

Argus-Retriever: Vision-LLM Late-Interaction Retrieval with Region-Aware Query-Conditioned MoE for Visual Document Retrieval

Abdelrahman Abdallah¹, Mahmoud Abdalla², Mohammed Ali¹, Adam Jatowt¹

¹University of Innsbruck, Austria ²Chungbuk National University, South Korea
{abdelrahman.abdallah, adam.jatowt}@uibk.ac.at

Abstract

Late-interaction vision-language retrievers represent each document page as many visual token embeddings and score queries with MaxSim. In systems such as ColPali, ColQwen, ColNomic, and Nemotron ColEmbed, the document embeddings are produced without seeing the query, so the same page is represented identically for a table lookup, a chart question, and a layout-sensitive evidence request. We introduce **Argus**, a family of query-conditioned late-interaction retrievers built on Qwen3.5-VL. Argus adds a region-aware Mixture-of-Experts module: the query encoder produces both retrieval embeddings and a compact context vector, the document page is pooled into spatial regions, and a query-aware router selects latent experts per region before MaxSim. The output remains a multi-vector index compatible with ColPali-style retrieval, but the document representation is now dependent on the query (i.e., $D(q)$). All Argus models use a 1024-dimensional retrieval head, compared with the 2560-dimensional and 4096-dimensional heads of recent state-of-the-art systems, and are trained on a source-balanced 594K-pair mixture of public supervision. The 9B model reaches **92.67** NDCG@5 on ViDoRe V1 and **86.0** NDCG@5 on the combined V1+V2 leaderboard, the highest reported value for an open late-interaction model on the combined leaderboard. Inside a Qwen3.6-27B agentic pipeline on ViDoRe V3, Argus-9B improves from 62.5 to **67.1** NDCG@10, so the same retriever also serves as a search primitive for LLM agents.¹

1 Introduction

Retrieval-Augmented Generation (RAG) has become a standard way to ground language models in external knowledge (Lewis et al., 2020; Abdallah

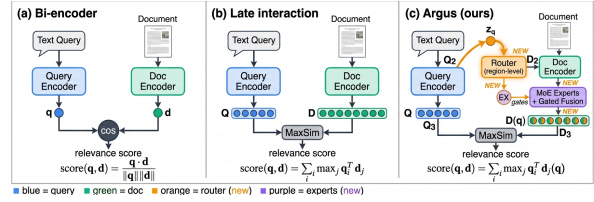


Figure 1: **Where does the query interact with the document?** Bi-encoders compare single vectors, while standard late-interaction models compare query tokens against fixed document tokens. Argus keeps MaxSim scoring, but adds a query-conditioned region router and MoE expert fusion.

et al., 2025). The first step in this pipeline is retrieval: given a user query, the system must find the pages or passages that contain the evidence needed by the generator. Many strong retrieval models have been developed for clean text (Karpukhin et al., 2020; Wu et al., 2024; Abdallah et al., 2026a; Zhan et al., 2021), but practical enterprise and scientific corpora often contain PDFs, presentation slides, scanned pages, financial reports, and technical manuals. In these documents, the answer may be stored in a table cell, a chart axis, a figure caption, or a formula. Pure text extraction can miss this information, and OCR-based pipelines often ignore the layout that makes the evidence interpretable (Huang et al., 2022).

Visual Document Retrieval (VDR) addresses this problem by retrieving rendered document pages directly from their images. Recent VLMs can recognize rendered text, layout, and visual structure from pixels, which has made them natural backbones for VDR (Faysse et al., 2025; Moreira et al., 2026). The strongest VDR systems usually combine a vision-language backbone with the late-interaction mechanism introduced by ColBERT (Khattab and Zaharia, 2020). Instead of compressing a page into a single vector, late-interaction models project a document image into many patch-level embeddings. A query is also represented as multiple to-

¹The code and data are available: <https://github.com/DataScienceUIBK/Argus-Retriever>

ken embeddings, and relevance is computed with MaxSim, which lets each query token match its best document token. This design preserves fine-grained evidence that bi-encoder pooling tends to discard.

Despite this progress, late-interaction VDR retrievers compute document embeddings without seeing the query (Faysse et al., 2025; ViDoRe Team, 2024; Team, Nomic; Moreira et al., 2026). Argus removes this restriction by adding a query-conditioned region router and a Mixture-of-Experts fusion inside the document encoder, so the final document representation becomes $\mathbf{D}(q)$ while remaining a multi-vector index compatible with MaxSim scoring (Figure 1; details in §3). This design is meant to recover part of the adaptivity of cross-encoders (Nogueira et al., 2019) while preserving the practical benefits of late interaction. Argus also targets a practical training and storage regime. Rather than scaling the retrieval head to several thousand dimensions or relying on model soup and post-training checkpoint merging (Wortsman et al., 2022), all Argus models use a 1024-dimensional late-interaction head and are trained in a single run on a source-balanced 594K-pair mixture of public data, with no seed averaging, checkpoint merging, or mined hard negatives. The claim is not that Argus wins by being data-efficient, but that it does not win by having more supervision: 594K pairs is smaller than or comparable to the pools used by the strongest baselines (§5.2).

Contributions. (1) We introduce **Argus**, a query-conditioned late-interaction VLM retriever. A region-aware router selects latent experts that produce a query-dependent multi-vector document representation, $\mathbf{D}(q)$, scored against the query tokens by MaxSim. (2) The router conditions on three signals — per-region content, 2D coordinates, and a pooled query context $\mathbf{z}_q$ — so the same page routes differently for different queries while the output stays a ColPali-compatible multi-vector index. (3) The design is deployable: the image encoder runs once per page offline; only the query encoder, router, gated fusion, projection head, and MaxSim run at query time against cached visual grids.

2 Related Work

2.1 Visual Document Retrieval

Early multimodal retrievers followed the Contrastive Language–Image Pre-training (CLIP)

recipe, encoding an image and a text query into global vectors and comparing them with a single similarity score (Radford et al., 2021; Abdallah et al., 2026b). That design is simple and fast, but it is a poor fit for pages whose evidence is spread across table cells, captions, figures, formulas, and layout.

Visual document retrieval moved the field closer to the document itself by retrieving rendered page images instead of relying only on OCR text. ColPali (Faysse et al., 2025) was the key step in this direction: it used a vision-language backbone to turn a page image into many visual token embeddings and scored them with late interaction. The same recipe has since been extended with stronger backbones and training recipes, including ColQwen, ColNomic (Team, Nomic), and Nemotron ColEmbed (Moreira et al., 2026). Modern VLM backbones make this possible because they already carry strong document understanding capabilities. PaliGemma (Beyer et al., 2024), Qwen2-VL (Wang et al., 2024), Qwen3-VL (Bai et al., 2025), and NVIDIA Eagle 2 (Li et al., 2025) can read rendered text and use layout cues directly from pixels. Argus builds on this visual retrieval line.

2.2 Late Interaction

The MaxSim operator (Khattab and Zaharia, 2020) defines relevance between a query $q = (q_1, \dots, q_m)$ and a document $d = (d_1, \dots, d_n)$ as

$$\text{MaxSim}(q, d) = \sum_{i=1}^m \max_{j=1, \dots, n} q_i^\top d_j. \quad (1)$$

For each query token, MaxSim selects its best-matching document token, and the resulting per-token scores are summed into a single relevance score. We use this scoring function throughout. Computing MaxSim for a single (query, document) pair requires $m \times n$ inner products, which reduce to a single matrix multiplication on GPU; the dominant cost of late interaction is therefore the storage of all per-token document embeddings, which we revisit in §6.4.

2.3 Contrastive Learning

Late-interaction retrievers are trained with the contrastive InfoNCE objective (Oord et al., 2018):

$$\mathcal{L}_{\text{NCE}} = -\log \frac{\exp(\text{sim}(q, d^+)/\tau)}{\sum_{d_i \in \{d^+\} \cup D_N} \exp(\text{sim}(q, d_i)/\tau)}, \quad (2)$$

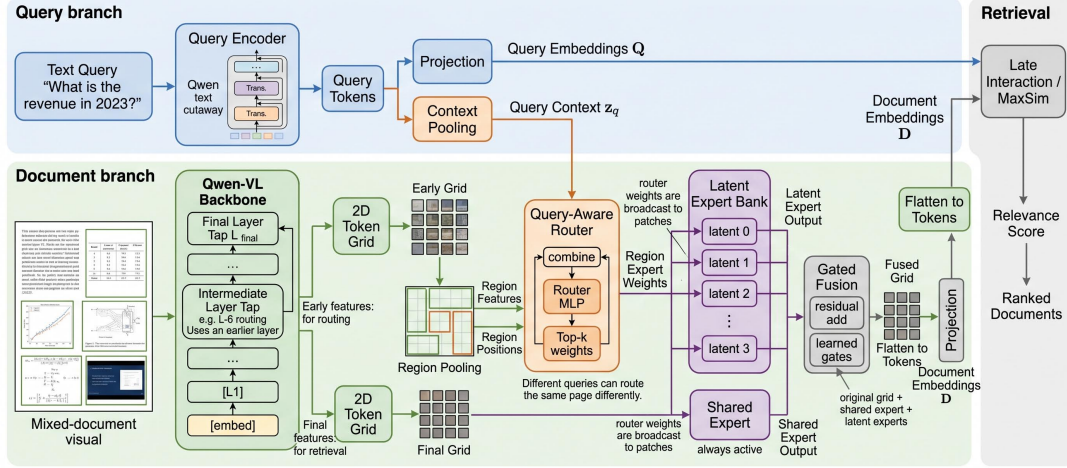


Figure 2: **Argus architecture.** The query branch yields retrieval embeddings $\mathbf{Q}$ and a pooled context $\mathbf{z}_q$ for routing. The document branch taps Qwen3.5-VL at two depths: $L-5$ feeds region pooling and the query-aware router; L feeds the latent expert bank and shared expert. Top- k router weights are broadcast to patches to mix expert outputs, and a gated residual fusion produces the query-conditioned grid $\mathbf{D}(q)$, scored against $\mathbf{Q}$ by MaxSim. At inference, the image encoder runs once per page; cached grids and region features are reused across queries (Appendix D).

where $\text{sim}(\cdot, \cdot)$ is MaxSim, τ is a temperature, and D_N is the negative set. The Argus training objective uses this loss with in-batch negatives only, as detailed in §4.1.

