LIGHT-DIFFRETRIEVER: BUDGET-FLEXIBLE RETRIEVAL WITH DIFFUSION LANGUAGE MODELS VIA MATRYOSHKA REPRESENTATIVE TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models produce multiple retrieval vectors in one forward pass, but fixed-budget training does not supervise effectiveness under truncation. We introduce LIGHT-DIFFRETRIEVER, a budget-flexible extension of DiffRetriever (Wang et al., 2026) that learns Matryoshka representative tokens: ordered vector prefixes supporting different query and passage budgets from one checkpoint. Learnable slot offsets with decaying-scale initialization introduce an order-dependent bias; curriculum learning gradually adds prefix contrastive losses to the full-budget objective. Post-encoding truncation reduces dense storage and matching cost without retraining or corpus re-encoding. On Dream and LLaDA-1.5, reducing the passage budget from 16 to 4 vectors retains 98.2% and 98.0% of full-budget MS MARCO MRR@10, and 98.4% and 97.8% of BEIR-7 nDCG@10, respectively, with one quarter of the dense-vector storage. These results support nested representative-token learning for resource-adaptive retrieval.

1 INTRODUCTION

Neural retrieval must balance semantic coverage against the cost of storing and searching representations. Single-vector retrievers compress each passage into one embedding, enabling efficient similarity search, whereas multi-vector retrievers preserve finer-grained evidence through late interaction (Karpukhin et al., 2020; Khattab & Zaharia, 2020). This trade-off becomes consequential when a retriever serves collections or deployments with different memory and search-compute constraints: a representation that is effective at one budget need not remain effective after compression. The resulting objective is to preserve retrieval effectiveness across deployment-specific representation budgets, rather than optimize exclusively for a single encoding capacity.

Large language models have strengthened text embeddings through their pretrained knowledge and retrieval-oriented adaptation (Ma et al., 2024; Wang et al., 2024; Lee et al., 2025; Zhang et al., 2025c). Diffusion language models (DLMs) offer a complementary interface, with bidirectional context processing learned through masked-token prediction (Ye et al., 2025; Nie et al., 2025). DiffEmbed and pplx-embed exploit this context for pooled embeddings (Zhang et al., 2025b; Eslami et al., 2026); DiffRetriever instead appends masked representative positions and obtains multiple retrieval vectors jointly in one forward pass (Wang et al., 2026). This avoids sequential autoregressive generation of representative tokens and makes compact multi-vector retrieval natural for DLMs. Parallel vectors still incur downstream costs: storage grows with the passage budget, and exhaustive late interaction with the product of query and passage budgets. Figure 1 contrasts these interfaces: single-vector compression can limit fine-grained matching (a), while fixed-budget multi-vector training does not directly optimize retrieval at reduced budgets (b).

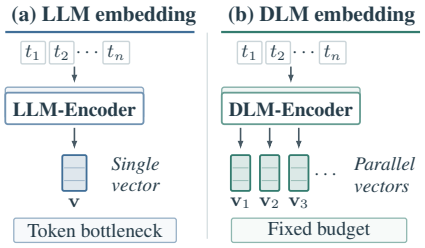


Figure 1: **(a)** A single-vector LLM embedding; **(b)** parallel DLM representatives trained at a fixed budget.

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.

From fixed capacity to nested representations. DiffRetriever optimizes a chosen pair of query and passage representation counts. Its full-budget matching objective is invariant to permutations of the output vectors and does not explicitly supervise the retrieval effectiveness of individual prefixes. Consequently, discriminative information may be distributed across positions that are excluded under a reduced deployment budget. Changing the number of masks before encoding is also different from truncating the encoded vectors: under bidirectional attention, the remaining representations can depend on the masks that were removed. This motivates a representation that is encoded at a fixed capacity and supports *nested, retrieval-effective subsets* across downstream budgets.

We address this problem with LIGHT-DIFFRETRIEVER, which extends DiffRetriever with Matryoshka learning along the *representative-token axis*, rather than the embedding-coordinate axis of conventional Matryoshka representation learning (Kusupati et al., 2022). The method couples three design choices. First, learnable slot-specific offsets parameterize representative positions in addition to the backbone's positional encoding. Second, linearly decaying initialization introduces an order-dependent bias in offset magnitude without prescribing a semantic decomposition. Third, curriculum learning progressively incorporates contrastive losses on nested query–passage prefixes after an initial full-budget optimization phase. This formulation jointly supervises multiple retained-vector budgets without budget-specific models or query-dependent routing.

