UNLOCKING AND UNDERSTANDING THE RETRIEVAL POTENTIAL OF FROZEN GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative models encode rich information about their inputs, yet their default hidden-state representations are often poorly suited for retrieval. We investigate the retrieval potential of frozen generative models and show that optimizing a few continuous guiding embeddings, with under 50K trainable parameters, yields effective retrieval representations without modifying backbone weights or causal attention. Experiments on three retrieval datasets demonstrate substantial improvements over frozen baselines, with performance approaching or exceeding specialized retrievers. Even a single guiding embedding, requiring only a few thousand parameters in total, retains competitive performance. The learned guiding embeddings also generalize well across datasets without further training. Beyond retrieval quality, our representation analysis reveals that distinct guiding embeddings learned in separate runs steer the same frozen backbone toward increasingly similar input-dependent representations. Queries and candidate answers encoded using guiding embeddings from different runs can be matched with little average performance loss, demonstrating functional compatibility beyond comparable retrieval scores. Together, these findings reveal substantial retrieval capability accessible through remarkably small adaptations to frozen generative models. This capability is not tied to a unique set of guiding embeddings. Distinct learned input parameterizations can induce closely aligned representations and compatible retrieval behavior within the same frozen backbone.

1 INTRODUCTION

Generative models encode inputs to support next-token prediction, not to organize independently encoded queries and candidate answers by relevance. Their default hidden states can therefore be poorly suited for retrieval despite containing rich information about the inputs. Sentence embedding methods address this mismatch through paired or contrastive supervision (Reimers & Gurevych, 2019; Gao et al., 2021), and recent approaches adapt generative models into strong encoders through model fine-tuning or changes to attention and pooling (Wang et al., 2024; BehnamGhader et al., 2024; Lee et al., 2025). These successes leave open how much retrieval capability can be accessed without changing the generative backbone itself.

Prompting offers a simpler route. PromptEOL and E5-V show that asking for a one-word summary can improve text and multimodal representations without fine-tuning (Jiang et al., 2024a;b). The instruction changes the hidden state used for retrieval. The substantial gains achieved through prompting alone suggest that generative models possess considerable latent potential to produce high-quality retrieval representations. However, natural-language prompts can only select from a finite vocabulary, substantially limiting the flexibility of input conditioning.

We therefore study a few continuous *guiding embeddings*: vectors with the same dimension and input role as word embeddings, but optimized freely within the model's continuous input embedding space rather than restricted to vectors associated with vocabulary tokens. Appended after the input content, they guide a frozen model to produce a retrieval-effective final hidden state. Only these vectors are optimized. Backbone weights, the vocabulary table, and causal attention remain unchanged. Building on soft-prompt tuning and its application to frozen language-model encoders (Lester et al., 2021; Zhao et al., 2025), we aim both to *unlock* retrieval capability through compact input adaptation and to *understand* how the learned embeddings guide frozen generative models toward effective retrieval.

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.

Experiments on MMDocRAG, SciMMIR, and ViDoSeek show substantial retrieval gains over default and fixed-prompt representations, approaching or even exceeding specialized retrievers in several settings. On MMDocRAG, adaptation succeeds across five backbones with ten guiding embeddings shared between queries and candidate answers, totaling 25,600–40,960 trainable parameters. Even a single shared embedding retains strong retrieval with only 2,560 parameters in Qwen3-VL-4B. Furthermore, although dataset-specific adaptation yields the strongest results, directly transferring the learned guiding embeddings to other datasets still substantially outperforms the unadapted base model, without any target-dataset training.

Although motivated by retrieval-oriented prompts, we find that effective guiding embeddings do not require meaningful initialization. Training succeeds from natural-language prompts, random vocabulary entries, and Gaussian samples. The resulting embeddings remain distinct yet all support effective retrieval. Moreover, they steer the same frozen backbone toward increasingly similar input-dependent representations. Queries and candidate answers encoded using embeddings learned in separate runs can be matched with little average performance loss, without a learned alignment map. Across different backbones, we also observe similar relational structure among representations. Together, these findings suggest that retrieval capability is not tied to a particular set of guiding embeddings, but can arise from distinct embeddings that induce closely aligned, retrieval-effective representations within the same frozen backbone.

In summary, our contributions are as follows:

1. Revealing retrieval potential in frozen generative models. We show that generative models possess substantial retrieval capability that can be elicited through input adaptation alone, without modifying backbone weights or causal attention. Our learned guiding embeddings substantially improve retrieval, achieving performance competitive with specialized retrievers.

2. Accessing retrieval capability with remarkably few parameters. We demonstrate that this capability can be unlocked by optimizing only a few guiding embeddings shared between queries and candidate answers. Even a single learned embedding, totaling 2,560 trainable parameters, retains strong retrieval performance, while cross-dataset transfer without further training makes this adaptation both parameter-efficient and reusable.

3. Uncovering compatible retrieval behavior across distinct adaptations. We find that different initializations yield distinct guiding embeddings that steer the same frozen backbone toward increasingly similar representations. Queries and candidate answers encoded across separate runs remain compatible with little average performance loss and no learned alignment, showing that effective retrieval is not tied to a unique input parameterization.

2 Related Work

Retrieval representations from generative models. LLM-based encoders improve retrieval through contrastive training and, in some methods, modified attention or pooling (Wang et al., 2024; BehnamGhader et al., 2024; Lee et al., 2025). Input-only strategies also improve extracted representations through summarization or repetition (Jiang et al., 2024a;b; Springer et al., 2025). These findings motivate exploring continuous input conditioning beyond fixed discrete prompts.

Continuous conditioning and transfer. Prompt and prefix tuning condition frozen models with trainable vectors (Lester et al., 2021; Li & Liang, 2021); most directly, Zhao et al. (2025) apply prompt tuning to frozen causal text encoders. SPoT, ATTEMPT, and MPT transfer source-prompt knowledge while learning target-task parameters (Vu et al., 2022; Asai et al., 2022; Wang et al., 2023). Building on this foundation, we study compact, shared guiding embeddings for retrieval.

Multimodal retrieval. VLM2Vec, MM-Embed, and GME learn multimodal embeddings through contrastive adaptation (Jiang et al., 2024c; Lin et al., 2025; Zhang et al., 2025), while ColPali and ColQwen use multi-vector late interaction (Faysse et al., 2025; 2024). We study single-vector retrieval from frozen backbones on complementary visual benchmarks (Dong et al., 2026; Wu et al., 2024; Wang et al., 2025), rather than updating the backbone.

Representation similarity and functional compatibility. Prior work quantifies representation similarity and investigates agreement across models and datasets (Kornblith et al., 2019; Huh et al., 2024; Ciernik et al., 2024). Model stitching tests functional compatibility through a learned connecting layer (Bansal et al., 2021). Our cross-run query–candidate answer matching instead tests compatibility within a frozen backbone without learned alignment.

3 Retrieval with Guiding Embeddings

3.1 Problem Setting

Given a query q and its candidate set $\mathcal{A}(q)$, retrieval ranks candidate answers by relevance to q. Here, a *candidate answer* denotes a retrievable item, such as an image or document page, rather than a generated answer. Each query can have multiple relevant candidates. Queries and candidates are encoded independently into single vectors, and matching uses cosine similarity. The candidate set may be query-specific or shared across queries.

Let M_θ be a generative model with vocabulary embedding table E, where each word embedding is an h-dimensional vector. Following the last-token pooling practice in prior works (Jiang et al., 2024a;b), we use the final-layer hidden state of the last non-padding token in the formatted input sequence as the input representation, without generating additional tokens. We seek retrieval-effective representations while keeping θ, E, and causal attention fixed. Instead of updating the model or adding a projection head, we optimize a short sequence of input vectors. These *guiding embeddings* are trainable parameters shared across examples, whereas the resulting retrieval representations depend on each input.

