v2 domains. The specialist row uses target-specific fitting as in Appendix B.1.

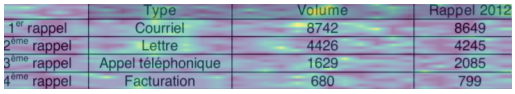
Table 9: **Codebook transfer on ColPali at** $64\times$**.** Specialist codebooks are fitted separately to each target subset; the other rows reuse one source-domain codebook family.

Training Source	Arxiv	Info	TabF	Shift	Health	ESG	Econ.	Insur.	Biomed.	Avg.
Specialist	**72.30**	**75.20**	**85.40**	**63.10**	**87.20**	**40.43**	**49.48**	**48.20**	**50.01**	**63.48**
Train on Arxiv	72.05	69.42	80.16	56.31	82.74	36.20	43.75	42.61	44.80	58.67
Train on TabF	66.83	70.05	84.72	57.80	81.36	35.80	43.10	41.85	43.70	58.36
Train on ESGReports	64.28	67.44	78.95	55.12	79.86	40.05	48.92	47.85	49.55	59.11

E VISUALIZATION OF PATCH-LEVEL DEPTH ALLOCATION

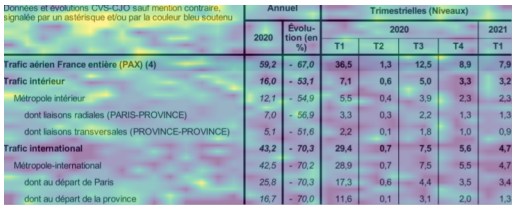


(a) Model diagram



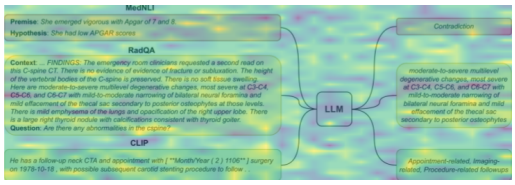
	Type	Volume	Rappel 2012
1er rappel	Courriel	8742	8649
2ème rappel	Lettre	4426	4245
3ème rappel	Appel téléphonique	1629	2085
4ème rappel	Facturation	680	799

(b) Structured table



Données et évolutions CVS-CJO sauf mention contraire, signalée par un astérisque et/ou par la couleur bleu soutenu	Annuel		Trimestrielles (Niveaux)				
	2020	Évolution (en %)	2020				2021
			T1	T2	T3	T4	T1
Trafic aérien France entière (PAX) (4)	59,2	- 67,0	36,5	1,3	12,5	8,9	7,9
Trafic intérieur	16,0	- 53,1	7,1	0,6	5,0	3,3	3,2
Métropole intérieur	12,1	- 54,9	5,5	0,4	3,9	2,3	2,3
dont liaisons radiales (PARIS-PROVINCE)	7,0	- 56,9	3,3	0,3	2,2	1,3	1,3
dont liaisons transversales (PROVINCE-PROVINCE)	5,1	- 51,6	2,2	0,1	1,8	1,0	0,9
Trafic international	43,2	- 70,3	29,4	0,7	7,5	5,6	4,7
Métropole-international	42,5	- 70,2	28,9	0,7	7,5	5,5	4,7
dont au départ de Paris	25,8	- 70,3	17,3	0,6	4,4	3,5	3,4
dont au départ de la province	16,7	- 70,0	11,6	0,1	3,1	2,0	1,3