2.4 Mixture of Experts in Retrieval

Mixture-of-Experts layers use a router to send each input to a small subset of specialist networks (Shazeer et al., 2017; Fedus et al., 2022). This idea is common in language models because it increases capacity without activating every parameter on every example. In retrieval the idea has been explored more sparingly: Guo et al. (2025) apply a competitively-learned MoE to first-stage text retrieval; Nussbaum and Duderstadt (2025) train sparse-MoE text embedding models; and Sokli et al. (2025) study MoE in dense retrieval. To our knowledge none of these prior systems makes the routing decision depend on the *query* at the document side, and none target visual document retrieval, where the main pressure is not only model capacity but also indexing and query-time cost.

3 Argus: A Family of Region-Aware MoE Late-Interaction Models

3.1 Model Family

Argus is a family of three late-interaction VLM retrievers, all built on Qwen3.5-VL backbones, all using MaxSim scoring at embedding dimension $d = 1024$ (Table 1). The three sizes share the same query-conditioned region-aware MoE module

Table 1: The Argus family. All sizes share the same query-conditioned MoE ($K=4$ latent + 1 shared expert, top-2 routing, region $r=4$, router tap $L-5$) and a 1024-dim retrieval head.

Model	#Params	Dim	Experts
<i>Argus-2B</i>	2.32 B	1024	4 (top-2)
<i>Argus-4B</i>	4.71 B	1024	4 (top-2)
<i>Argus-9B</i>	8.82 B	1024	4 (top-2)

described in §3.2; only the backbone differs.

3.2 Architecture Overview

Figure 2 shows the full training-time architecture. Argus reads a text query and a document image through a shared Qwen3.5-VL backbone, taps the backbone at two depths to obtain a routing-friendly intermediate visual grid and a retrieval-friendly final visual grid, and inserts a query-conditioned region-aware MoE module between the final layer and the projection head. Figure 2 shows the training-time forward pass; at inference, the document side is cached once per page and only the query-side path runs per query, so the query-conditioned design remains compatible with offline indexing (Appendix D).

3.3 Query Branch: $\mathbf{Q}$ and $\mathbf{z}_q$

The text query is encoded by the shared backbone and projected to retrieval dimension $d=1024$ to give per-token embeddings $\mathbf{Q} \in \mathbb{R}^{m \times d}$ used in MaxSim. We additionally pool $\mathbf{Q}$ with the attention

mask and L2 normalize to obtain a single query context vector

$$\mathbf{z}_q = \frac{\sum_i a_i \mathbf{Q}_i}{\|\sum_i a_i \mathbf{Q}_i\|_2} \in \mathbb{R}^d, \quad (3)$$

where $a_i \in \{0, 1\}$ is the query attention mask. $\mathbf{z}_q$ is *not* used directly in MaxSim. It is sent into the document branch to condition the router, so the same $\mathbf{Q}$ that scores documents also shapes how those documents are encoded.

3.4 Document Branch: Region Pooling at Two Depths

We extract two hidden states from the document forward pass: $\mathbf{H}_{\text{router}} = \mathbf{H}^{(L-5)}$ for routing and $\mathbf{H}_{\text{final}} = \mathbf{H}^{(L)}$ for retrieval. The intermediate tap preserves layout and visual structure useful for region-level decisions; the final tap carries the strongest semantic features for retrieval. This separation is consistent with probing studies showing that structural and positional information peaks in middle-to-late transformer layers, before the final layers specialize for the task head (Tenney et al., 2019; Karamcheti et al., 2024), and with the standard VLM practice of taking vision features from a layer *before* the final one (Liu et al., 2023). After selecting image-token positions, we reshape both to 2D grids

$$\mathbf{G}_{\text{early}}, \mathbf{G}_{\text{final}} \in \mathbb{R}^{H \times W \times h}. \quad (4)$$

We pad each grid to a multiple of the region size $r=4$ and average-pool valid patches within each $r \times r$ block to obtain N region tokens $\mathbf{R} \in \mathbb{R}^{N \times h}$ from $\mathbf{G}_{\text{early}}$, a region validity mask, and normalized box coordinates $\mathbf{c} \in \mathbb{R}^{N \times 4}$ of the form $[x_0, y_0, x_1, y_1]$ for each region.

3.5 Query-Aware Router

For each region r , the router input is the sum of three projected signals: region content, region position, and query context, all lifted to the backbone hidden dimension h :

$$\mathbf{u}_r = \mathbf{R}_r + \mathbf{W}_{\text{coord}} \mathbf{c}_r + \mathbf{W}_{\text{query}} \mathbf{z}_q. \quad (5)$$

where $\mathbf{W}_{\text{coord}} \in \mathbb{R}^{h \times 4}$ and $\mathbf{W}_{\text{query}} \in \mathbb{R}^{h \times d}$ are learned linear projections (no bias) that lift the 4-dimensional region coordinates $\mathbf{c}_r$ and the d -dimensional pooled query context $\mathbf{z}_q$ into the backbone hidden dimension h . Addition (rather than concatenation) keeps the router input dimension fixed at h and acts as a positional and query bias

on the region content, in the same spirit as additive positional embeddings in Transformers. A two-layer MLP $\mathcal{R} : \mathbb{R}^h \rightarrow \mathbb{R}^K$ produces routing logits over $K=4$ latent experts. We set $K=4$ to roughly match the recurring content types found in visual documents – running text, tables, charts, and figures/formulas – giving the router enough capacity to send different regions to different specialists while keeping the expert bank small enough to receive sufficient gradient signal under our balanced training mixture. Top- k sparse softmax with temperature τ_R converts these to expert weights:

$$\mathbf{g}_r = \text{softmax}_k \left(\frac{\mathcal{R}(\mathbf{u}_r) + \epsilon}{\tau_R} \right), \quad (6)$$

where $\tau_R = 1$ throughout training and inference, $\epsilon \sim \mathcal{N}(0, \sigma_R^2)$ with $\sigma_R = 0.1$ during training and $\epsilon = 0$ at inference, and only the top- k logits are kept (the rest are masked to $-\infty$ before the softmax). The same $\mathbf{z}_q$ is broadcast to every region of a given page, but each region’s content $\mathbf{R}_r$ and position $\mathbf{c}_r$ differ, so different regions receive different routing decisions. The same region also receives different routing for different queries, which is the property that makes Argus query-conditioned.

3.6 Latent Expert Bank and Shared Expert

The latent expert bank consists of $K=4$ small position-wise feed-forward modules of the form $\text{LN} \rightarrow \text{Linear}(h, 2h) \rightarrow \text{GELU} \rightarrow \text{Linear}(2h, h)$, each applied to $\mathbf{G}_{\text{final}}$ to produce K candidate grids $\mathbf{E}_k \in \mathbb{R}^{H \times W \times h}$. A separate *shared expert* (LN, expansion factor 4, GELU, down-projection; always active) provides a stable dense fallback path. Region expert weights $\mathbf{g}_r$ are broadcast from each region to its constituent patches, and the routed specialist output is the per-patch weighted sum

$$\mathbf{S}_{ij} = \sum_{k=1}^K g_{r(i,j),k} [\mathbf{E}_k(\mathbf{G}_{\text{final}})]_{ij}. \quad (7)$$

Routing is computed at the region level for stability and cost; expert mixing is applied at the patch level so that MaxSim retains its fine-grained evidence.

3.7 Gated Residual Fusion

The final grid is updated residually under learned scalar gates:

$$\mathbf{F} = \mathbf{G}_{\text{final}} + \sigma(\alpha) \mathbf{S}_{\text{shared}} + \sigma(\beta) \mathbf{S}. \quad (8)$$

The two scalars are jointly learned with the rest of the model and govern how strongly the dense

Table 2: Retrieval accuracy (NDCG@5) on the ViDoRe V1+V2 leaderboard. Argus rows are shaded; per-column best values across listed models are bolded. Baselines listed in §4.3.

Model	ViDoRe V1										ViDoRe V2				Average		
	ArXiv	DocVQA	InfoVQA	Shift	AI	Ene	Gov	Hlt	TabF	TatD	BMed	ESG	ESGHL	Econ	V1	V2	Avg
ColPali v1.3	83.3	58.4	85.5	77.3	97.4	94.6	96.1	97.4	86.7	70.8	54.6	54.9	58.5	48.6	84.8	54.2	76.0
ColSmol-500M	74.9	58.0	87.2	66.8	98.4	94.6	94.7	98.0	74.7	77.5	44.9	40.4	52.4	36.4	82.5	43.5	71.4
ColQwen2 v1.0	88.0	61.5	92.5	89.9	99.0	95.9	95.5	98.8	89.0	82.2	56.3	52.5	60.4	50.6	89.2	55.0	79.4
ColQwen2.5	88.9	63.6	92.5	87.9	99.6	96.1	95.8	98.0	90.8	82.1	59.2	58.3	66.4	53.3	89.5	59.3	80.9
ColNomic-3b	88.1	61.3	92.8	90.1	96.3	97.4	96.6	98.3	94.4	83.2	62.5	49.2	57.0	53.3	89.9	55.5	80.0
ColNomic-7b	88.3	60.1	92.2	89.3	98.8	96.3	95.9	99.3	96.0	81.1	63.4	54.8	68.7	56.2	89.7	60.8	81.5
Sauerkraut-4b	91.8	67.0	94.2	90.5	99.6	96.5	96.2	100.0	89.5	82.7	60.0	58.8	69.2	56.0	90.8	61.0	82.3
Sauerkraut-8b	93.8	64.7	94.5	90.4	98.6	96.5	96.8	99.3	92.2	84.0	64.9	58.4	70.8	57.6	91.1	62.9	83.0
Nemotron-colembed-3b-v1	88.4	66.2	94.9	90.7	99.6	96.6	97.8	99.3	95.9	80.6	64.3	58.6	75.4	55.9	91.0	63.5	83.2
Nemotron-colembed-4b-v2	92.0	67.4	93.3	92.3	99.3	96.2	98.0	98.5	98.1	81.2	64.0	62.4	71.4	57.7	91.6	63.9	83.7
Ops-Colqwen3-4B	91.8	66.5	94.0	90.8	99.6	97.3	98.0	99.6	93.6	82.4	64.9	65.6	78.6	61.9	91.4	67.8	84.6
Nemotron-colembed-8b-v2	93.1	68.1	94.6	93.3	100.0	97.9	98.9	99.6	97.7	83.4	67.0	59.0	73.2	60.4	92.7	64.9	84.7
Argus-2B	90.3	67.5	95.0	91.3	99.6	97.3	97.3	99.3	93.4	84.0	65.0	59.9	69.4	51.9	91.5	61.5	82.9
Argus-4B	91.0	67.7	94.6	94.7	99.6	97.9	97.8	99.6	95.3	84.8	64.4	62.2	69.9	59.8	92.3	64.1	84.2
Argus-9B	92.3	68.1	94.3	93.6	99.6	97.3	98.9	99.6	97.5	85.5	66.2	67.6	79.1	63.8	92.7	69.2	86.0