3.2 Training Guiding Embeddings

Continuous input conditioning. We learn a short, ordered *embedding sequence* $G = (g_1, g_2, \ldots, g_m)$, where each $g_i \in \mathbb{R}^h$ is a trainable guiding embedding. We focus on compact sequences, typically $m \leqslant 10$, with longer sequences considered only in capacity ablations. The complete sequence acts as an additional soft prompt. For every query and candidate answer, we append all m vectors after the input content in the same order, $g_1, g_2, \ldots, g_m$. The vector values are learned jointly in the continuous input embedding space.

Let e denote embedding lookup in the fixed vocabulary table E, so that $e(x_i) = E[x_i]$ is the embedding of token x_i. Given an input sequence $x = (x_1, x_2, \ldots, x_n)$, we append the guiding sequence G to its token embeddings, whether x is a query or a candidate answer. The model therefore processes $(e(x_1), \ldots, e(x_n), g_1, \ldots, g_m)$. Omitting fixed chat-template tokens for clarity, the retrieval representation is

$$r(x; G) = \mathrm{normalize}\big(\big[M_\theta\big(e(x_1), \ldots, e(x_n), g_1, \ldots, g_m\big)\big]_{\mathrm{last}}\big)\,. \tag{1}$$

Here $[\cdot]_{\mathrm{last}}$ selects the final-layer hidden state of the last non-padding token in the complete formatted input, and $\mathrm{normalize}$ applies L2 normalization. Queries and candidate answers are matched using the similarity score $s(q, a; G) = r(q; G)^\top r(a; G)$. In the implementation, we extract the input representation directly from the final hidden state without generating any answer tokens. For visual inputs, we retain the model's original visual encoding process. Analogous to obtaining text-token embeddings through e, images are converted into embeddings using the model's native visual encoder, and the guiding sequence G is appended to the resulting input embeddings.

To make these representations effective for retrieval, we optimize the shared guiding sequence G using a contrastive loss (Gao et al., 2021; Jiang et al., 2024b), while keeping the generative model parameters and vocabulary embedding table E frozen.

3.3 Analysis Protocols

We separate the choice of input parameters from the representations and retrieval behavior they induce. The following analyses test sensitivity to initialization, agreement across training runs, and compatibility under the retrieval score.

Initialization, sequence length, and transfer. We initialize G in three ways: **summarization**, copying vocabulary embeddings from a natural-language summarization instruction; **Gaussian**, sampling Gaussian vectors and rescaling each to the corresponding summarization vector's norm; and **vocabulary**, copying randomly sampled ordinary vocabulary entries without replacement or rescaling. We then vary the length of G and train each configuration from scratch to examine how many guiding embeddings are needed for effective retrieval. Together with the initialization comparisons, these experiments assess the effects of sequence length and meaningful initialization. Additionally, we evaluate cross-dataset transfer by directly reusing guiding embeddings learned on a source dataset, without further training or checkpoint selection on the target dataset. Specific prompts, sequence lengths, and training settings are given in Section 4.

Within-model representation agreement. To study how G guides input representations, we compare representations of the same input under different guiding embeddings and of different inputs under the same guiding embeddings. The learned embeddings depend on both the training settings and the checkpoint. We use u, v to index separate training runs, including their settings, and t to denote the number of training steps. Thus, $G^u(t_1)$ and $G^v(t_2)$ denote guiding sequences from runs u, v at steps t_1, t_2, which need not be equal. Each run uses the same sequence for queries and candidate answers. From an input set $\mathcal{X}$, we sample $x \in \mathcal{X}$ for the same-input comparison and pairs $(x, y) \in \mathcal{X} \times \mathcal{X}$ with $x \neq y$ for the different-input comparison. We measure their average cosine similarities as

$$S^{u,v}(t_1, t_2) = \mathbb{E}_{x \sim \mathcal{X}}\left[r(x; G^u(t_1))^\top r(x; G^v(t_2))\right], \tag{2}$$

$$S^{u}(t_1) = \mathbb{E}_{\substack{x,y \sim \mathcal{X} \\ x \neq y}}\left[r(x; G^u(t_1))^\top r(y; G^u(t_1))\right]. \tag{3}$$

We estimate these expectations by averaging over the sampled inputs and pairs. The different-input comparison provides a reference for assessing whether cross-run similarity reflects input-specific agreement rather than uniformly similar outputs. We perform both comparisons separately for queries and candidate answers.

Cross-run retrieval compatibility. Similarity alone does not establish useful matching. We therefore encode queries using $G^u(t_1)$ and candidate answers using $G^v(t_2)$, allowing the two runs to use different training steps, without retraining, calibration, or an alignment map:

$$s_{uv}(q, a; t_1, t_2) = r(q; G^u(t_1))^\top r(a; G^v(t_2)). \tag{4}$$

In this test, one guiding sequence encodes queries and another, learned in a separate run, encodes candidate answers. We then rank candidate answers using these similarity scores and evaluate retrieval performance to assess whether the resulting representations remain compatible.

Cross-model relational agreement. Different generative models need not share vector dimensions or coordinates. We compare representations from two models, guided by $G^u(t_1)$ and $G^v(t_2)$ learned in their respective training runs; the checkpoint steps t_1, t_2 need not match. For the same set of inputs across models, we compute linear CKA of normalized representations (Kornblith et al., 2019) and the Pearson correlation between each input's cosine-similarity profiles, excluding self-similarity from the profiles.

4 Experimental Setup

Datasets and metrics. We evaluate on MMDocRAG, SciMMIR, and ViDoSeek (Dong et al., 2026; Wu et al., 2024; Wang et al., 2025). For MMDocRAG, we train on the full official `dev_15` split (2,055 records) and test on `evaluation_15` (2,000 records). SciMMIR evaluates global retrieval over 16,263 test images with 16,263 queries. ViDoSeek contains 1,142 queries over 5,385 rendered pages and is evaluated by transferring MMDocRAG-trained guiding embeddings without target training. We use Recall@k (R@k) and MRR@20. For conciseness, the main text reports only R@1 among the recall cutoffs, alongside MRR@20. R@3, R@5, R@10, and R@20 follow the same broad trends and support the same conclusions. Complete retrieval metrics for the main comparisons, model and length experiments, and cross-run retrieval analyses are reported in Appendices G, H, and I, respectively. Metric definitions and aggregation are given in Appendix A.1.

Models and baselines. Our main generative model is Qwen3-VL-4B-Instruct (Bai et al., 2025); MMDocRAG experiments additionally cover LLaVA-OneVision-1.5-4B, InternVL3.5-4B, Ministral-3-3B, and Qwen3.5-9B. We compare against no-prompt and fixed-prompt versions of the same models, and specialized retrievers ColQwen2.5, Nomic-3B/7B, and ColNomic-7B (Faysse et al., 2024; Team, 2025). Nomic and our method use single vectors; ColQwen and ColNomic use multi-vector late interaction. Model-specific settings and computational differences are detailed in Appendix B.

Training and evaluation. Our main MMDocRAG and SciMMIR results are evaluated on their respective test sets after training on each dataset's training split; ViDoSeek results use MMDocRAG-trained embeddings without further training. We use ten guiding embeddings initialized from the 10-token prompt `Summarize the content above in one word:`, random vocabulary entries, or norm-matched Gaussian samples, with sequence-length ablations reported separately. Training uses AdamW for 2,500 steps at learning rate 10^{-3}, with checkpoints selected by validation performance. Selected configurations use two or three random seeds. Further details appear in Appendix A.