We instantiate LIGHT-DIFFRETRIEVER with Dream (Ye et al., 2025) and LLaDA-1.5 (Zhu et al., 2026) at fixed maximum capacity and evaluate nested budgets on MS MARCO, TREC DL19/20, and seven BEIR datasets. At $(4, 4)$, the dense passage representation occupies one quarter of the full-budget storage while retaining about 98% of full-budget aggregate retrieval quality on MS MARCO and BEIR-7. We evaluate truncation robustness, distinguishing storage and matching savings from encoding costs; full-budget accuracy gains and encoding speedups are not assumed.

Our contributions are summarized as follows:

- **Budget-flexible Retrieval.** We formulate diffusion retrieval through nested representative-token prefixes, decoupling maximum encoding capacity from the number of vectors retained for matching.
- **Order-aware Optimization.** We develop a training framework that combines learnable slot offsets, decaying-scale initialization, and curriculum prefix-level contrastive supervision.
- **Empirical Experiment.** We conduct experiments on two DLM backbones, demonstrating a fourfold reduction in vector storage while retaining nearly 99% of full-budget performance.

2 RELATED WORK

2.1 LLM-BASED AND DLM-BASED RETRIEVAL

Neural text retrieval evolved from pretrained bidirectional encoders, including Sentence-BERT and DPR (Reimers & Gurevych, 2019; Karpukhin et al., 2020), through contrastive learning in SimCSE and Contriever (Gao et al., 2021b; Izacard et al., 2022), to general-purpose embedding models such as E5 and GTE (Wang et al., 2022; Li et al., 2023). LLM-based approaches subsequently leveraged larger backbones: RepLLaMA and E5-Mistral adapt autoregressive models for retrieval (Ma et al., 2024; Wang et al., 2024), while GritLM, NV-Embed, and Qwen3-Embedding advance joint generative–representational training, pooling, and embedding-oriented adaptation (Muennighoff et al., 2025; Lee et al., 2025; Zhang et al., 2025c). LLM2Vec and Echo address restricted contextual access through bidirectional adaptation and input repetition, respectively (BehnamGhader et al., 2024; Springer et al., 2025); PromptReps derives dense and sparse representations from prompted token predictions (Zhuang et al., 2024). Diffusion backbones, exemplified by Dream and LLaDA, instead provide bidirectional masked-token modeling directly (Ye et al., 2025; Nie et al., 2025). DiffEmbed studies pooled DLM embeddings and reasoning-oriented contrastive adaptation (Zhang et al., 2025b); pplx-embed combines diffusion pretraining with quantization-aware dense embeddings and contextualized chunk representations (Eslami et al., 2026); DiffuRank applies diffusion likelihood to reranking (Liu et al., 2026). DiffRetriever most directly establishes parallel representative-token encoding and late interaction (Wang et al., 2026), which LIGHT-DIFFRETRIEVER extends with nested-prefix supervision to accommodate post-encoding retrieval budgets.

2.2 Multi-vector Retrieval

ColBERT established contextualized token-level late interaction with `MaxSim`, while ColBERTv2 and PLAID reduced storage and search costs through residual compression and specialized indexing (Khattab & Zaharia, 2020; Santhanam et al., 2022b;a). COIL restricts contextualized interactions to lexical matches, and BGE-M3 unifies dense, sparse, and multi-vector retrieval (Gao et al., 2021a; Chen et al., 2024). XTR improves token retrieval within the late-interaction pipeline, whereas MUVERA maps multi-vector similarity search to fixed-dimensional encodings (Lee et al., 2023; Dhulipala et al., 2024). Poly-encoders provide a complementary approach based on compact learned context representations (Humeau et al., 2020). These methods primarily optimize representation granularity, compression, or search; Light-DiffRetriever instead supervises nested subsets of DLM-produced representative vectors, providing a budget-adaptive representation that is complementary to quantization and efficient indexing.