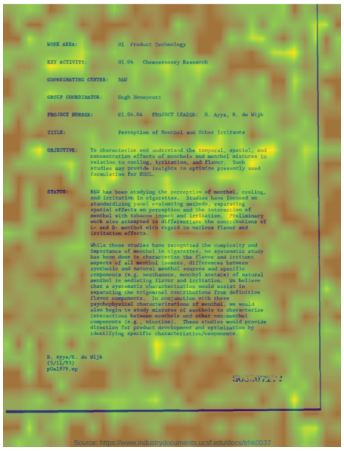
(c) Numerical table



(d) Medical reasoning



(e) Map document



(f) Technical report



Entreprises	Prime perçue	Caractéristiques	Bases de calcul
Seita (6 400 salariés)	1 500 F*	égalitaire	performances, commerciales/ résultats
Bonduelle (4 100 salariés)	1 900 F (salaire moyen)	60 % égalitaire 40 % proportion nette au salaire	résultat d'exploitation consolidé
IBM (22 000 salariés)	2 100 F* (salaire moyen non-cadre)	proportionnelle au salaire	productivité
Indosuez (2 700 salariés)	3 500 F	égalitaire	résultats, absentéisme
Elsydel (190 salariés)	4 000 F	égalitaire	résultats nets
Glaenzer-Spicer (1 520 salariés)	4 000 F*	égalitaire	productivité, qualité
Pont-à-Mousson (4 200 salariés)	4 000 F*	50 % égalitaire 50 % hiérarchisée	performances
Renault (71 900 salariés)	4 580 F (salaire minimum)	proportionnelle au salaire	résultats, performances par établissement
Générale sucrière (2 300 salariés)	5 400 F (salaire moyen ouvrier)	taux maximum 5 % du salaire annuel des ouvriers 10 % de celui des cadres supérieurs	performances par établissement, bénéfice net, dividendes versés aux actionnaires
Degrémont (750 salariés)	8 450 F (salaire moyen	50 % égalitaire 50 % hiérarchisée	résultats

(g) Financial table

Figure 7: **Examples of patch-level depth allocation.** We visualize adaptive residual-depth allocation on representative visual documents, including diagrams, medical reasoning pages, maps, and tables. Brighter regions indicate patches assigned deeper residual codes. The examples show spatially non-uniform code lengths across text, tables, diagrams, and background regions. Depth measures assigned coding capacity; it is not a query-relevance annotation.

Figure 7 presents additional depth maps, including the two examples shown in Figure 3. They cover diagrams, tables, a medical page, a map, and a technical report. Brighter regions indicate larger assigned residual depth. The maps reveal variation within a page, including around text blocks, boundaries, labels, and tabular structures.

These observations are consistent with heterogeneous approximation demands, but they are qualitative. No query, dense MaxSim winner, or counterfactual ranking is shown. Consequently, the figures cannot establish that the deepest patches are the most relevant or that a specific allocation preserves a winning match. Appendix F complements these qualitative examples with aggregate score-error and winner-preservation measurements.

F Reconstruction and Retrieval Diagnostics

F.1 Score Error and Winner Preservation

Table 10 evaluates the constructed indexes with queries; these diagnostics are not inputs to PARQ's allocation. The four metrics consistently favor marginal-gain allocation over fixed, random, and norm-based depth assignment.

Table 10: **Retrieval diagnostics under matched code budgets.** ColPali, $64\times$. Residual denotes normalized residual distortion; MaxSim error is relative score error; winner preservation measures agreement with dense MaxSim winners.

Allocator	Residual ↓	MaxSim error ↓	Winners ↑	nDCG@5 ↑
Fixed-depth	0.162	8.4%	71.8%	57.16
Random Adaptive	0.193	11.2%	63.4%	52.00
Norm Adaptive	0.179	9.7%	67.2%	54.79
PARQ	**0.137**	**6.1%**	**78.6%**	**63.48**

Across evaluated allocators, budgets, and subsets, residual distortion correlates with MaxSim error (Spearman $\rho = 0.91$), and winner preservation with nDCG@5 ($\rho = 0.88$). These cross-condition associations support the surrogate without guaranteeing ranking optimality.

F.2 Query-Informed Diagnostic

Using held-out queries, let w_j estimate patch j's dense-winner frequency and ϵ be a smoothing term. The diagnostic keeps prefix constraints and the code budget, but changes priorities to

$$\widetilde{\Delta}_{j,m} = (\epsilon + w_j)\Delta_{j,m}. \tag{20}$$

On ColPali at $64\times$, it reaches 64.12 nDCG@5, a 0.64-point gain over query-agnostic PARQ. This measures the benefit of available query information; it is neither an oracle upper bound nor part of PARQ's deployment requirements.

F.3 Matched-Distortion Controls

This appendix details the allocation controls in Section 4.3 and Table 3.

Experimental setup. All configurations use a frozen ColPali encoder and shared 256-entry stage codebooks, with 8-bit stages, depths in $[2, 12]$, and mean depth 4. Starting from PARQ, each exchange transfers one stage between two patches on the same page, preserving valid prefixes and the page code budget. Accepted allocations keep each page's reconstruction loss within 1% of its PARQ reference. A favors exchanges that reduce the mean squared residual of the highest-error 10% of patches; B samples uniformly from eligible exchanges; C favors exchanges that increase this statistic. Allocation uses document embeddings only; queries are introduced after each index is fixed.