5 Unlocking Retrieval with Few Trainable Parameters

5.1 Guiding embeddings substantially improve retrieval

Table 1 shows a progression from default generative representations to fixed prompting and learned guiding embeddings. On MMDocRAG, MRR@20 rises from 22.22 without a prompt to 52.09 with the fixed instruction and 70.79 after learning its input embeddings. On SciMMIR, the corresponding values are 0.11, 25.55, and 51.20. We use ViDoSeek specifically to evaluate cross-dataset transfer: guiding embeddings trained on MMDocRAG are directly reused without ViDoSeek training, raising MRR@20 from 48.04 with the fixed prompt to 76.20. Thus, continuous input adaptation substantially improves retrieval both in-domain and beyond the training dataset. Appendix E reports additional transfer experiments between MMDocRAG and SciMMIR that support the same conclusion.

Learned guiding embeddings achieve retrieval quality comparable to specialized retrievers on MMDocRAG and SciMMIR, often trailing individual baselines by only 2–3 percentage points in R@1. In some comparisons they perform better, exceeding ColQwen2.5 on SciMMIR in both R@1 and MRR@20. Section 5.2 further shows that guiding embeddings applied to other generative models surpass these specialized baselines in additional settings. The gaps are larger on ViDoSeek under cross-dataset transfer, but the learned embeddings still move the frozen generative model from far below specialized retrievers to a competitive level without target-dataset adaptation.

Notably, these results require only 25,600 trainable parameters, compared with approximately 64–90 million for the LoRA-based retrievers in Table 1. Our adaptation therefore uses just 0.028–0.040% of their trainable parameter counts, less than one-thousandth. This advantage concerns adaptation size rather than total training or inference cost. Together, these results show that a frozen generative model

Table 1: Retrieval performance and trainable parameter counts. MRR denotes MRR@20. "Trainable" denotes the number of parameters optimized for retrieval adaptation. Baseline counts are approximate. Input tuning optimizes only our shared ten-embedding sequence, initialized from the summarization prompt and trained in-domain for MMDocRAG/SciMMIR or transferred from MMDocRAG for ViDoSeek. Baselines use LoRA adaptation and differ in training history and visual budget.

			MMDocRAG		SciMMIR		ViDoSeek	
Configuration	Tuning	Trainable	R@1	MRR	R@1	MRR	R@1	MRR
No added prompt	None	0	7.03	22.22	0.02	0.11	0.18	0.79
Fixed summarization	None	0	28.74	52.09	19.10	25.55	36.69	48.04
Guiding embeddings (ours)	Input	25.6K	45.10	70.79	41.27	51.20	65.67	76.20
ColQwen2.5-3B	LoRA	∼64M	47.51	71.69	40.93	49.63	75.22	83.92
Nomic-3B	LoRA	∼64M	47.30	72.23	44.17	52.89	71.72	81.36
Nomic-7B	LoRA	∼90M	48.25	73.64	46.44	55.48	71.28	81.16
ColNomic-7B	LoRA	∼90M	48.78	73.07	43.48	52.49	73.47	82.22

Table 2: MMDocRAG R@1 and MRR@20 across five frozen backbones. The `None` column uses no added prompt; `Fixed S` uses the untrained summarization instruction. Columns `G`, `V`, and `S` report trained Gaussian, vocabulary, and summarization initializations, respectively, with ten shared guiding embeddings. `None` and `Fixed S` have no trainable parameters.

		None		Fixed S		G		V		S	
Backbone	Params.	R@1	MRR	R@1	MRR	R@1	MRR	R@1	MRR	R@1	MRR
Qwen3-VL-4B	25,600	7.03	22.22	28.74	52.09	45.30	70.95	47.60	73.21	45.10	70.79
LLaVA-OV1.5-4B	25,600	4.72	17.47	18.78	37.89	35.46	61.40	40.47	66.31	36.94	62.72
InternVL3.5-4B	25,600	11.54	29.73	19.56	40.50	39.84	65.38	41.91	67.69	37.14	63.17
Ministral-3-3B	30,720	4.74	19.04	15.63	36.47	46.06	72.16	49.01	75.21	48.80	74.77
Qwen3.5-9B	40,960	4.71	19.33	4.88	17.86	48.95	75.32	47.35	73.88	45.98	72.20

can produce strong retrieval representations through a remarkably small learned input sequence, without modifying its model weights or causal attention.

5.2 Retrieval potential extends across frozen backbones

To test whether compact input adaptation unlocks retrieval beyond a single model, we evaluate five generative models spanning different architectures and sizes from 3B to 9B. We also train guiding embeddings from summarization prompts, random vocabulary entries, and Gaussian samples to test whether their effectiveness depends on a particular prompt or meaningful initialization. All settings use ten shared guiding embeddings and leave the generative model frozen.

Table 2 shows substantial retrieval gains across all five models and all three initialization strategies. These gains require only 25,600–40,960 trainable parameters, demonstrating that retrieval potential accessible through small input adaptations is shared across the tested architectures and model sizes. The effect is not restricted to models that already respond well to fixed prompting: Qwen3.5 improves only from 4.71 to 4.88 R@1 with the summarization instruction, but exceeds 45 R@1 after optimization across all three initialization strategies. Thus, weak default retrieval or limited gains from a discrete prompt need not indicate limited retrieval potential.

Extending the comparisons in Section 5.1, several configurations outperform every specialized retriever evaluated in Table 1 on both MMDocRAG metrics. Ministral with vocabulary initialization achieves 49.01 R@1 and 75.21 MRR, while Qwen3.5 with Gaussian initialization reaches 48.95 and 75.32, respectively. These results establish the strongest scores among the evaluated methods despite using only a small trainable input sequence.

Across the tested models, all three initialization strategies yield effective retrieval, with broadly similar performance within each model rather than a consistent advantage for meaningful prompts. Successful guidance therefore does not require a particular natural-language starting point. Inspection of the learned guiding embeddings further shows that they remain substantially different across initialization settings despite their comparable retrieval quality. Effective guiding embeddings are consequently not unique: distinct learned input sequences can unlock strong retrieval within the same frozen model. Section 6 examines how these different sequences shape output representations and retrieval behavior, while Appendix G reports the complete results.

5.3 Parameter Efficiency of Guiding Embeddings

Having established the effectiveness of guiding embeddings, we explore sequence length in both directions from our ten-embedding configuration: reducing it to a single embedding and extending it beyond the original length. These experiments examine how little input adaptation can sustain effective retrieval and whether additional embeddings provide further gains.

As shown in Table 3, a single guiding embedding retains competitive retrieval performance on MMDocRAG. Shared between queries and candidate answers, this vector requires only 2,560 trainable parameters. Its effectiveness is observed with both prompt-derived and random initialization (Appendix F), rather than depending on retaining a complete instruction. This further demonstrates how few parameters are needed to access substantial retrieval capability in a frozen generative model.

Table 3: Guiding-sequence length experiments of Qwen3-VL-4B on MMDocRAG. MRR denotes MRR@20.

Embeddings	1	3	5	7	10	15	20
Parameters	2,560	7,680	12,800	17,920	25,600	38,400	51,200
R@1	42.96	42.61	44.95	45.70	47.60	47.96	44.25
MRR	69.34	68.66	71.30	71.60	73.21	73.73	70.47

Second, more guiding embeddings do not necessarily yield better retrieval. Extending the sequence beyond the main configuration provides only modest gains. Thus, retrieval quality does not improve monotonically with the size of the trainable input sequence. Together, these findings support compact guiding sequences as an effective means of accessing the model's retrieval potential, without requiring a large adaptation parameter budget.