2.3 Matryoshka Representation Learning

Matryoshka representation learning (MRL) jointly supervises nested coordinate prefixes of a single embedding, enabling deployment at multiple dimensionalities (Kusupati et al., 2022). Subsequent work extends this principle along complementary axes: 2D Matryoshka sentence embeddings jointly consider encoder depth and embedding dimension (Li et al., 2024), while Matryoshka-Adaptor learns dimension-reducing transformations of pretrained embeddings in supervised and unsupervised settings (Yoon et al., 2024). Retrieval-specific developments include 2D Matryoshka training for information retrieval and the improved depth–dimension training formulation of Starbucks (Wang et al., 2025; Zhuang et al., 2026). SMEC further investigates sequential Matryoshka optimization, adaptive dimension selection, and cross-batch supervision for embedding compression (Zhang et al., 2025a). Dimension-wise MRL also appears in the contextual-embedding training of pplx-embed (Eslami et al., 2026). Beyond coordinates and depth, MQT supervises latent-query prefixes for visual-token reduction, and MetaEmbed learns nested retrieval-token groups (Hu et al., 2024; Xiao et al., 2025). Light-DiffRetriever studies this *representative-vector-count* axis for DLM retrieval: every retained vector preserves its hidden dimension, and curriculum late-interaction objectives supervise ordered query and passage prefixes. An extended discussion of diffusion LLMs, textual and multimodal multi-vector retrieval, and alternative Matryoshka nesting axes appears in Appendix A.

3 Preliminary Experiment

Before introducing nested-prefix training, we measured fixed-budget DiffRetriever across the available query–passage capacities. We use the equal-weight ten-task mean only for capacity selection; task-specific metrics follow. As shown in Figure 2, $(K_q, K_p) = (4, 16)$ attains the highest score, consistent with DiffRetriever's observation that parallel representative tokens benefit from a smaller query bank and a larger passage bank (Wang et al., 2026). We therefore use this capacity as the full-budget reference and vary only retained prefixes at deployment, allowing one encoder and cached corpus representation to serve heterogeneous budgets rather than training a separate model for each.

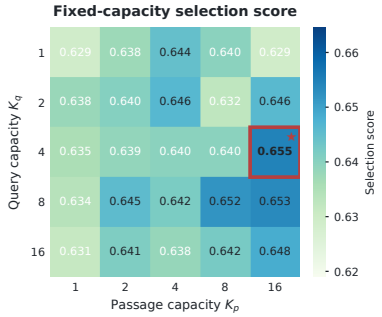


Figure 2: Capacity-selection landscape for fixed-budget Dream configurations.

4 Methodology

Light-DiffRetriever decouples encoding capacity from the number of representative vectors retained for retrieval. Figure 3 follows three components: order-aware encoding (a), nested contrastive learning (b), and budget-flexible inference (c). All losses reuse the same encoded banks. Algorithm 1 gives the complete procedure; section 4.4 analyzes its representation and ranking properties.

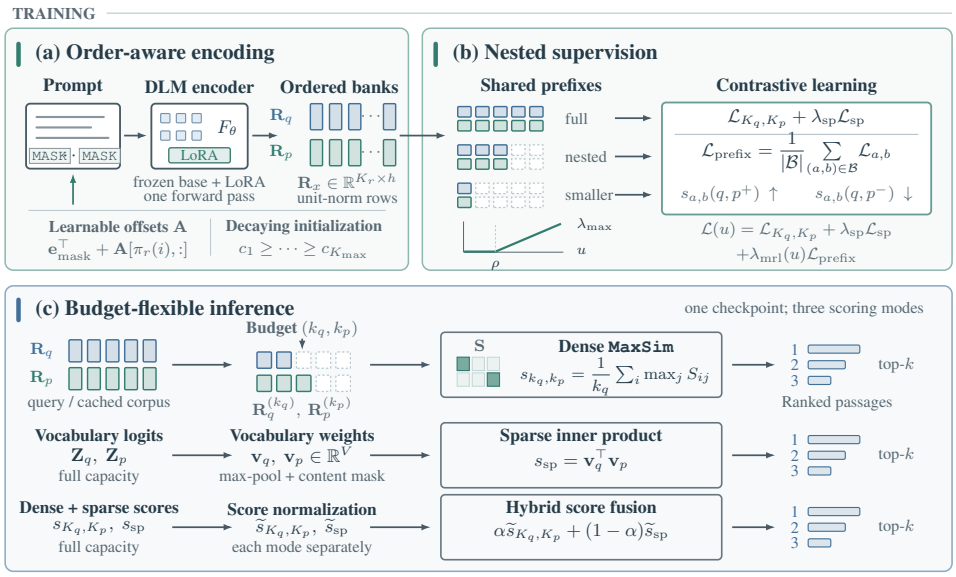


Figure 3: Training: **(a)** slot-aware DLM encoding and **(b)** curriculum prefix supervision. Inference: **(c)** budget-selected dense matching or full-capacity sparse/hybrid scoring. Blue/green denote query/passage banks; dashed slots are discarded. Counts are schematic.