fallback and the routed specialists are mixed into the original document features. Initializing them so that $\sigma(\alpha) = \sigma(\beta) = 0.5$ keeps Argus close to its dense ColQwen3.5 baseline at the start of training. The fused tokens are written back into the image-token positions of $\mathbf{H}_{\text{final}}$, and the standard projection head produces the final document embeddings $\mathbf{D}(q) \in \mathbb{R}^{n \times d}$, which are L2-normalized and scored against $\mathbf{Q}$ by MaxSim.

4 Experimental Setup

4.1 Training Objective

The joint training phase combines two loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{NCE}} + \lambda_{\text{bal}} \mathcal{L}_{\text{bal}}. \quad (9)$$

Retrieval (NCE). The retrieval term is the contrastive InfoNCE loss of §2.3 with $\text{sim}(\cdot, \cdot)$ set to MaxSim. For each positive pair (q, d^+) , the negative set D_N contains only the *other documents in the same batch*. We do not mine explicit hard negatives.

Load balance. To keep the latent expert bank well used, we adopt the load-balance loss of Fedus et al. (2022). Let m_k be the average softmax routing probability of expert k over the valid pooled regions in a batch and f_k be the fraction of those regions whose top-1 routing assignment is expert k . The balance loss is

$$\mathcal{L}_{\text{bal}} = K \sum_{k=1}^K m_k f_k. \quad (10)$$

This term is minimized when both the soft and hard routing distributions are uniform over the $K=4$ latent experts, which discourages collapse to a single

specialist while remaining compatible with sparse top- k selection.

Warmup-only routing supervision. The router warmup phase (§4.2, “Router warmup”) adds a small cross-entropy term $\mathcal{L}_{\text{rt}}$ on the routing logits, weighted by λ_{rt} , to encourage every latent expert to be exercised before retrieval gradients are introduced. The retrieval term $\mathcal{L}_{\text{NCE}}$ is disabled during warmup, and $\mathcal{L}_{\text{rt}}$ is set to zero again in the joint phase.

4.2 Training Data and Hyperparameters

All three Argus sizes initialize from Qwen3.5-VL- $\{2B, 4B, 9B\}$ -Instruct and are trained with PEFT/LoRA (Hu et al., 2022) of rank 32 on $4 \times \text{H100 } 80 \text{ GB}$. The training mixture contains 593,677 pairs drawn from ViDoRe ColPali, VDR multilingual, VisRAG, TabFQuAD, TatDQA, and a capped Docmatix-IR subset. We use $K=4$ latent experts with top-2 routing, region size $r=4$, and a routing tap at backbone layer $L-5$. Source-level data counts and full optimizer hyperparameters are in Appendix A and Appendix B.

Router warmup. A routed MoE module is prone to collapse if it sees retrieval gradients before learning to use all experts, so we first run a short warmup phase. The MoE module is attached to the backbone with the fusion gates frozen near zero, PEFT weights frozen, and the router and experts trained only against the routing and balance auxiliary losses. This phase has no retrieval signal: its only job is to make the router use all K experts before the retrieval objective is added.

Table 3: Retrieval accuracy (NDCG@10) on the eight public ViDoRe V3 tasks. Baselines listed in §4.3.

Model	CompSci	Energy	FinanceEn	FinanceFr	HR	Industrial	Pharma	Physics	Avg
Nemotron-colembed-8b-v2	79.30	69.82	67.19	51.54	66.32	56.03	67.19	50.84	63.53
Tomoro-colqwen3-embed-8b	75.35	68.41	65.08	49.10	63.98	54.41	66.36	50.13	61.60
Nemotron-colembed-4b-v2	78.56	67.48	65.02	49.01	62.39	56.10	64.86	52.78	62.02
Ops-Colqwen3-4B	77.74	66.49	65.71	48.81	61.81	53.99	66.42	49.14	61.26
Tomoro-colqwen3-embed-4b	75.44	66.43	63.84	46.83	60.09	53.58	65.74	49.32	60.16
Llama-Nemotron-colembed-3b-v2	77.09	64.88	64.23	44.41	62.28	51.71	66.04	46.93	59.70
Jina-embeddings-v4	71.81	63.50	59.30	46.10	59.53	50.38	63.09	46.63	57.54
ColNomic-embed-multimodal-7b	76.20	63.58	56.57	45.46	58.67	50.13	62.26	48.25	57.64
Llama-Nemoretriever-colembed-3b-v1	75.16	62.07	60.88	43.77	58.69	47.09	63.74	45.13	57.07
<i>Argus-2B</i>	75.83	65.82	64.27	47.62	62.33	50.76	64.86	49.27	60.09
<i>Argus-4B</i>	76.97	68.58	63.54	51.43	65.64	54.47	66.08	50.06	62.09
<i>Argus-9B</i>	78.03	69.59	64.74	50.90	65.46	53.45	67.29	50.51	62.50

4.3 Baselines

We compare Argus against the strongest open late-interaction VDR systems available on the public MTEB leaderboards: ColPali (Faysse et al., 2025), ColSmol-500M (ViDoRe Team, 2025b), ColQwen2 (ViDoRe Team, 2024), ColQwen2.5 (ViDoRe Team, 2025a), ColNomic-3b/7b (Nomic AI, 2025a,b), Sauerkraut-4b/8b (VAGO Solutions, 2025a,b), Nemotron-colembed (NVIDIA, 2025, 2026a,b), Ops-Colqwen3-4B (Alibaba Cloud OpenSearch-AI, 2026), Tomoro-colqwen3-embed-4b/8b (Tomoro AI, 2026a,b), ColNomic-embed-multimodal-7b (Nomic AI, 2025b), and Jina-embeddings-v4 (Günther et al., 2025). For MIRACL-Vision we additionally compare against the baselines published with that benchmark (Osmulski et al., 2025).

5 Main Results

We evaluate Argus on three retrieval benchmarks and one agentic-retrieval setting. The standalone leaderboards are the ViDoRe V1+V2 mix (§5.1), the more recent V3 mix (§5.3), and the multilingual MIRACL-Vision benchmark (§5.5). On V3 we additionally compare against the agentic version of the same retriever (§5.4).

5.1 ViDoRe V1+V2 Leaderboard

Following the convention used by the official MTEB ViDoRe V1+V2 leaderboard (ViDoRe, 2026) and prior work (Moreira et al., 2026), we report **NDCG@5** on V1 (10 in-domain tasks) and V2 (4 out-of-domain tasks) in Table 2. *Argus-9B* attains the highest overall average (86.0), beating *nemotron-colembed-vl-8b-v2* (84.7) by **+1.3** and *Ops-Colqwen3-4B* (84.6) by **+1.4**. The gap is largest on the harder out-of-domain V2 split (+4.3 over Nemotron-8b-v2), and *Argus-9B* is the best system on three of the four V2 tasks — ESGReports

(67.6), ESGReports-HighLevel (79.1), and EconomicsReports (63.8) — and second on BiomedicalLectures (66.2 vs. 67.0). This supports the hypothesis that adapting the document representation to the query helps most when documents are out-of-distribution. *Argus-4B* (84.2) is competitive with the 4B SOTA *Ops-Colqwen3-4B* (84.6) and outperforms *Nemotron-4b-v2* (83.7) and *Sauerkraut-4b* (82.3). *Argus-2B* (82.9) is competitive with several much larger 7B and 8B baselines, including *ColNomic-7b* (81.5), and approaches *Sauerkraut-8b* (83.0) at less than a third of the parameter count. Argus reaches these scores with a 1024-dimensional retrieval head and the balanced training mixture of Appendix A, narrower and smaller than the head dimensions and training pools used by the strongest baselines (Appendix C), and without seed averaging or post-training checkpoint merging.

5.2 Controlling for the Backbone

Argus is built on Qwen3.5-VL while most published baselines use earlier backbones, so we separate the method from the backbone in both directions (Table 5). First, several public dense retrievers already use the same Qwen3.5-VL backbones; Argus-9B leads that group outright (86.0 vs. 85.3) as a single training run with in-batch negatives and public data only, while every competitor at that scale uses model merging, mined hard negatives, or proprietary supervision. Second, we retrained Argus-4B on the previous-generation Qwen3-VL-4B backbone with an identical recipe: it still exceeds the strongest same-size Qwen3-VL system (84.7 vs. 84.6, with a $2.5\times$ narrower head) and matches the 8B Nemotron model. Argus-4B on the older backbone (84.7) is in fact no worse than Argus-4B on Qwen3.5 (84.2), so the backbone upgrade does not account for the gains. Together

Table 4: MIRACL-Vision results (NDCG@10) on multilingual visual document retrieval.