6 Understanding Representation and Functional Consistency

Having established the effectiveness of guiding embeddings, we next investigate how they guide frozen generative models to produce retrieval-effective representations. We systematically analyze multiple independently learned guiding sequences and the final input representations they induce, examining their relationships within and across models. Our analysis reveals three key findings.

6.1 Different guiding embeddings induce increasingly similar representations

To analyze how the final representations induced by guiding embeddings evolve during training, we examine checkpoints at 20%, 40%, 60%, 80%, and 100% of each run's best validation step. Runs are trained separately but share the backbone, task, and data.

Using these checkpoints, we compare the same-input and different-input output-cosine measures defined in Section 3.3 (Equations 2–3), separately for queries and candidate-answer images.

Figure 1(a,b) shows a substantial overall increase in same-input cross-run similarity as training progresses, alongside a decrease in different-input within-run similarity. This pattern holds for both queries and candidate answers. Although the guiding embeddings start from different initializations and remain distinct after training, they increasingly steer the same frozen model toward similar representations of the same input. This agreement therefore emerges in the induced output representations, rather than requiring the guiding embeddings themselves to converge to identical values. Meanwhile, the decreasing similarity between different inputs indicates that this agreement is input-specific, rather than a tendency to make all outputs uniformly similar. These trends are not strictly monotonic; Appendix D reports all pairwise measurements.

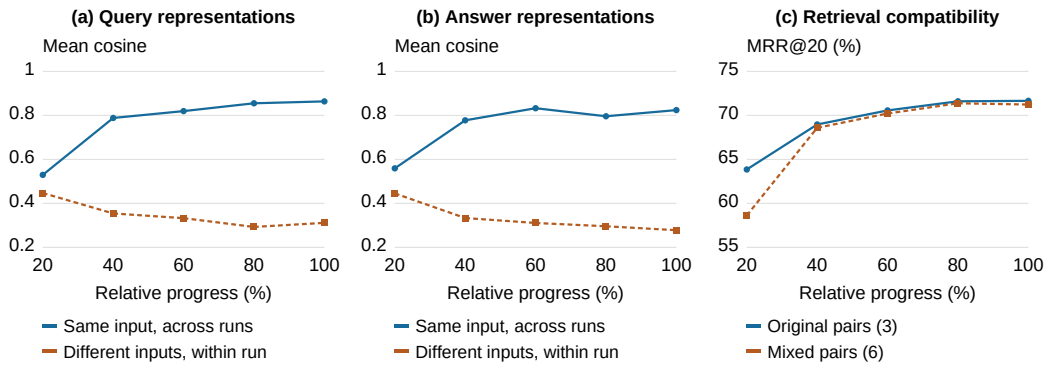

Figure 1: MMDocRAG training trajectories. Panels (a,b) average same-input output cosine over three cross-initialization pairs and different-input cosine over three within-run controls. Panel (c) averages three original and six cross-run pairings. Progress is relative to each run's selected checkpoint.

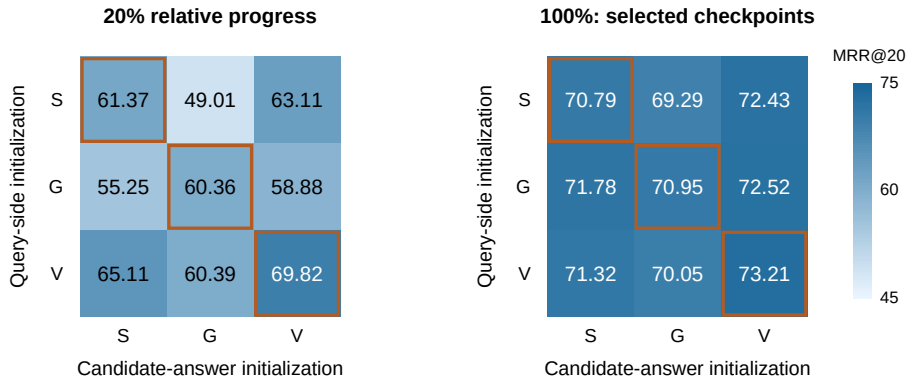

Figure 2: Cross-run matching on MMDocRAG. Rows select the run supplying query guiding embeddings; columns select the run supplying candidate-answer guiding embeddings. S: summarization, G: Gaussian, V: vocabulary. Outlined diagonal cells use the same run for both sides; all others combine separately trained runs without alignment or retraining. Values are MRR@20 percentages.

6.2 Queries and candidate answers remain compatible across training runs

To test whether the representation agreement observed above also supports retrieval, we encode queries and candidate answers using guiding embeddings learned in different runs of the same frozen model. We use three independently trained guiding sequences, initialized from summarization prompts (S), Gaussian samples (G), and random vocabulary entries (V). Each run trains one sequence shared by queries and candidate answers. We mix sequences from different runs only at evaluation.

For example, we encode queries using the S-trained sequence and candidate answers using the G-trained sequence, then retrieve answers by their representation similarity using Equation 4, without retraining or learning an alignment. Writing each pairing as (query run, candidate-answer run), the same-run references are (S,S), (G,G), and (V,V). The cross-run tests are (S,G), (S,V), (G,S), (G,V), (V,S), and (V,G). Reversing a pairing is a distinct test because it swaps which sequence encodes queries and which encodes candidate answers. We compare the average MRR@20 of these cross-run tests with that of the same-run references.

Figure 1(c) tracks these two averages during training: "Original pairs" denotes the same-run references, and "Mixed pairs" denotes the cross-run tests. At 20% progress, they achieve 63.85 and 58.63 MRR, respectively. At the selected checkpoints, these averages reach 71.65 and 71.23, reducing the mean gap from 5.22 to 0.42 points. Thus, independently learned guiding sequences become increasingly compatible for retrieval as training progresses.

Figure 2 shows the individual pairings: diagonal cells use the same guiding sequence for both queries and candidate answers, whereas off-diagonal cells use sequences from different runs. All off-diagonal pairings retain strong retrieval at the selected checkpoints, although their scores span 69.29–72.52 and differ when the query and candidate-answer sequences are swapped. Queries and candidate answers therefore need not be encoded using guiding embeddings from the same training run to support effective matching. This goes beyond similarity in aggregate retrieval scores: different learned guiding embeddings induce output representations that remain compatible under the deployed cosine rule.

6.3 Different backbones exhibit related input-relation structure

We next investigate whether the agreement observed within a single model also extends to the representation structures of different generative models. We compare Qwen3-VL, InternVL, LLaVA, and Ministral after guiding-embedding training, using the same MMDocRAG queries and candidate-answer images.

We use linear centered kernel alignment (CKA) (Kornblith et al., 2019) to measure how similarly two models organize relationships among these inputs. Rather than comparing individual vector

Table 4: Cross-backbone relational agreement after vocabulary-initialized training. Profile denotes mean per-input cosine-profile Pearson correlation; CKA is computed on L2-normalized readouts. Q/A denote queries/candidate answers. These are matched-input structure comparisons, not cross-backbone retrieval pairings.

Backbone pair	Profile Q	Profile A	CKA Q	CKA A
Qwen3–InternVL	0.727	0.765	0.847	0.850
Qwen3–LLaVA	0.726	0.705	0.856	0.773
Qwen3–Ministral	0.763	0.767	0.852	0.826
InternVL–LLaVA	0.748	0.730	0.867	0.826
InternVL–Ministral	0.696	0.692	0.822	0.800
LLaVA–Ministral	0.702	0.693	0.808	0.741

coordinates, CKA compares the centered pairwise-similarity structures of their representations, allowing models with different output dimensions to be evaluated. Higher CKA indicates greater structural agreement. We also report cosine-profile correlation, which measures whether the models agree on each input's similarity to other inputs. Both measures are computed separately for queries and candidate answers; calculation details appear in Appendix C.