4.1 Order-aware Representative Encoding

Let q denote a query, $p \in \mathcal{C}$ a passage in corpus $\mathcal{C}$, and $r \in \{q, p\}$ the input role. The maximum numbers of representative positions are K_q and K_p, with $K_{\max} = \max(K_q, K_p)$. For an input x of role r, we construct a prompted sequence of length L_x containing an instruction, the text, an assistant prefix, K_r consecutive `[MASK]` positions, and fixed closing tokens. Let $\mathcal{G}_x = (g_{x,1}, \ldots, g_{x,K_r})$ index those positions. A bidirectional DLM F_θ produces

$$\mathbf{H}_x = F_\theta(\mathbf{U}_x) \in \mathbb{R}^{L_x \times h}, \qquad \mathbf{R}_x[i,:] = \frac{\mathbf{H}_x[g_{x,i},:]}{\|\mathbf{H}_x[g_{x,i},:]\|_2}, \qquad \mathbf{R}_x \in \mathbb{R}^{K_r \times h}, \tag{1}$$

where $\mathbf{U}_x \in \mathbb{R}^{L_x \times h}$ is the input embedding matrix and h the hidden dimension. All representative positions share one forward pass, without iterative denoising or token generation.

Learnable slot offsets. We introduce a trainable offset table $\mathbf{A} \in \mathbb{R}^{K_{\max} \times h}$ shared by both roles. The embedding at representative position i is

$$\mathbf{U}_x[g_{x,i},:] = \mathbf{e}_{\text{mask}}^\top + \mathbf{A}[\pi_r(i),:], \qquad \pi_r(i) = K_{\max} - K_r + i, \tag{2}$$

where $\mathbf{e}_{\text{mask}} \in \mathbb{R}^h$ is the backbone mask embedding. The role-dependent map aligns each offset table to the structural tail of the prompt. These offsets augment, rather than replace, the backbone positional encoding and make the representative slots distinguishable before any prefix is selected.

Decaying-scale initialization. We initialize independent Gaussian rows with ordered scales:

$$\mathbf{A}[j,:] = c_j \boldsymbol{\epsilon}_j^\top, \qquad \boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_h), \qquad c_1 \geq c_2 \geq \cdots \geq c_{K_{\max}} > 0. \tag{3}$$

The $K_{\max} h$ added parameters have ordered initial expected norms, $\mathbb{E}\|\mathbf{A}[j,:]\|_2^2 = h\sigma^2 c_j^2$. This is an initialization bias, not a constraint on trained norms or vector semantics (Proposition C.5).

4.2 Nested Matryoshka Contrastive Learning

Budget-conditioned dense scoring. For a retained budget (a, b) with $1 \leq a \leq K_q$ and $1 \leq b \leq K_p$, define $\mathbf{R}_q^{(a)} = \mathbf{R}_q[1:a,:]$ and $\mathbf{R}_p^{(b)} = \mathbf{R}_p[1:b,:]$. Their pairwise similarity matrix is

$\mathbf{S} = \mathbf{R}_q^{(a)}(\mathbf{R}_p^{(b)})^\top \in \mathbb{R}^{a\times b}$. We score a query–passage pair with query-averaged `MaxSim`:

$$s_{a,b}(q,p) = \frac{1}{a}\sum_{i=1}^{a} \max_{1\le j\le b} \mathbf{R}_q[i,:]\mathbf{R}_p[j,:]^\top. \tag{4}$$

Unit-norm rows give cosine similarities; one vector per side recovers single-vector scoring. Explicit prefix supervision addresses the ordering ambiguity analyzed in section 4.4.

Nested contrastive supervision. For a batch of B queries, let p_n^+ be the positive passage for q_n and let $\mathcal{P}$ collect positives and sampled hard negatives, including cross-device passages. At any budget (a,b), the dense contrastive loss is

$$\mathcal{L}_{a,b} = -\frac{1}{B}\sum_{n=1}^{B} \log \frac{\exp(s_{a,b}(q_n,p_n^+)/\tau)}{\sum_{p\in\mathcal{P}}\exp(s_{a,b}(q_n,p)/\tau)}, \tag{5}$$

where $\tau > 0$ is a temperature and $\mathcal{B}$ is a nonempty set of smaller, nested budgets excluding (K_q, K_p). The full-capacity loss is separate from the averaged prefix losses:

$$\mathcal{L}_{\text{prefix}} = \frac{1}{|\mathcal{B}|}\sum_{(a,b)\in\mathcal{B}} \mathcal{L}_{a,b}. \tag{6}$$

Every term reuses the same full-capacity forward outputs, so prefix supervision adds scoring operations but no additional backbone pass. The nesting is along the representative-vector axis; the hidden dimension h of every retained vector is unchanged.