Language	dse-qwen2 -2b-mrl-v1	gme-Qwen2 -VL-2B	vdr-2b -multi-v1	colqwen2 -v1.0	llama-nemo -3b-v1	nemo-cb -3b-v2	nemo-vl -4b-v2	nemo-vl -8b-v2	Argus-9B
German	0.6267	0.6345	0.6205	0.5995	0.6831	0.6924	0.7100	0.7233	0.7236
English	0.6605	0.6784	0.6784	0.6417	0.7363	0.7397	0.7246	0.7480	0.7149
Spanish	0.5927	0.6277	0.6274	0.6224	0.7109	0.7236	0.7033	0.7089	0.7307
Finnish	0.4162	0.6863	0.5283	0.6604	0.8513	0.8541	0.8398	0.8726	0.8519
French	0.7160	0.6851	0.7194	0.6876	0.7988	0.7943	0.7943	0.8171	0.8301
Indonesian	0.4866	0.5416	0.5254	0.5320	0.6428	0.6550	0.6480	0.6680	0.6125
Japanese	0.6232	0.7305	0.6553	0.6970	0.7260	0.7493	0.8326	0.8690	0.7750
Russian	0.6505	0.7202	0.6995	0.6811	0.7670	0.7920	0.7879	0.8399	0.8184
Yoruba	0.4178	0.4884	0.4577	0.5120	0.5888	0.5943	0.4469	0.5252	0.8099
Chinese	0.5962	0.6314	0.5963	0.4926	0.4355	0.4878	0.6697	0.7204	0.6851
Average	0.5786	0.6424	0.6108	0.6126	0.6941	0.7083	0.7157	0.7492	0.7552

Table 5: Backbone controls (NDCG@5, ViDoRe V1+V2). Top: public retrievers sharing the Qwen3.5-VL backbone. Bottom: Argus-4B retrained on the previous-generation Qwen3-VL backbone with the identical recipe. n/r: not reported on the model card.

Model	Post-training	V1	V2	Avg
<i>Qwen3.5-VL backbone</i>				
ColQwen3.5-4.5B-v3	seed avg + merge	91.6	64.3	83.7
VultronRetrieverCore-4.5B	merge + hard neg.	92.2	66.1	84.8
webAI-ColVec1-9b	none reported	91.4	n/r	n/r
VultronRetrieverPrime-8B	merge + hard neg.	92.1	68.2	85.3
Argus-4B	none	92.3	64.1	84.2
Argus-9B	none	92.7	69.2	86.0
<i>Qwen3-VL backbone</i>				
Ops-Colqwen3-4B	none	91.4	67.8	84.6
Nemotron-colembed-vl-8b-v2	merge	92.7	64.9	84.7
Argus-4B (Qwen3-VL)	none	92.2	66.1	84.7

with the fixed-backbone ablations of §6.2 and Appendix J, this isolates the architecture contribution from both directions.

5.3 ViDoRe V3 Leaderboard

Table 3 reports the ViDoRe V3 leaderboard restricted to the eight public domains. The Argus models at 2B, 4B, and 9B scale are listed alongside the strongest published systems. Argus-9B (62.50) is competitive with the best 8B baseline, Nemotron-colembed-8b-v2 (63.53), at a 1024-dimensional retrieval head versus the 4096-dimensional head used by the latter.

5.4 Agentic ViDoRe V3 Retrieval

We additionally evaluate Argus as the retriever inside an agentic search pipeline. The agent uses a vLLM-served Qwen3.6-27B model and exposes three tools (think, retrieve, final-results) that let it rewrite or decompose the query, call the late-interaction retriever multiple times, and return a final ranked list of document ids. The reported metric is NDCG@10 over the final ranked list; no

Table 6: Matched agentic control on the eight public ViDoRe V3 tasks (NDCG@10): identical Qwen3.6-27B agent, prompts, tool set, candidate budget, final-list size, and fallback logic for both retrievers.

Retriever	Standalone	+ agent	Δ
Nemotron-colembed-8b-v2	63.5	66.6	+3.1
Argus-9B	62.5	67.1	+4.6

generated answer is scored. We set the target list size to 10 and the retriever candidate budget to 100. Full prompts, model serving details, and fallback logic are in Appendix I. Figure 3 compares the same Argus-9B retriever before and after wrapping it in the Qwen3.6 agent on the eight public ViDoRe V3 tasks. The agent improves the macro average from 62.50 to 67.10 NDCG@10.

Because an agent lifts any retriever, we ran the same loop with Nemotron-colembed-8b-v2 under identical prompts, budgets, and fallback logic (Table 6). Both improve, so much of the lift belongs to the reformulation loop rather than to Argus. The retriever is nonetheless not interchangeable: under the identical agent Argus-9B overtakes Nemotron-8b-v2 (67.1 vs. 66.6) despite starting one point behind standalone, and gains more from the loop (+4.6 vs. +3.1). This is consistent with the design — $\mathbf{D}(q)$ is re-conditioned on every reformulated query, so agentic query diversity translates into document-side adaptation — and Argus’s $2\times$ lower per-query latency matters more in this setting, since the agent calls the retriever many times per question.

5.5 Multilingual MIRACL-Vision

Table 4 reports NDCG@10 on MIRACL-Vision, alongside the baselines published with that benchmark. Argus reaches the highest macro average

across the ten languages (0.7552 versus 0.7492 for Nemotron-colembed-8b-v2) and is the best system on five of the ten languages, including the long-tail Yoruba split (0.8099 versus 0.5252 for Nemotron-8b-v2). This complements the V1+V2 picture by showing that the gains from query-conditioned routing carry over to a different multilingual evaluation distribution.

6 Analysis

The previous section established the headline retrieval numbers. This section unpacks them along three axes: how the gain distributes across the 2B/4B/9B family (§6.1), which design choices of the query-conditioned MoE module carry the gain (§6.2), and what the learned router actually does (§6.3).

6.1 Scaling Across the Argus Family

Argus improves from 82.9 to 86.0 average NDCG@5 as the backbone grows from 2B to 9B (Table 2). The main change is on ViDoRe V2: the 4B to 9B step adds +5.1 points on V2, while V1 changes only modestly. Larger backbones therefore help the query-conditioned router most under domain shift, and the V1 ceiling is approximately reached already at the 4B scale. A visualization of this trend is given in Appendix H (Figure H.1).

6.2 Which Components Carry the Gain?

Table 7 ablates the three signals that the query-aware router combines: the query context $\mathbf{z}_q$, the region position $\mathbf{c}_r$, and the always-active shared expert. All ablations are run on the Argus-4B configuration and reported as NDCG@5 on the V1 and V2 splits. Each ablated variant is retrained from the same Qwen3.5-VL-4B initialization, with the same training data mixture (§A), the same hyperparameters (§B), and the same two-phase router-warmup-then-joint schedule as the released Argus-4B model; only the listed component is removed before training begins. Removing any of the three components drops V1+V2 average by roughly 3 to 4 points, with the largest losses on the out-of-domain V2 split, confirming that all three components contribute materially to performance. The hypothesis behind each ablation row is given in Appendix F.

6.3 What Does the Router Learn?

The router is explicitly conditioned on the query context $\mathbf{z}_q$, so the same page can receive different

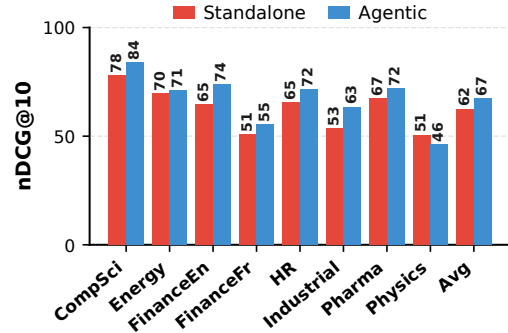


Figure 3: Standalone versus agentic Argus retrieval on the public ViDoRe V3 tasks.

Table 7: Argus-4B component ablation (NDCG@5). Each row retrains the full model with the listed component removed.

Configuration	V1	V2
Argus-4B (full)	92.3	64.1
– query conditioning ($\mathbf{z}_q$)	88.9	59.5
– region position ($\mathbf{c}_r$)	88.8	59.4
– shared expert ($\sigma(\alpha) = 0$)	89.7	59.1

sparse routing patterns for different queries. On a mixed-content ViDoRe V2 Economics page, the released Argus-4B reroutes 24 of 63 pooled regions to a different top-1 expert when the query changes; a side-by-side visualization of this example is given in Appendix G (Figure G.1). We treat this as qualitative evidence only; the controlled test of query conditioning is the ablation in Table 7, where $\mathbf{z}_q$ is removed while the rest of the model is kept fixed.

Aggregate top-2 utilization across the same ViDoRe V2 Economics sample is shown in Figure 4; all four latent experts remain active at the corpus level. The dashed line marks the uniform $1/K$ reference. The measured shares are (0.17, 0.32, 0.31, 0.20), with entropy 1.36 and 3.88 effective experts. This aggregate view complements the single-query sharp routing pattern in Figure G.1. We additionally sweep the expert-bank size and the router tap depth in Appendix J and Appendix K. Both ablations show an inverted-U with a peak at the released configuration ($K=4$ experts, router tap $L=5$), and both confirm that V2 is the more sensitive split.

6.4 Index Storage

Late-interaction retrievers must store *all* per-token document embeddings. Per-page index size therefore scales linearly with both the number of visual

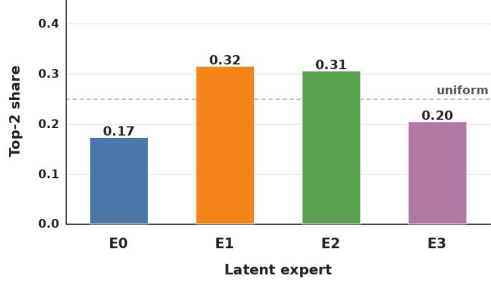


Figure 4: **Expert utilization on ViDoRe V2.** Average share of regions that select each latent expert in their top-2 set.

tokens and the embedding dimension. We compute storage as

$$\text{MB/page} = \frac{\text{tokens/page} \times d \times 2}{10^6},$$

assuming bf16 token embeddings. Argus deliberately keeps the embedding dimension fixed at $d=1024$ for all released sizes. This is narrower than the 2560-dimensional heads used by Ops-Colqwen3-4B and Nemotron-4B-v2, and the 4096-dimensional head used by Nemotron-8B-v2. Since token budgets differ across systems, Table 8 reports both axes explicitly rather than comparing dimensions alone. Two footprints must be distinguished.