Table 4 shows substantial structural agreement across all six model pairs: query CKA ranges from 0.808 to 0.867, and candidate-answer CKA from 0.741 to 0.850. The consistently positive cosine-profile correlations support the same pattern. Thus, after guiding-embedding training, different generative models exhibit similar relationships among inputs despite their different architectures and representation dimensions. This extends our observations beyond separate runs of the same model: agreement is also present across models at the level of representation structure, without implying that their vectors can be directly mixed for retrieval.

7 Conclusions, Limitations, and Future Work

We develop a guiding-embedding approach that elicits strong retrieval capability from frozen generative models by optimizing a short sequence of continuous input embeddings shared by queries and candidate answers. Without modifying model weights or causal attention, this approach requires only thousands to tens of thousands of trainable parameters. Experiments on MMDocRAG and SciMMIR, together with direct transfer to ViDoSeek, demonstrate substantial improvements over unadapted models and performance approaching or exceeding specialized retrievers in several settings.

Further analysis shows that retrieval guidance can be both compact and transferable: a single shared embedding remains competitive for retrieval, and learned embeddings generalize across datasets without further training. Different initializations yield distinct guiding embeddings that nevertheless induce increasingly aligned representations and compatible cross-run retrieval within the same model. Different models also exhibit related representation structures. Together, these findings show that poor default representations underestimate the retrieval capability accessible through input adaptation, and that this capability is not tied to a unique guiding sequence. Such adaptation requires very few trainable parameters, although it does not eliminate the computational cost of running the generative model.

Our study has two main limitations. First, computational constraints prevent us from evaluating substantially larger generative models. Second, we cannot analyze more powerful closed-source models because the access to input embeddings and internal representations required by our method and analyses is unavailable.

Future work will investigate how model size affects the retrieval capability accessible through guiding embeddings and the number of parameters needed to elicit it. Beyond scaling, we will explore alternative training objectives and improved optimization methods to identify simpler and more effective guiding sequences. These directions aim to better translate the rich input information encoded by generative models into retrieval-effective representations, while further reducing the adaptation required.

AI USE STATEMENT

We have used AI to correct grammar, assist with writing and coding, and polish the manuscript. All code and the edited sections of the manuscript have been checked manually. We take responsibility for the final text, claims, code, figures, and artifacts.

REPRODUCIBILITY STATEMENT

The experimental setup and corresponding appendix describe the parameters and protocols used in our experiments, including training settings, checkpoint selection, candidate pools, and evaluation metrics. The supplementary material also provides the detailed implementation of the guiding-embedding training and evaluation code.

REFERENCES

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 6655–6672, 2022.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*, 2024.

Laure Ciernik, Lorenz Linhardt, Marco Morik, Jonas Dippel, Simon Kornblith, and Lukas Muttenthaler. Objective drives the consistency of representational similarity across datasets. *arXiv preprint arXiv:2411.05561*, 2024.

Kuicai Dong, CHANG YUJING, Shijie Huang, Yasheng Wang, Ruiming Tang, and Yong Liu. Benchmarking retrieval-augmented multimomal generation for document question answering. *Advances in Neural Information Processing Systems*, 38, 2026.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Colpali: Efficient document retrieval with vision language models, 2024. URL https://arxiv.org/abs/2407.01449.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 6894–6910, 2021.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20617–20642. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/huh24a.html.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models. In *Findings of the association for computational linguistics: EMNLP 2024*, pp. 3182–3196, 2024a.

Ting Jiang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. E5-v: Universal embeddings with multimodal large language models. *arXiv preprint arXiv:2407.12580*, 2024b.

Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. Vlm2vec: Training vision-language models for massive multimodal embedding tasks. *arXiv preprint arXiv:2410.05160*, 2024c.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. In *International Conference on Learning Representations*, volume 2025, pp. 79310–79333, 2025.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. 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. 4582–4597, 2021.

Sheng-Chieh Lin, Chankyu Lee, Mohammad Shoeybi, Jimmy Lin, Bryan Catanzaro, and Wei Ping. Mm-embed: Universal multimodal retrieval with multimodal llms. In *International Conference on Learning Representations*, volume 2025, pp. 44215–44234, 2025.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Jacob Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. Repetition improves language model embeddings. In *International Conference on Learning Representations*, volume 2025, pp. 93543–93579, 2025.

Nomic Team. Nomic embed multimodal: Interleaved text, image, and screenshots for visual document retrieval. *Nomic embed multimodal: Interleaved text, image, and screenshots for visual document retrieval*, 2025.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5039–5059, 2022.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 11897–11916, 2024.

Qiuchen Wang, Ruixue Ding, Zehui Chen, Weiqi Wu, Shihang Wang, Pengjun Xie, and Feng Zhao. Vidorag: Visual document retrieval-augmented generation via dynamic iterative reasoning agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9124–9145, 2025.

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. Multitask prompt tuning enables parameter-efficient transfer learning. *arXiv preprint arXiv:2303.02861*, 2023.

Siwei Wu, Yizhi Li, Kang Zhu, Ge Zhang, Yiming Liang, Kaijing Ma, Chenghao Xiao, Haoran Zhang, Bohao Yang, Wenhu Chen, et al. Scimmir: Benchmarking scientific multi-modal information retrieval. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12560–12574, 2024.

Xin Zhang, Yanzhao Zhang, Wen Xie, Mingxin Li, Ziqi Dai, Dingkun Long, Pengjun Xie, Meishan Zhang, Wenjie Li, and Min Zhang. Bridging modalities: Improving universal multimodal retrieval by multimodal large language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9274–9285. IEEE, 2025.

Kaiyan Zhao, Qiyu Wu, Zhongtao Miao, and Yoshimasa Tsuruoka. Prompt tuning can simply adapt large language models to text encoders. In *Proceedings of the 10th Workshop on Representation Learning for NLP (RepL4NLP-2025)*, pp. 38–50, 2025.

A Implementation and Protocol Details

A.1 Evaluation Metrics

Let $\mathcal{Q}$ be the evaluation queries, $\mathcal{P}(q)$ the nonempty set of relevant candidate answers for query q, and $\mathrm{Top}_k(q)$ its top-k retrieved candidates. Recall is macro-averaged over queries:

$$\mathrm{R@}k = \frac{100}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{|\,\mathrm{Top}_k(q) \cap \mathcal{P}(q)|}{|\mathcal{P}(q)|}. \tag{5}$$

For multi-positive queries, this measures the fraction of relevant items retrieved, rather than whether any positive appears. Let $\rho(q)$ be the rank of the first relevant candidate, with $\rho(q) = \infty$ if none is retrieved. Truncated mean reciprocal rank is

$$\mathrm{MRR@20} = \frac{100}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \begin{cases} 1/\rho(q), & \rho(q) \leqslant 20, \\ 0, & \rho(q) > 20. \end{cases} \tag{6}$$

Both retrieval metrics are reported as percentages; representation cosines are not. Cross-run retrieval performance in Section 3.3 is evaluated using MRR@20.

A.2 Implementation Details

Fixed prompt. We use `Summarize the content above in one word:` as the fixed summarization prompt. The Qwen3-VL-4B tokenizer splits this instruction into ten tokens:

```
Sum | mar | ize | the | content | above | in | one | word | :
```

The fixed-prompt baseline uses their original vocabulary embeddings without training. Summarization-initialized guiding embeddings start from these same vectors and are subsequently optimized. For prompt-derived length ablations, we remove selected tokens while preserving the remaining token IDs and their original leading spaces, rather than retokenizing the shortened text.