Dense, sparse, and hybrid modes. We retain DiffRetriever's sparse auxiliary objective. Let $\mathbf{Z}_x \in \mathbb{R}^{K_r\times V}$ contain representative logits for a vocabulary of size V. With content-token filter $\mathbf{c}_x \in \{0,1\}^V$, the sparse representation and score are

$$\mathbf{v}_x[v] = \mathbf{c}_x[v] \max_{1\le i\le K_r} \log(1+\mathrm{ReLU}(\mathbf{Z}_x[i,v]))\,, \qquad s_{\text{sp}}(q,p) = \mathbf{v}_q^\top \mathbf{v}_p. \tag{7}$$

The sparse contrastive loss replaces $s_{a,b}/\tau$ in (5) with s_{sp}. Only dense banks receive prefix losses. At retrieval, *dense* uses budget-selected late interaction, *sparse* uses s_{sp}, and *hybrid* combines full-capacity scores as $\alpha\widetilde{s}_{K_q,K_p} + (1-\alpha)\widetilde{s}_{\text{sp}}$, with independent normalization and fixed weight $\alpha \in [0,1]$. Sparse/hybrid retain full banks (Figure 3(c); Appendix B.1).

Curriculum. At optimizer progress $u = t/T \in [0,1]$, with warm-up fraction $0 < \rho < 1$,

$$\mathcal{L}(u) = \mathcal{L}_{K_q,K_p} + \lambda_{\text{sp}}\mathcal{L}_{\text{sp}} + \lambda_{\text{mrl}}(u)\mathcal{L}_{\text{prefix}}, \qquad \lambda_{\text{mrl}}(u) = \begin{cases} 0, & u < \rho, \\ \lambda_{\max}\dfrac{u-\rho}{1-\rho}, & u \ge \rho. \end{cases} \tag{8}$$

Full-capacity supervision remains active throughout training. We optimize the LoRA parameters ϕ within F_θ and the offset table $\mathbf{A}$, keeping pretrained weights frozen (Hu et al., 2022). Appendix B.4 specifies the budget set and the constant-weight, no-curriculum variant.

4.3 Budget-flexible Inference and Complexity

At deployment, we encode each input at (K_q, K_p) and retain (k_q, k_p) rows for matching, where $k_q \le K_q$ and $k_p \le K_p$. For $N = |\mathcal{C}|$ passages and b_{elt} bytes per stored coordinate, the raw dense-vector payload and exhaustive pairwise matching work scale as

$$M(k_p) = Nk_phb_{\text{elt}}, \qquad C(k_q,k_p) = \Theta(Nk_qk_ph). \tag{9}$$

One cached passage bank serves multiple budgets through post-encoding slicing. Increasing a budget requires retaining the full bank or re-encoding; removing input masks before a bidirectional forward pass is not equivalent to slicing its output. Savings concern dense-vector storage and late interaction, not encoding or measured end-to-end latency (Appendix B.1).

4.4 Theoretical Analysis

We analyze fixed, unit-norm banks truncated after encoding (section 4.3).

Prefix identifiability. Full-budget `MaxSim` is invariant to independent query/passage row permutations, whereas prefix scores need not be (Proposition C.1). The full-budget objective therefore cannot distinguish orders with different truncation quality. Nested losses break this indifference across supervised prefix boundaries; slot offsets supply position-specific parameters without guaranteeing semantic order. This concerns the objective, not encoder equivariance.

Ranking stability. Let $\epsilon_q(a;q)$ be the minimum mass-weighted Euclidean transport cost between uniform full/prefix query banks, and $\epsilon_p(b;p)$ the largest distance from a full passage vector to its nearest retained row (Appendix C.3). The maximum of unit-norm linear functions is 1-Lipschitz; query transport therefore bounds the change in the average, while passage coverage bounds the loss of a maximizer. Theorem C.2 yields

$$|s_{a,b}(q,p) - s_{K_q,K_p}(q,p)| \le \epsilon_q(a;q) + \epsilon_p(b;p) =: \epsilon_{a,b}(q,p). \tag{10}$$

More precisely, removing passage rows cannot increase a score at fixed query budget. For the full-budget margin $\gamma(q;p^+,p^-) = s_{K_q,K_p}(q,p^+) - s_{K_q,K_p}(q,p^-)$, this one-sided property gives

$$s_{a,b}(q,p^+) - s_{a,b}(q,p^-) \ge \gamma(q;p^+,p^-) - 2\epsilon_q(a;q) - \epsilon_p(b;p^+). \tag{11}$$

A positive right-hand side certifies preservation of the pairwise order (Corollary C.3). Thus, stable truncation depends on coverage relative to ranking margins, not vector count alone. These discrepancies are analytical certificates, not added training losses.