The projected index ($T \times d$) is the quantity comparable across late-interaction systems, and Argus keeps it narrow at $d=1024$. Under query conditioning, however, $\mathbf{D}(q)$ is transient: it is materialized per query from the cache and never stored. What Argus stores per page is the working cache of backbone-dimension features, which is larger than the projected index and grows with the backbone hidden size. Under this stricter accounting, Argus-9B stores 17.8 MB/page against 18.9 MB/page for the index alone of Nemotron-colembled-8b-v2, and Argus-4B stores 11.2 against 11.8 for Nemotron-colembled-4b-v2 (Table 8). The resulting trade-off is therefore not a storage reduction but a storage *parity*: at comparable per-page footprint, Argus reaches a higher V1+V2 average (86.0 vs. 84.7) and a $2.0\times$ lower measured online latency (135.8 vs. 277.6 ms per query under the identical protocol, Appendix D). The stored column adds the backbone-dimension cache the router and experts consume: $(T + N) \times h$ values in bf16 with $N=128$ pooled regions, giving 8.9, 11.2, and 17.8 MB per page at $h=2048$, 2560, and 4096. Figure D.2 in Appendix D plots this trade-off.

Table 8: Per-page storage in bf16. *Projected index* is the multi-vector footprint at the retrieval-head dimension ($T \times d$). For query-independent retrievers this tensor *is* the stored index; for Argus it is produced per query from the cache and never written to storage, so the stored artifact is the *working cache* (final-layer grid, pooled region descriptors, region coordinates, stage-1 single vector).

Model	Tok/page	Dim	Projected (MB)	Stored (MB)
Tomoro-colqwen3-4b	1280	320	0.8	0.8
Ops-Colqwen3-4B	1280	2560	6.6	6.6
Nemotron-colembled-4b-v2	2304	2560	11.8	11.8
Nemotron-colembled-8b-v2	2304	4096	18.9	18.9
Argus-2B	2048	1024	4.2	8.9
Argus-4B	2048	1024	4.2	11.2
Argus-9B	2048	1024	4.2	17.8

7 Conclusion

Argus introduces a query-conditioned region-aware MoE module into the late-interaction VLM retrieval pipeline, allowing the same document page to be processed differently for different queries while preserving all of the retrieval-side properties of ColPali-style multi-vector indexing. Across the Argus family (2B/4B/9B), the 9B model achieves a new state of the art on ViDoRe V1+V2 among open late-interaction models, with the largest gains on out-of-domain V2 tasks, at an index footprint comparable to the strongest baseline and $2\times$ lower query latency.

Limitations

Argus inherits the per-query latency and per-page storage overheads of late interaction, and query conditioning changes what is stored rather than whether an index exists. The image encoder still runs once per page offline; what cannot be precomputed is the final projected embedding $\mathbf{D}(q)$, so the stored artifact is a backbone-dimension working cache (17.8 MB/page at 9B) rather than the narrower projected index (Table 8). All reported results use single-stage scoring over this cached index; beyond benchmark scale, per-query work is capped by scoring a candidate set (Appendix D), which is standard practice for every late-interaction retriever rather than an Argus-specific concession.

Our V3 evaluation covers only the eight public domains; the private Nuclear and Telecom splits are held out by the MTEB maintainers. The agentic results use a single Qwen3.6-27B agent at fixed search budgets; results may differ with other backbones or budgets, and, as Table 6 shows, a sub-

stantial part of the agentic lift is attributable to the reformulation loop itself rather than to the retriever. We did not sweep training mixtures, since each variant requires a full two-phase retraining; the mixture’s contribution is bounded only indirectly, by systems trained on comparable public pools (§5.2). Finally, the latent experts are not constrained to interpretable categories such as “table” or “chart”. Specialization is implicit and discovered during training, which limits the interpretability of individual routing decisions.

Acknowledgments

The authors would like to acknowledge the financial support provided by the Austrian Research Agency (FFG) for the project “AI Enabled Sustainability Jurisdiction Demonstrator” (project No. 915229). The computational results presented in this work have been achieved using the MUSICA cluster, part of the Austrian Scientific Computing (ASC) infrastructure.

References

- Abdelrahman Abdallah, Jamie Holdcroft, Mohammed Ali, and Adam Jatowt. 2026a. Are llm-based retrievers worth their cost? an empirical study of efficiency, robustness, and reasoning overhead. *arXiv preprint arXiv:2604.03676*.
- Abdelrahman Abdallah, Mohamed Darwish Mounis, Mahmoud Abdalla, Mahmoud SalahEldin Kasem, Mostafa Farouk Senussi, Mohamed Mahmoud, Mohammed Ali, Adam Jatowt, and Hyun Soo Kang. 2026b. Mm-bright: A multi-task multimodal benchmark for reasoning-intensive retrieval. pages 8604–8612.
- Abdelrahman Abdallah, Jamshid Mozafari, Bhawna Piryani, and Adam Jatowt. 2025. Dear: Dual-stage document reranking with reasoning agents via llm distillation. *arXiv preprint arXiv:2508.16998*.
- Alibaba Cloud OpenSearch-AI. 2026. Ops-Colqwen3-4B. <https://huggingface.co/OpenSearch-AI/Ops-Colqwen3-4B>. Hugging Face model card. Accessed 2026-05-18.
- athrael soju. 2026. colqwen3.5-4.5B-v3. <https://huggingface.co/athrael-soju/colqwen3.5-4.5B-v3>. Hugging Face model card. Accessed 2026-05-18.
- Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, and 1 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.
- Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, and 1 others. 2024. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*.
- Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. In *International Conference on Learning Representations*, volume 2025, pages 61424–61449.
- William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.
- Jiafeng Guo, Yinqiong Cai, Keping Bi, Yixing Fan, Wei Chen, Ruqing Zhang, and Xueqi Cheng. 2025. Came: Competitively learning a mixture-of-experts model for first-stage retrieval. *ACM Transactions on Information Systems*, 43(2):1–25.
- Michael Günther, Saba Sturua, Mohammad Kalim Akram, Isabelle Mohr, Andrei Ungureanu, Sedigheh Eslami, Scott Martens, Bo Wang, Nan Wang, and Han Xiao. 2025. [jina-embeddings-v4: Universal embeddings for multimodal multilingual retrieval](#). *Preprint*, arXiv:2506.18902.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models](#).
- Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM international conference on multimedia*, pages 4083–4091.
- Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. 2024. Prismatic vlms: Investigating the design space of visually-conditioned language models.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.
- Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient

- memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.
- Zhiqi Li, Guo Chen, Shilong Liu, Shihao Wang, Vibashan VS, Yishen Ji, Shiyi Lan, Hao Zhang, Yilin Zhao, Subhashree Radhakrishnan, and 1 others. 2025. Eagle 2: Building post-training data strategies from scratch for frontier vision-language models. *arXiv preprint arXiv:2501.14818*.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning.
- Gabriel de Souza P Moreira, Ronay Ak, Mengyao Xu, Oliver Holworthy, Benedikt Schifferer, Zhiding Yu, Yauhen Babakhin, Radek Osmulski, Jiarui Cai, Ryan Chesler, and 1 others. 2026. Nemotron colembed v2: Top-performing late interaction embedding models for visual document retrieval. *arXiv preprint arXiv:2602.03992*.
- Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.
- Nomic AI. 2025a. colnomic-embed-multimodal-3b. <https://huggingface.co/nomic-ai/colnomic-embed-multimodal-3b>. Hugging Face model card. Accessed 2026-05-18.
- Nomic AI. 2025b. colnomic-embed-multimodal-7b. <https://huggingface.co/nomic-ai/colnomic-embed-multimodal-7b>. Hugging Face model card. Accessed 2026-05-18.
- Zach Nussbaum and Brandon Duderstadt. 2025. Training sparse mixture of experts text embedding models. *arXiv preprint arXiv:2502.07972*.
- NVIDIA. 2025. llama-nemoretriever-colembed-3b-v1. <https://huggingface.co/nvidia/llama-nemoretriever-colembed-3b-v1>. Hugging Face model card. Accessed 2026-05-18.
- NVIDIA. 2026a. nemotron-colembed-v1-4b-v2. <https://huggingface.co/nvidia/nemotron-colembed-v1-4b-v2>. Hugging Face model card. Accessed 2026-05-18.
- NVIDIA. 2026b. nemotron-colembed-v1-8b-v2. <https://huggingface.co/nvidia/nemotron-colembed-v1-8b-v2>. Hugging Face model card. Accessed 2026-05-18.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.
- Radek Osmulski, Gabriel de Souza P Moreira, Ronay Ak, Mengyao Xu, Benedikt Schifferer, and Even Oldridge. 2025. Miracl-vision: A large, multilingual, visual document retrieval benchmark. *arXiv preprint arXiv:2505.11651*.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.
- Effrosyni Sokli, Georgios Peikos, Pranav Kasela, and Gabriella Pasi. 2025. *Leveraging cognitive complexity of texts for contextualization in dense retrieval*. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 27083–27096, Suzhou, China. Association for Computational Linguistics.
- Team, Nomic. Nomic embed multimodal: Interleaved text, image, and screenshots for visual document retrieval, 2025. URL <https://nomic.ai/blog/posts/nomic-embed-multimodal>.
- Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4593–4601.
- Tomoro AI. 2026a. tomoro-colqwen3-embed-4b. <https://huggingface.co/TomoroAI/tomoro-colqwen3-embed-4b>. Hugging Face model card. Accessed 2026-05-18.
- Tomoro AI. 2026b. tomoro-colqwen3-embed-8b. <https://huggingface.co/TomoroAI/tomoro-colqwen3-embed-8b>. Hugging Face model card. Accessed 2026-05-18.
- VAGO Solutions. 2025a. SauerkrautLM-ColQwen3-4b-v0.1. <https://huggingface.co/VAGOSolutions/SauerkrautLM-ColQwen3-4b-v0.1>. Hugging Face model card. Accessed 2026-05-18.
- VAGO Solutions. 2025b. SauerkrautLM-ColQwen3-8b-v0.1. <https://huggingface.co/VAGOSolutions/SauerkrautLM-ColQwen3-8b-v0.1>. Hugging Face model card. Accessed 2026-05-18.
- ViDoRe. 2026. ViDoRe leaderboard. <https://huggingface.co/spaces/vidore/vidore-leaderboard>. Accessed 2026-05-15.
- ViDoRe Team. 2024. colqwen2-v1.0. <https://huggingface.co/vidore/colqwen2-v1.0>. Hugging Face model card. Accessed 2026-05-18.