Training records. Training uses AdamW with learning rate 10^{-3}, zero weight decay, temperature 0.05, query batch size four, and up to eight negatives per query. The inspected trainer uses BF16, scaled dot-product attention, clipping at 1.0, image microbatch size one, and recomputation/gradient caching. All positives and sampled negatives enter the masked loss. The workbooks record 2,500-step main budgets and source-selected endpoints. Validation is recorded at step zero and every 100 steps. Checkpoint selection maximizes R@1, then R@3. Exact ties retain the earlier checkpoint.

B Backbone Configurations

The five generative models use the same MMDocRAG split, objective, and 2,500-step training budget, while retaining native templates, visual preprocessing, and readout boundaries. No-prompt and fixed-prompt controls use the same chat/readout implementation as their adapted model.

The exact checkpoints are Qwen3-VL-4B-Instruct, LLaVA-OneVision-1.5-4B-Instruct, InternVL3.5-4B, Ministral-3-3B-Instruct-2512-BF16, and Qwen3.5-9B. Qwen3-VL, LLaVA, and InternVL have width 2,560; Ministral has width 3,072 and Qwen3.5 width 4,096.

Qwen3-VL, LLaVA, and Qwen3.5 use visual budgets of 1,800; InternVL uses 1,792 with 448-pixel dynamic tiles (up to six plus a thumbnail); Ministral uses 1,764 with its Pixtral processor and a 1,176-pixel longest edge. LLaVA retains its Qwen2.5-VL processor and Rice visual tower, InternVL its native Chinese system template, and Ministral its native English system template. Computation uses BF16 and SDPA, with Qwen3.5's native linear attention using the FLA fast kernel.

C Details of Cross-Model Representation Comparison

This section supplements Section 6.3 by detailing how we compare the relationships among inputs encoded by different generative models. We first describe the cross-model measures used in Table 4, then report a separate within-model similarity analysis.

Experimental setting. We use vocabulary-initialized guiding embeddings trained separately for Qwen3-VL, InternVL, LLaVA, and Ministral, with selected checkpoints at steps 2,300, 2,500, 2,000, and 2,000, respectively. Comparisons are performed separately for queries and candidate answers, using the same input order across models.

Cross-model representation structure. The question is whether different models agree on which inputs are similar, rather than whether their output vectors share the same coordinates. For a set of n inputs, let Z_b contain the L2-normalized final representations produced by model b, one input per row. The matrix $K_b = Z_b Z_b^\top$ then records the cosine similarity between every pair of inputs. With the centering matrix $H = I - \mathbf{1}\mathbf{1}^\top/n$, linear CKA compares these similarity structures as

$$\mathrm{CKA}(b, c) = \frac{\langle HK_bH, HK_cH\rangle_F}{\|HK_bH\|_F \, \|HK_cH\|_F}. \tag{7}$$

Here $\langle\cdot,\cdot\rangle_F$ and $\|\cdot\|_F$ denote the Frobenius inner product and norm. CKA uses the full similarity matrices, including their diagonals. For cosine-profile correlation, we instead take each input's similarities to all other inputs, excluding itself, and compute the Pearson correlation between these lists from the two models. We report the average correlation over inputs. Both measures compare relationships among the same inputs and therefore allow different representation dimensions without fitting an alignment map.

D Detailed Training Trajectories

Table 5 provides the numerical values underlying Figure 1. At each of the five training stages, it summarizes retrieval with same-run and cross-run encodings, together with same-input similarity across runs and different-input similarity within runs. S, G, and V denote summarization, Gaussian, and random-vocabulary initialization, respectively. The results show increasing same-input agreement and a narrowing cross-run retrieval gap, while different-input similarity generally decreases; individual stages need not change monotonically.

Table 5: Training-stage summaries. MRR denotes MRR@20. Original and Mixed average the three same-run and six cross-run retrieval pairings, respectively; Gap is Original minus Mixed. Q/I denote queries/candidate-answer images. Same denotes similarity between representations of the same input across runs; Different denotes similarity between representations of different inputs within a run.

Progress	Original MRR	Mixed MRR	Gap	Same Q	Same I	Different Q	Different I
20%	63.85	58.63	5.22	0.529	0.559	0.447	0.445
40%	68.96	68.59	0.38	0.788	0.777	0.354	0.333
60%	70.56	70.21	0.35	0.819	0.832	0.333	0.311
80%	71.60	71.37	0.23	0.855	0.796	0.293	0.296
100%	71.65	71.23	0.42	0.863	0.824	0.311	0.278

Table 6 expands the average similarities into individual comparisons, supplementing Section 6.1. Rows such as S–G compare representations of the same inputs across two independently trained guiding sequences, whereas rows such as S compare different inputs encoded with one guiding sequence. Reporting queries and candidate answers separately shows how the aggregate trends vary across initialization pairs and input types, including the non-monotonic changes obscured by the averages.

Table 6: Individual cosine-similarity comparisons at each training stage. S/G/V denote summarization/Gaussian/random-vocabulary initialization. Image denotes candidate-answer images.

Pair / within-run control	20%	40%	60%	80%	100%
Query					
S–G, same input	0.391	0.743	0.798	0.864	0.855
S–V, same input	0.711	0.847	0.821	0.818	0.873
G–V, same input	0.487	0.774	0.838	0.883	0.862
S, different inputs	0.475	0.367	0.360	0.321	0.336
G, different inputs	0.575	0.383	0.356	0.267	0.300
V, different inputs	0.292	0.312	0.282	0.291	0.298
Image					
S–G, same input	0.491	0.788	0.830	0.764	0.821
S–V, same input	0.774	0.803	0.829	0.858	0.839
G–V, same input	0.412	0.741	0.837	0.766	0.812
S, different inputs	0.402	0.366	0.309	0.315	0.289
G, different inputs	0.570	0.319	0.342	0.299	0.275
V, different inputs	0.364	0.315	0.281	0.274	0.269

Complete retrieval metrics for all nine query–candidate answer pairings at every training stage are provided in Appendix I, Tables 17–21. These include the MRR values underlying the cross-run compatibility analysis in Section 6.2 and Figure 2, alongside all reported recall cutoffs.

E Additional Cross-Dataset Transfer Results

We further evaluate cross-dataset transfer between MMDocRAG and SciMMIR using Qwen3-VL-4B and ten shared guiding embeddings. In both directions, embeddings learned on the source dataset are directly reused for retrieval on the target dataset without further training. These experiments complement the MMDocRAG-to-ViDoSeek results in Section 5.1.

Table 7 reports retrieval on SciMMIR using MMDocRAG-trained guiding embeddings from summarization, Gaussian, and random-vocabulary initialization, alongside fixed-prompt and SciMMIR-specific training results. All three transferred sequences substantially outperform the fixed prompt, although in-domain training remains stronger. Their similar transfer performance further shows that cross-dataset guidance does not require meaningful initialization.

Table 7: Direct transfer from MMDocRAG to SciMMIR, compared with a fixed summarization prompt and SciMMIR-specific training. All results are evaluated on SciMMIR. S/G/V denote summarization/Gaussian/random-vocabulary initialization. MRR denotes MRR@20.

Configuration	R@1	MRR
Fixed summarization prompt	19.10	25.55
MMDocRAG-trained, frozen transfer (S)	35.24	44.50
MMDocRAG-trained, frozen transfer (G)	34.80	44.03
MMDocRAG-trained, frozen transfer (V)	36.06	45.32
SciMMIR-trained, in-domain reference	41.27	51.20

Table 8 reports the reverse transfer from SciMMIR to MMDocRAG, alongside fixed-prompt and in-domain training results. Directly reusing SciMMIR-trained embeddings improves both R@1 and MRR over the fixed prompt, although MMDocRAG-specific training remains stronger. Together, the two transfer directions support the same conclusion as ViDoSeek: learned guiding embeddings provide transferable retrieval guidance, while target-specific adaptation can further improve performance. Complete metrics for both directions are reported in Tables 11 and 12.

Table 8: Direct transfer from SciMMIR to MMDocRAG, compared with a fixed summarization prompt and MMDocRAG-specific training. All configurations use ten embeddings. MRR denotes MRR@20.

Configuration	R@1	MRR
Fixed summarization prompt	28.74	52.09
SciMMIR-trained, frozen transfer	41.76	65.64
MMDocRAG-trained, in-domain reference	45.10	70.79

F Additional Single-Embedding Results

Table 9: Individual reduced-token results with in-domain training on Qwen3-VL-4B. Ten shared embeddings require 25,600 parameters in total; one requires 2,560. Random vocabulary single-embedding rows use different IDs with the same training seed. MRR cutoff is 20.

Configuration	Embeddings	MMDocRAG MRR	SciMMIR MRR
Summarization, trained	10	70.79	51.20
`content`, fixed	1	21.13	0.15
`content`, trained	1	71.76	44.25
Gaussian, trained	1	68.82	—
Vocabulary ID 123652, trained	1	69.34	—
Vocabulary ID 100728, trained	1	66.25	—
Vocabulary ID 118676, trained	1	69.44	—

Table 9 gives the individual reduced-token results, supplementing the vocabulary-length comparison in Table 3. Single-`content` MMDocRAG-to-ViDoSeek transfer reaches 73.76 MRR, below ten-token transfer at 76.20.

G Full Metrics for the Main Retrieval Comparisons

Table 10 expands Table 1 with all evaluated recall cutoffs. All metrics are percentages, and MRR denotes MRR@20 throughout these tables.

Table 10: Complete metrics for the main retrieval comparisons. MRR denotes MRR@20. Our MMDocRAG and SciMMIR results use in-domain training; ViDoSeek uses direct transfer from MMDocRAG. Settings and baseline scores match Table 1.

Configuration	R@1	R@3	R@5	R@10	R@20	MRR
MMDocRAG						
No added prompt	7.03	19.75	29.68	47.83	67.29	22.22
Fixed summarization	28.74	50.91	60.75	74.80	86.19	52.09
Guiding embeddings (ours)	45.10	68.93	76.67	86.81	94.50	70.79
ColQwen2.5-3B	47.51	68.85	76.01	85.42	93.36	71.69
Nomic-3B	47.30	68.90	77.35	86.64	93.84	72.23
Nomic-7B	48.25	70.92	78.28	87.33	94.02	73.64
ColNomic-7B	48.78	69.65	77.29	86.70	93.39	73.07
SciMMIR						
No added prompt	0.02	0.10	0.18	0.34	0.66	0.11
Fixed summarization	19.10	28.59	33.00	39.28	45.37	25.55
Guiding embeddings (ours)	41.27	57.49	63.44	70.73	76.50	51.20
ColQwen2.5-3B	40.93	55.73	60.59	65.76	70.12	49.63
Nomic-3B	44.17	58.88	63.88	68.95	73.33	52.89
Nomic-7B	46.44	61.92	66.92	71.99	75.94	55.48
ColNomic-7B	43.48	58.87	63.85	68.89	72.85	52.49
ViDoSeek						
No added prompt	0.18	1.05	1.40	2.19	3.77	0.79
Fixed summarization	36.69	54.38	63.22	70.93	79.16	48.04
Guiding embeddings (ours)	65.67	84.41	90.46	95.36	96.94	76.20
ColQwen2.5-3B	75.22	91.68	95.01	97.72	99.21	83.92
Nomic-3B	71.72	89.14	94.48	97.81	99.12	81.36
Nomic-7B	71.28	89.14	94.57	97.72	99.12	81.16
ColNomic-7B	73.47	89.75	93.61	97.64	99.21	82.22

Table 11: Complete metrics for MMDocRAG-to-SciMMIR transfer and its controls. MRR denotes MRR@20. S/G/V denote summarization, Gaussian, and vocabulary initialization. Transferred embeddings receive no target-dataset training.

SciMMIR configuration	R@1	R@3	R@5	R@10	R@20	MRR
Fixed summarization	19.10	28.59	33.00	39.28	45.37	25.55
In-domain summarization	41.27	57.49	63.44	70.73	76.50	51.20
Transferred S	35.24	50.51	56.04	62.61	68.12	44.50
Transferred G	34.80	49.55	55.46	62.67	68.16	44.03
Transferred V	36.06	50.85	56.91	63.66	69.29	45.32

Table 12: Complete metrics for direct transfer from SciMMIR to MMDocRAG and its fixed-prompt and in-domain training controls, corresponding to Table 8. MRR denotes MRR@20.

MMDocRAG configuration	R@1	R@3	R@5	R@10	R@20	MRR
Fixed summarization	28.74	50.91	60.75	74.80	86.19	52.09
SciMMIR-trained transfer	41.76	60.21	68.64	80.54	94.32	65.64
MMDocRAG-trained reference	45.10	68.93	76.67	86.81	94.50	70.79

H Full Metrics for Model and Length Experiments

The following tables expand the cross-model and sequence-length experiments in Sections 5.2 and 5.3. They retain the same runs and checkpoints as the main tables.

Table 13: Complete MMDocRAG metrics for Table 2. MRR denotes MRR@20. `None` and `Fixed S` are frozen controls; `G`, `V`, and `S` are trained Gaussian, vocabulary, and summarization initializations.

Setting	R@1	R@3	R@5	R@10	R@20	MRR
Qwen3-VL-4B						
None	7.03	19.75	29.68	47.83	67.29	22.22
Fixed S	28.74	50.91	60.75	74.80	86.19	52.09
G	45.30	68.24	77.73	87.78	94.50	70.95
V	47.60	69.12	78.85	88.16	94.80	73.21
S	45.10	68.93	76.67	86.81	94.50	70.79
LLaVA-OV1.5-4B						
None	4.72	13.79	22.95	39.15	58.11	17.47
Fixed S	18.78	35.17	44.70	59.26	73.89	37.89
G	35.46	60.10	70.10	83.62	93.05	61.40
V	40.47	63.62	74.16	84.73	93.55	66.31
S	36.94	60.43	71.43	83.95	93.01	62.72
InternVL3.5-4B						
None	11.54	25.68	36.00	53.79	72.09	29.73
Fixed S	19.56	37.76	48.19	63.57	78.38	40.50
G	39.84	63.42	73.35	84.52	92.93	65.38
V	41.91	65.14	74.80	86.40	93.77	67.69
S	37.14	60.76	70.94	83.65	92.22	63.17
Ministral-3-3B						
None	4.74	15.70	24.92	42.35	61.37	19.04
Fixed S	15.63	33.71	44.63	61.94	78.69	36.47
G	46.06	68.69	77.60	88.27	94.66	72.16
V	49.01	72.25	80.67	90.26	95.46	75.21
S	48.80	72.00	80.46	89.31	94.98	74.77
Qwen3.5-9B						
None	4.71	15.08	25.29	44.66	63.54	19.33
Fixed S	4.88	14.67	22.57	39.99	58.31	17.86
G	48.95	73.28	81.90	90.98	96.34	75.32
V	47.35	70.97	79.42	89.89	95.72	73.88
S	45.98	69.80	77.75	88.47	95.12	72.20

Table 14: Complete MMDocRAG metrics for the individual length configurations in Table 3. MRR denotes MRR@20. Each row uses the same initialization and single run as its main-table counterpart.