ViDoRe Team. 2025a. colqwen2.5-v0.2. <https://huggingface.co/vidore/colqwen2.5-v0.2>. Hugging Face model card. Accessed 2026-05-18.

ViDoRe Team. 2025b. colSmol-500M. <https://huggingface.co/vidore/colSmol-500M>. Hugging Face model card. Accessed 2026-05-18.

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

webAI. 2026. webAI-ColVec1-9b. <https://huggingface.co/webAI-Official/webAI-ColVec1-9b>. Hugging Face model card. Accessed 2026-05-18.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and 1 others. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. 2024. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. *Advances in Neural Information Processing Systems*, 37:127129–127153.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 1503–1512.

Appendices

This appendix provides supplementary material for Argus. It is organized as follows:

- Appendix A reports the source-level training data mixture used by all released Argus checkpoints.
- Appendix B lists the optimizer and routing hyperparameters for the warmup and joint training phases.
- Appendix C summarizes the head dimensions, training pool sizes, and post-training tricks used by the strongest open baselines on the V1+V2 leaderboard.
- Appendix D details the offline-cache inference pipeline and reports per-page encoding and per-query latency measurements.
- Appendix E compares Argus-9B under multi-vector late interaction against average-pooled single-vector scoring on V1+V2.
- Appendix F states the hypothesis behind each row of the component ablation in Table 7.
- Appendix G shows a same-page two-query routing visualization on a ViDoRe V2 Economics page.
- Appendix H visualizes the V1 / V2 scaling trend across the Argus 2B/4B/9B family.
- Appendix I gives the full agent prompts and serving protocol used in the agentic ViDoRe V3 evaluation.
- Appendix J sweeps the latent expert-bank size K on Argus-4B.
- Appendix K sweeps the router tap depth on Argus-4B.

Key Tables

- Table A.1: training mixture after source balancing (593,677 pairs).
- Table B.1: training hyperparameters for the warmup and joint phases.
- Table D.1: per-page offline encoding cost and per-query online retrieval latency on ViDoRe V2 ESG Reports.
- Table D.2: family-wide encoding and latency measurements on V2 Economics Reports.
- Table E.1: Argus-9B under multi-vector late interaction vs. average-pooled single-vector scoring.
- Table J.1: expert-count ablation on Argus-4B.
- Table K.1: router tap-depth ablation on Argus-4B.

Key Figures

- Figure D.1: the Argus inference pipeline showing offline document caching and the per-query online path.
- Figure D.2: storage–accuracy trade-off across systems with both storage and V1+V2 results available.
- Figure G.1: same-page, different-query routing visualization on a ViDoRe V2 Economics page.
- Figure H.1: V1/V2 scaling of the Argus family.
- Figure I.1: main prompt used by the Qwen3.6-27B agent.
- Figure I.2: selection- agent prompt used as an auxiliary reranking stage.

A Training Data Mixture

Table A.1 gives the source-level training mixture used by the released Argus checkpoints. The main paper reports only the aggregate size; the table is included here for reproducibility.

Table A.1: Training mixture after source balancing. The final training set uses 593,677 examples. The aggregate ratio to the 6.4M-example source pool is dominated by the Docmatix-IR cap (60K of 5.6M); excluding Docmatix-IR, 533K of 776K pairs ($\approx 69\%$) are used, and the core visual-document sources are used in full.

Source	Full	Used
visrag_synthetic	239,206	60,000
visrag_in_domain	122,752	60,000
tatdqa_train	13,251	13,251
tabfquad_train	1,552	1,552
vdr_multilingual_en	53,512	53,512
vdr_multilingual_de	58,217	58,217
vdr_multilingual_fr	55,270	55,270
vdr_multilingual_es	58,738	58,738
vdr_multilingual_it	54,942	54,942
colpali_arxiv_qa	9,949	9,949
colpali_docvqa	39,302	39,302
colpali_infographic_vqa	10,038	10,038
colpali_pdf	45,718	45,718
colpali_tatdqa	13,188	13,188
docmatix_resolved	5,612,890	60,000
Total	6,388,525	593,677

B Training Hyperparameters

Table B.1 summarizes the training hyperparameters used for the released Argus checkpoints. The same

values are used at every scale (2B, 4B, 9B); only the per-device micro-batch size differs, with gradient accumulation keeping the effective batch size fixed at 64. All three scales train on the identical 593,677-pair mixture (Table A.1), so the scaling comparison in §6.1 is unconfounded. All weights are saved in fp32 with merged PEFT after training.

Table B.1: Training hyperparameters for the released Argus checkpoints. The warmup phase trains only the router and experts against the auxiliary losses; the joint phase unfreezes PEFT and the fusion gates and adds the retrieval objective.

Hyperparameter	Router warmup	Joint
PEFT/LoRA rank	frozen	32
Train MoE gates	frozen	yes
K (latent experts)	4	4
Top- k routing	2	2
Region size r	4	4
Router layer	$L-5$	$L-5$
Router temperature	1.0	1.0
Router noise std	0.1	0.1
λ_{rt}	0.1	0.0
λ_{bal}	0.5	0.5
Negatives per sample	2	2
Learning rate	1×10^{-5}	5×10^{-6}
Effective batch size	64	64
Precision	bf16/fp32	bf16/fp32

C Baseline Configuration Context

For context on the V1+V2 leaderboard comparison in §5.1, the strongest open baselines use the following design choices. The Nemotron ColEmbed V2 cards report 4096-dimensional outputs for the 8B model and 2560-dimensional outputs for the 4B model, with model merging and an approximately 500k-image training mixture (NVIDIA, 2026b,a). Ops-Colqwen3-4B reports its strongest V1+V2 setting at 2560 dimensions (Alibaba Cloud OpenSearch-AI, 2026). The webAI 9B card reports a 2560-dimensional projection head and roughly 2M question-image training pairs (webAI, 2026), while the athrael-soju ColQwen3.5 card reports a 320-dimensional trained head, roughly 776k pairs, three training seeds, seed averaging, and model soup (athrael soju, 2026). Argus uses a 1024-dimensional head at every size and the balanced training mixture summarized in Appendix A, without seed averaging or post-training checkpoint merging.

D Query-Time Inference Cost

Although the Argus document representation is query-conditioned, the per-query cost can be

made comparable to that of a query-independent late-interaction retriever by caching all query-independent visual features once per corpus (§6.4). To verify that the added router and fusion path does not impose a large query-time penalty, we measure end-to-end retrieval latency on a fixed set of 50 ViDoRe V2 queries.

Figure D.1 separates the offline and online parts of the computation. During offline indexing, each page is passed through the Qwen-VL document image encoder once. This produces the intermediate grid used by the router, the final grid used by the expert bank, pooled region descriptors, and region coordinates. None of these cached tensors depends on the query.

At query time, Argus does not concatenate the query with the image and does not rerun the image encoder. The text query is encoded once to obtain query tokens $\mathbf{Q}$ and the routing context $\mathbf{z}_q$. When a candidate set is used, it comes from single-vector Argus over the same cached index (Appendix E); no second model is required. For each candidate page, the router combines $\mathbf{z}_q$ with cached region descriptors and coordinates, produces top- k expert weights, and applies gated fusion to the cached final grid. The projection head then produces $\mathbf{D}(q)$ for MaxSim scoring. Thus, query conditioning is implemented after visual feature extraction, not by repeating the full document encoder per query.

There are two practical caching modes. In the default mode, the latent experts are applied online to the cached final grid for the candidate pages only. In the larger-storage mode, the outputs of all latent experts are also cached offline, so online document work reduces to router weights, weighted summation, gated residual fusion, projection, and MaxSim. In both modes, the cost scales with the candidate set size rather than with the full corpus.

Protocol. We use the ViDoRe V2 ESG Reports task as a stable, leaderboard-equivalent benchmark: the same 452-page corpus and 232 queries that produce the V2 ESG score in Table 2. For each model we take the first 100 corpus pages and the first 55 queries (5 discarded as warmup, 50 timed). The document encoder is run once per page and the resulting query-independent features are cached (Figure D.1). The online path encodes the query text, runs the router, expert mixing, gated fusion, and projection over the cached features, and scores the resulting $\mathbf{D}(q)$ with MaxSim; for Nemotron the same timed region covers query encoding and

MaxSim against its cached index. We synchronize the GPU before and after each timed region. All measurements use a single NVIDIA H100 80 GB GPU, bf16 inference, batch size 1 (worst case for latency), and the same Python environment (PyTorch 2.11, transformers 5.7, MTEB 2.12, flash-attn 2.6) for both models.

Table D.1: Per-page offline encoding cost and per-query online retrieval latency on the ViDoRe V2 ESG Reports task (first 100 pages, 50 timed queries), measured on one H100 80 GB at batch 1.

	Argus-9B	Nemotron-colembed-8b-v2
Doc encode (ms / page, offline)	373.9	5090.1
Query online, mean (ms)	135.8	277.6
Query online, std (ms)	16.2	17.3
Query online, p50 (ms)	124.7	268.2
Query online, p95 (ms)	161.1	305.9

On this leaderboard-equivalent V2 ESG corpus, Argus-9B is $13.6\times$ faster than Nemotron-colembed-8b-v2 at offline document encoding (374 vs. 5090 ms per page) and $2.0\times$ faster per query at online retrieval (136 vs. 278 ms). The adaptive-tiling regime that Nemotron uses to handle large PDF-style pages amplifies the cost gap on visually rich documents; Argus’s fixed token budget keeps both the offline index and the per-query MaxSim cost bounded. At a per-page working cache comparable to Nemotron’s index (17.8 vs. 18.9 MB, Table 8), this makes the deployment trade-off favorable for Argus on the V2 distribution where its accuracy advantage is also the largest.

Family-wide scaling. For completeness, we also measure offline and online latency across the full released Argus family on the V2 Economics Reports task (the same 100-page / 50-timed-query protocol as above). The query text load differs slightly from the ESG split, so the absolute numbers in Table D.2 are not directly comparable to Table D.1; the goal here is only to track how the cost moves with model size on a fixed task.