Embeddings	R@1	R@3	R@5	R@10	R@20	MRR
1	42.96	66.87	76.48	87.07	94.54	69.34
3	42.61	66.90	75.56	86.51	93.85	68.66
5	44.95	68.61	78.25	88.10	94.81	71.30
7	45.70	68.97	77.56	88.06	94.66	71.60
10	47.60	69.12	78.85	88.16	94.80	73.21
15	47.96	70.50	79.28	88.96	95.24	73.73
20	44.25	68.13	77.25	87.32	94.43	70.47

Table 15: Complete metrics for the single-embedding experiments supporting Section 5.3 and Appendix F. MRR denotes MRR@20. Trained configurations use the respective dataset for training; vocabulary IDs identify distinct one-token initializations.

Configuration	R@1	R@3	R@5	R@10	R@20	MRR
MMDocRAG: fixed single content	6.82	17.82	26.74	46.74	67.27	21.13
MMDocRAG: trained single content	45.63	68.31	78.19	88.29	94.91	71.76
MMDocRAG: Gaussian single	42.77	65.65	75.20	85.51	93.85	68.82
MMDocRAG: vocabulary ID 123652	42.96	66.87	76.48	87.07	94.54	69.34
MMDocRAG: vocabulary ID 100728	39.47	64.47	74.34	85.62	93.81	66.25
MMDocRAG: vocabulary ID 118676	42.92	66.73	76.38	86.94	94.57	69.44
SciMMIR: fixed single content	0.07	0.13	0.20	0.36	0.58	0.15
SciMMIR: trained single content	34.45	50.21	56.34	63.71	69.97	44.25

I Full Metrics for Cross-Run Retrieval

Table 16 expands the training-progress comparison in Figure 1(c). The subsequent tables report every query–candidate answer pairing at every measured stage, including the two stages visualized in Figure 2. S/G/V denote the summarization-, Gaussian-, and vocabulary-initialized runs. The first letter selects query guiding embeddings and the second selects candidate-answer guiding embeddings. Progress is relative to each run's validation-selected checkpoint. The recall cutoffs show the same broad improvement and increasing cross-run compatibility as MRR.

Table 16: Mean retrieval metrics over the three same-run or six cross-run pairings. MRR denotes MRR@20. Each mean is computed from the corresponding individual results below.

Progress	Pairing	R@1	R@3	R@5	R@10	R@20	MRR
20%	Same-run	38.13	61.53	71.69	83.59	92.37	63.85
20%	Cross-run	32.96	56.60	67.75	81.19	91.26	58.63
40%	Same-run	42.87	66.64	75.98	86.41	93.78	68.96
40%	Cross-run	42.44	66.02	75.59	86.15	93.80	68.59
60%	Same-run	44.46	67.38	77.34	87.48	94.41	70.56
60%	Cross-run	44.11	67.44	76.64	87.26	94.44	70.21
80%	Same-run	45.75	68.33	77.47	87.52	94.63	71.60
80%	Cross-run	45.50	68.02	77.59	87.41	94.58	71.37
100%	Same-run	46.00	68.77	77.75	87.58	94.60	71.65
100%	Cross-run	45.42	68.73	77.67	87.36	94.70	71.23

Table 17: Complete cross-run retrieval metrics at 20% progress. MRR denotes MRR@20. Same-run references appear above the separator; cross-run tests appear below it.

Query/answer run	R@1	R@3	R@5	R@10	R@20	MRR
S/S	35.04	60.61	71.06	83.62	92.77	61.37
G/G	35.36	57.44	67.65	79.99	90.67	60.36
V/V	43.98	66.54	76.36	87.16	93.65	69.82
S/G	24.68	47.14	59.45	75.28	87.60	49.01
S/V	37.21	60.57	71.69	84.06	93.12	63.11
G/S	29.82	53.19	64.94	78.86	89.92	55.25
G/V	33.68	56.12	67.38	81.35	91.50	58.88
V/S	38.46	63.53	73.94	85.34	93.73	65.11
V/G	33.92	59.02	69.09	82.24	91.70	60.39

Table 18: Complete cross-run retrieval metrics at 40% progress. MRR denotes MRR@20.

Query/answer run	R@1	R@3	R@5	R@10	R@20	MRR
S/S	42.76	66.79	75.20	85.84	93.03	68.58
G/G	42.32	65.41	75.72	86.32	93.84	68.21
V/V	43.52	67.73	77.02	87.07	94.46	70.11
S/G	41.69	64.23	74.30	85.38	93.64	67.20
S/V	41.99	65.67	75.09	85.45	93.09	67.96
G/S	42.32	66.35	75.91	86.27	93.94	68.69
G/V	40.94	65.60	75.17	86.24	93.74	67.58
V/S	44.21	67.42	76.34	86.69	94.09	70.33
V/G	43.51	66.84	76.73	86.86	94.29	69.75

Table 19: Complete cross-run retrieval metrics at 60% progress. MRR denotes MRR@20.

Query/answer run	R@1	R@3	R@5	R@10	R@20	MRR
S/S	43.13	67.25	76.79	87.33	94.22	69.37
G/G	43.77	66.88	77.28	87.44	94.16	69.97
V/V	46.48	68.02	77.96	87.68	94.84	72.34
S/G	43.34	66.40	76.05	86.74	94.09	69.47
S/V	43.48	66.96	75.52	86.88	94.07	69.48
G/S	43.31	67.55	77.04	87.49	94.71	69.59
G/V	44.78	67.99	76.90	87.65	94.59	70.66
V/S	44.48	68.16	77.22	87.45	94.91	70.80
V/G	45.24	67.57	77.10	87.36	94.31	71.28

Table 20: Complete cross-run retrieval metrics at 80% progress. MRR denotes MRR@20.

Query/answer run	R@1	R@3	R@5	R@10	R@20	MRR
S/S	45.69	68.72	77.77	87.90	94.69	71.71
G/G	44.56	67.43	76.97	86.98	94.41	70.42
V/V	47.01	68.84	77.68	87.68	94.80	72.66
S/G	43.84	67.04	77.24	87.04	93.94	70.01
S/V	46.06	68.59	77.96	87.55	94.40	71.79
G/S	45.49	68.06	77.50	87.31	94.69	71.37
G/V	46.41	68.34	77.47	87.96	94.71	72.24
V/S	46.07	68.73	78.15	87.67	95.10	72.00
V/G	45.13	67.35	77.25	86.93	94.63	70.82

Table 21: Complete cross-run retrieval metrics at 100% progress, corresponding to each run's validation-selected checkpoint. MRR denotes MRR@20.

Query/answer run	R@1	R@3	R@5	R@10	R@20	MRR
S/S	45.10	68.93	76.67	86.81	94.50	70.79
G/G	45.30	68.24	77.73	87.78	94.50	70.95
V/V	47.60	69.12	78.85	88.16	94.80	73.21
S/G	43.55	67.78	76.74	86.34	94.36	69.29
S/V	46.74	69.18	78.13	87.83	94.88	72.43
G/S	45.93	69.75	78.38	87.39	94.96	71.78
G/V	46.34	69.75	79.10	88.30	94.91	72.52
V/S	45.69	68.82	77.31	87.37	94.66	71.32
V/G	44.28	67.07	76.35	86.91	94.43	70.05