E Single-Vector vs. Late-Interaction Scoring

To help understand the retrieval-accuracy difference obtained with a single pooled vector and with late-interaction scoring, we evaluate the same released Argus-9B checkpoint under both regimes, using the same processor and the same MTEB ViDoRe V1 and V2 evaluation protocol used for

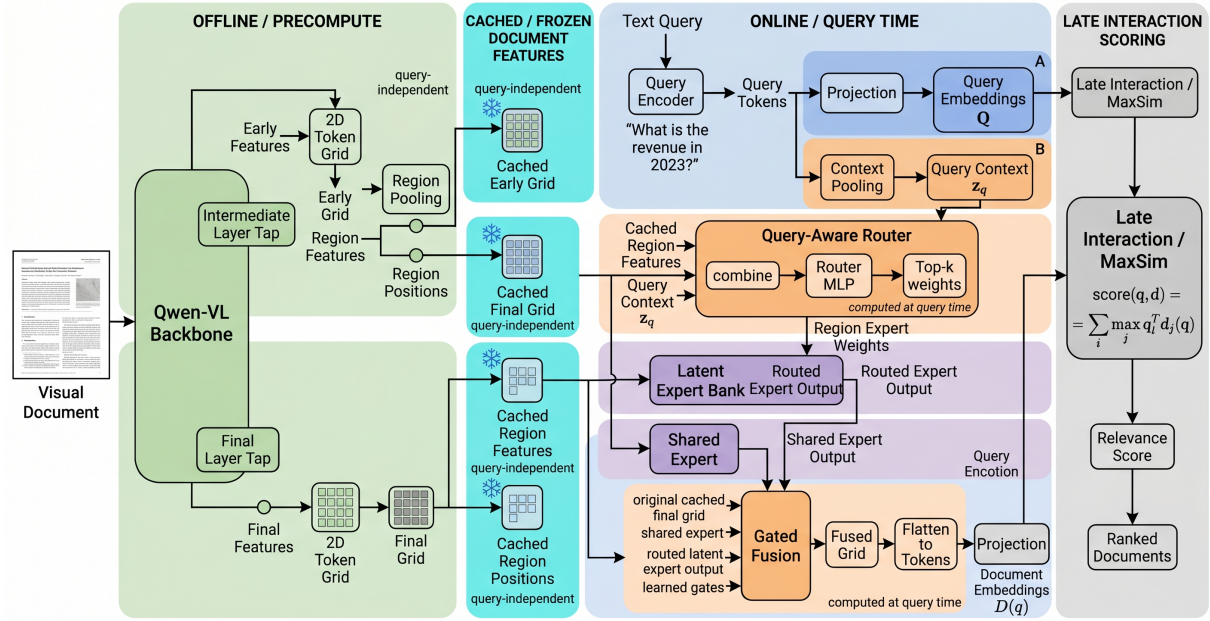


Figure D.1: **Argus inference pipeline.** The offline precompute pass caches all query-independent document features (early grid, final grid, region features, region positions, and optionally the latent expert outputs). The document image encoder is not run at query time. The online path encodes the text query, routes cached regions with z_q , fuses cached visual features, projects the resulting tokens, and applies MaxSim to candidate pages, producing the query-conditioned document embeddings $D(q)$.

Table D.2: Per-page encoding cost and per-query latency for the Argus family on the V2 Economics Reports task (first 100 pages, 50 timed queries), same single-H100 / batch-1 protocol as Table D.1.

Model	Doc/page (ms)	Query mean (ms)	Query p50 (ms)
Argus-2B	184.1	102.6	101.1
Argus-4B	226.1	131.5	130.3
Argus-9B	231.0	132.8	131.5

Table 2. In the single-vector regime, the per-token document and query embeddings are average-pooled along the token axis and L2-normalized, and relevance is computed by cosine similarity between the resulting single vectors; in the late-interaction regime, MaxSim is applied to the same multi-vector outputs as in the main paper. No re-training is performed.

Table E.1: Argus-9B retrieval accuracy (NDCG@5) under multi-vector late interaction vs. average-pooled single-vector scoring, on ViDoRe V1+V2 with the same checkpoint and evaluation protocol.

Argus-9B	V1	V2	V1+V2
Single vector	82.7	59.2	76.0
Late interaction	92.7	69.2	86.0
% improvement	+12.1%	+16.9%	+13.2%

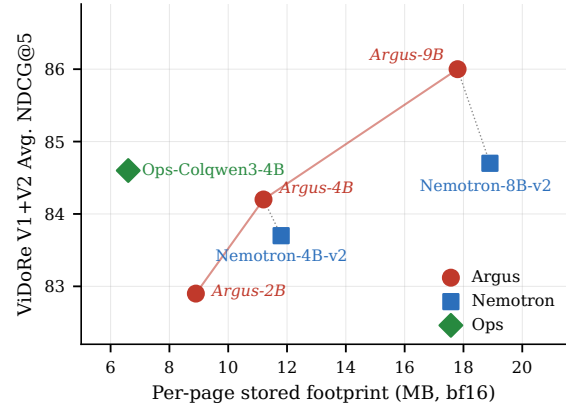


Figure D.2: **Storage and accuracy trade-off.** Per-page stored footprint (Table 8, bf16) versus ViDoRe V1+V2 average NDCG@5, for models where both storage settings and V1+V2 results are available. Dotted lines connect matched-size pairs: Argus stores slightly less than the corresponding Nemotron model at higher accuracy.

The single-vector mode trades roughly 13% relative NDCG@5 for a $\sim 2,000\times$ reduction relative to the projected multi-vector index, and $\sim 8,900\times$ relative to the Argus-9B working cache (Table 8), and a correspondingly cheap ANN index. The V2 gap is larger than the V1 gap (+16.9% vs. +12.1%), which is consistent with the observation that out-

of-domain pages have evidence spread over many small regions and benefit most from the per-token matching that late interaction enables.

F Ablation Component Hypotheses

Each row of Table 7 isolates one hypothesis behind the Argus design.

- **query conditioning** ($\mathbf{z}_q$). Drops $\mathbf{W}_{\text{query}}\mathbf{z}_q$ from the router input, so routing depends only on region content and position. This measures the contribution of query-conditioned routing, the central novelty of Argus.
- **region position** ($\mathbf{c}_r$). Drops $\mathbf{W}_{\text{coord}}\mathbf{c}_r$, so the router sees content and query but not location, testing the value of positional bias.
- **shared expert** ($\sigma(\alpha)=0$). Forces $\sigma(\alpha)=0$, so the entire expert contribution must flow through routed latent experts. This removes the always-on dense fallback path that stabilizes early training.

G Per-Query Routing Visualization

Figure G.1 accompanies the qualitative claim in §6.3 that the same document page can receive different sparse routing patterns under different queries. We treat the example as illustrative; the controlled test of query conditioning is the ablation in Table 7.

H Scaling Visualization

Figure H.1 visualizes the scaling trend discussed in §6.1: ViDoRe V1 saturates near 92.7 at the 4B scale, while the V2 (out-of-domain) average continues to grow with backbone size.

I Agentic Retrieval Evaluation Details

The agentic results in Figure 3 evaluate Argus as the retrieval engine inside our agentic search pipeline. The purpose of this experiment is to test whether a strong late-interaction retriever remains useful when the query is no longer a single fixed string, but is instead reformulated and expanded by an LLM during search. The agent is given the user query, an initial Argus ranking, and a small set of retrieval tools. It can issue additional searches, inspect the returned document snippets, and then commit to a final ranked list. The evaluation uses only the final list of document ids, so the reported score measures retrieval quality rather than answer generation.

Model serving and evaluation protocol. We serve Qwen3.6-27B with vLLM (Kwon et al., 2023) on two H100 GPUs using tensor parallelism 2, a 65,536 token context window, and the Qwen3 reasoning and XML tool-call parsers. Argus runs on a separate H100 GPU and serves all retrieval calls made by the agent. Each run uses a target list size of 10, a retriever candidate budget of 100, concurrency 4, high reasoning effort, and the corresponding ViDoRe V3 language filter. The metric is NDCG@10 over the final document ranking returned by the agent. No generated answer is evaluated.

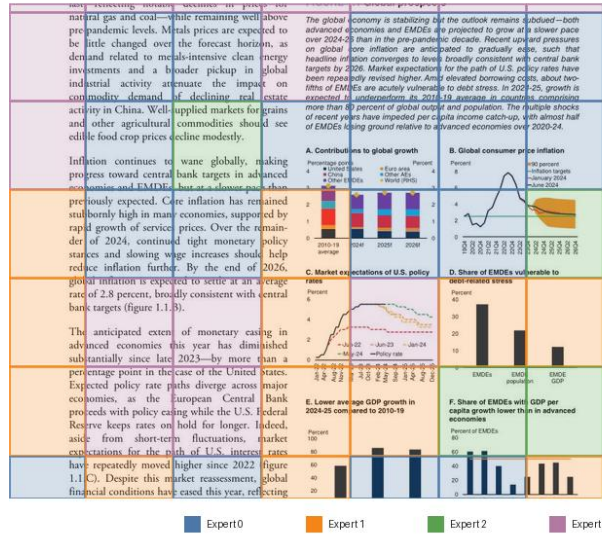
Agent state and actions. The agent starts from the original query and the first Argus retrieval result list. It may then run for at most 200 LLM steps. At each step, the LLM can choose one of three actions: retrieve, think, or final-results. The retrieve action sends a new text query to Argus and returns document ids, retrieval scores, and markdown snippets. This allows the agent to search for alternative phrasing, missing entities, related terminology, or subquestions that are implicit in the original query. The implementation tracks which documents have already been shown to the agent, so repeated documents are not resent with full content. The think action records intermediate reasoning without calling the retriever. The final-results action ends the trajectory and must return exactly ten document ids sorted by decreasing relevance.

Final ranking extraction. The primary output is the document list passed to final-results. This list is used directly whenever it is valid. If an interaction does not produce a valid final tool call, we apply deterministic fallbacks so that every query still has a reproducible ranking: first reciprocal-rank fusion over all retrieval calls in the trajectory, then the auxiliary selection output, and finally the initial Argus ranking if the LLM call fails. The run dictionary assigns larger scores to higher-ranked final document ids, and the ViDoRe evaluator computes NDCG@10 from this ranked list.

J Number of Experts Ablation

Argus uses $K=4$ latent experts with top-2 routing. This section sweeps the expert-bank size while holding everything else fixed: the Argus-4B initialization, training mixture (§A), warmup-then-joint schedule, and router tap at $L-5$. Only K and the top- k selection are changed; each vari-

Which economies had the highest projected growth in 2025?



What trend does the chart show over time?

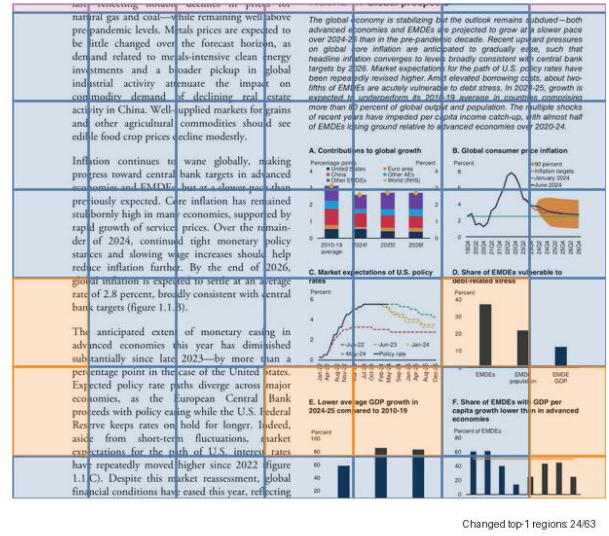


Figure G.1: **Same page, different query contexts.** Top-1 expert assignment per region of an identical ViDoRe V2 Economics page under two queries. The routing pattern changes from top-1 expert counts of (18, 17, 13, 15) to (39, 14, 4, 6), and 24 of 63 regions change their top-1 expert. This single example illustrates query-sensitive routing, but does not assign fixed semantic labels to experts.

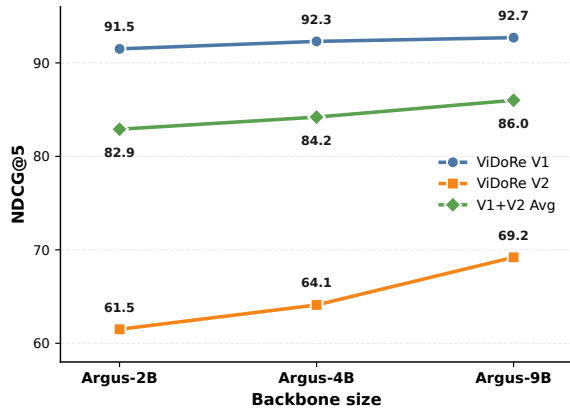


Figure H.1: **Scaling of the Argus family on ViDoRe V1+V2.** The 4B→9B step is concentrated in the harder out-of-domain V2 average (+5.1), while in-domain V1 saturates around 92.7.

ant is retrained from scratch. The hypothesis behind $K=4$ is that visual documents have roughly four recurring content modes (running text, tables, charts, and figures/formulas), so $K=4$ should give the router enough capacity to specialize without starving any expert of gradient signal under our $\sim 594k$ -pair training mixture.

The V1+V2 average forms an inverted-U with a peak at $K=4$. Moving from $K=1$ to $K=4$ adds +3.6 on average, with the larger share of the gain (+4.6) on the out-of-domain V2 split: specializa-

Table J.1: Argus-4B expert-count ablation (NDCG@5). All variants share the released training schedule, mixture, and router tap; only the expert-bank size K and top- k change.

Configuration	K	top- k	V1	V2	Avg
– MoE (single FFN)	1	1	89.0	59.5	80.6
Two experts, hard route	2	1	90.5	61.5	82.2
Two experts, soft mixture	2	2	91.0	62.0	82.7
Argus-4B (full)	4	2	92.3	64.1	84.2
Wider bank, same top- k	8	2	92.4	63.5	84.0
Wider bank, larger top- k	8	4	92.5	63.8	84.2
Very wide bank	16	2	91.5	61.5	82.9

tion helps most where the document distribution drifts from the training data. Beyond $K=4$, V1 holds roughly flat while V2 begins to drop, consistent with each expert receiving fewer training updates as the bank widens. The $K=8$, top-4 row recovers the lost V2 ground partially, which is consistent with the per-expert undertraining explanation: giving each expert more gradient signal (via a larger top- k) compensates for the wider bank. The $K=16$ row drops below $K=4$ on both splits, indicating that the load-balance loss can no longer keep all experts active under the available supervision.

$K=1$. Collapses the expert bank to a single position-wise FFN; routing becomes a no-op. This is the strongest test of the MoE hypothesis: removing all specialization should drop both V1 and V2.

$K=2$, **top- $k=1$ vs. top- $k=2$** . Distinguishes hard binary specialization from a learned soft mixture of two experts. Both lose to $K=4$, showing two specialists are not enough to cover the recurring visual-document content types.

$K=8$, **top- $k \in \{2, 4\}$** . Tests whether more capacity helps if each expert still receives enough gradient. Top-4 partially closes the gap to $K=4$, supporting the per-expert-undertraining account.

$K=16$. Stress test for the load-balance loss under our balanced training mixture. The degradation is the expected sign of expert collapse.

K Router Tap Depth Ablation

Argus taps the document backbone at two depths: $L-5$ for the router input and L for the latent expert bank and shared expert. The choice of $L-5$ is motivated by probing studies showing structural and positional information peaks in middle-to-late transformer layers (Tenney et al., 2019; Karamcheti et al., 2024). This section sweeps the router tap depth while holding the retrieval tap at L , the expert bank, the training mixture, and the schedule fixed.

Table K.1: Argus-4B router tap-depth ablation (NDCG@5). Each row retrains the full model with the router tap moved to a different backbone layer; the retrieval tap stays at L .

Router tap	V1	V2	Avg
L (final layer)	91.0	62.0	82.7
$L-2$	91.8	63.2	83.4
$L-3$	92.2	63.8	84.0
Argus-4B (full, $L-5$)	92.3	64.1	84.2
$L-7$	92.0	63.5	83.9
$L-10$ (mid stack)	91.3	62.5	83.2
$L-15$ (early stack)	89.5	59.5	81.0

The average follows the same inverted-U shape as the expert-count sweep, peaking at $L-5$ and degrading at both ends. Tapping at L forces the router and retrieval head to share final-layer features that have already specialized for the retrieval objective; routing then becomes nearly redundant with the shared-expert path, and the model loses approximately 1.5 V1+V2 points. Tapping at $L-15$ drops the average by 3.2 points: at that depth the document tokens carry mostly low-level visual features, and the router has too little semantic structure to discriminate between content modes. Tapping at $L-10$ also degrades, indicating that purely mid-stack features are too generic for query-conditioned routing. The $L-3$ to $L-7$ band is the flattest part

of the curve; $L-5$ is the empirical maximum but anywhere in this band would work, supporting the qualitative claim that the relevant signal lives in the late-but-not-final layers.

As in the expert-count ablation, V2 is the more sensitive axis: the spread across the sweep is roughly 4.6 points on V2 versus 2.8 on V1. This reinforces the broader pattern that query-conditioned routing pays off most under domain shift, where the document representation has the most to adapt to.

Agentic retrieval main prompt

System Prompt: You are a retrieval agent that finds all documents related to a given query.

Goal. You are given a search query and a list of documents retrieved for that query. Your task is to write new queries and use the search tool to find all related and somewhat related documents to the given query, with the objective of maximizing recall. If the user's query is a question, do not answer the question yourself. Instead, find the related documents for the query.

Relevance definition.

- The meaning of query, document, and relevance can be more complex than in standard web search.
- A relevant document is one that is useful for a user who is searching for the given query.
- The agent should analyze the query and available documents, then reason about what relevance means for this specific task.
- If the query is itself a prompt for an LLM, relevant documents are those that help the LLM answer or solve it.

Workflow.

- The agent is given a retrieval tool powered by a dense embedding model.
- The agent may call the retrieval tool multiple times.
- The agent should search from different angles and revise queries based on documents found in earlier steps.
- Once confident that the relevant and somewhat relevant documents have been found, the agent must call `final_results`.
- The `final_results` call must contain exactly ten document ids.
- The returned document ids must be sorted in decreasing relevance to the query.

Best practices.

- Be thorough and find all related and somewhat related documents.
- Prioritize recall: if multiple documents are relevant, find and report all of them, even if only a subset is enough to answer the query.
- Use the initially retrieved documents to decide what additional queries could reveal missing relevant documents.

Available tools.

- `retrieve(query, top_k)`: search the corpus with the active late-interaction retriever.
- `think(thought)`: log intermediate reasoning without retrieving new information.
- `final_results(doc_ids, message, search_successful)`: return the final ranked list and end the interaction.

Figure I.1: Main prompt used by the Qwen3.6 27B agent in the agentic ViDoRe V3 retrieval evaluation. The prompt asks the agent to treat retrieval as a recall-oriented search problem, use Argus for additional query formulations when needed, and return a ranked top-ten document list.

Agentic retrieval selection prompt

System Prompt: You are a document re-ranker agent, which is the final stage in an information retrieval pipeline.

Role. You are given a search query and a list of retrieved candidate documents that are potentially relevant to the query. Your goal is to identify the most relevant documents from the candidate list.

Workflow.

- Read the query carefully and understand the user's information need.
- Compare the query with each candidate document and judge how useful the document is for the query.
- Select the required number of most relevant candidate documents.
- Sort the selected documents by decreasing relevance.
- Call `log_selected_documents` to report the final selection and finish the task.

Thinking tool.

- Use think for complex analysis, ambiguous queries, difficult relevance judgments, and ordering selected documents.
- The think tool does not retrieve new information; it only records reasoning for transparency.

Figure I.2: Selection-agent prompt used as an auxiliary reranking stage. The pipeline computes top-five and top-ten selections, but the primary evaluated output is the final-results tool call from the main agent; selection results are used only as a later fallback when extracting a final ranked list.