Budget-wise ranking control. A designated positive tied or outranked by a sampled negative has softmax probability at most $1/2$ and contrastive loss at least $\log 2$. Let $E_{a,b}$ be the fraction of such training queries. Averaging over queries and budgets yields (Proposition C.4)

$$\frac{1}{|\mathcal{B}|}\sum_{(a,b)\in\mathcal{B}} E_{a,b} \le \frac{\mathcal{L}_{\text{prefix}}}{\log 2}, \qquad E_{K_q,K_p} \le \frac{\mathcal{L}_{K_q,K_p}}{\log 2}. \tag{12}$$

Moreover, nonnegativity gives $\max_{(a,b)\in\mathcal{B}} E_{a,b} \le |\mathcal{B}|\mathcal{L}_{\text{prefix}}/\log 2$: averaging prefixes also bounds each supervised budget, albeit more loosely. The curriculum gradually introduces these constraints while retaining the full-budget loss. These finite-candidate bounds do not imply unconditional lossless compression or unseen-budget generalization. Appendix C details the proofs and their scope.

5 Experiments

5.1 Experimental Setting

Models and training configuration. We study Dream-v0-Instruct-7B and LLaDA-1.5 (8B) (Ye et al., 2025; Nie et al., 2025; Zhu et al., 2026), both at $(K_q, K_p) = (4, 16)$ with single-pass encoding and the complete offset–initialization–curriculum recipe. Training uses Tevatron MS MARCO augmented triples (Nguyen et al., 2016; Gao et al., 2023), one positive and fifteen hard negatives per query, and cross-device negatives. The launcher specifies rank-16 LoRA, AdamW at 10^{-4}, cosine decay, 6% learning-rate warm-up, and bf16/ZeRO-2. Full configuration and truncation defaults appear in Table 3; Algorithm 1 gives the training procedure.

Benchmarks and metrics. Following DiffRetriever (Wang et al., 2026), we evaluate *in-domain* on MS MARCO (MRR@10) and TREC DL19/20 (nDCG@10), and *out-of-domain* on the seven BEIR tasks listed in Table 2 (nDCG@10) (Nguyen et al., 2016; Craswell et al., 2020b;a; Thakur et al., 2021). BEIR Avg is their unweighted mean. Full-capacity evaluation includes dense, sparse, and equal-weight min–max-normalized hybrid scoring; budget curves use dense post-encoding prefixes. Appendix B.1 documents the scoring conventions.

Baselines and reporting. We evaluate fixed-budget DiffRetriever and Light-DiffRetriever under the same full-capacity pipeline, and include RepLLaMA, DiffEmbed, and BM25 as protocol-matched comparator rows. The main tables report only full-capacity scores, whereas the budget figure reuses the same checkpoints at retained prefixes. Each score uses standard benchmark aggregation, with matched metrics and retrieval modes for both methods.

Table 1: Full-budget in-domain retrieval: MS MARCO MRR@10 and DL19/20 nDCG@10. Bold/underline mark the best/second-best scores within each backbone. Subscripts give percentages of mode-matched DiffRetriever scores (dark red: above 100%).

Method	Mode	MS MARCO	TREC DL19	TREC DL20
BM25	lexical	.184	.506	.480
Qwen2.5 / Dream backbone				
DiffEmbed (Zhang et al., 2025b)	dense	**.405**	.720	.693
DiffRetriever (Wang et al., 2026)	dense	.395	.748	**.754**
	sparse	.317	.596	.624
	hybrid	.378	.719	.721
LIGHT-DIFFRETRIEVER	dense	**.405**$_{102.7\%}$	**.752**$_{100.5\%}$	.742$_{98.4\%}$
	sparse	.318$_{100.6\%}$	.599$_{100.4\%}$	.610$_{97.7\%}$
	hybrid	.384$_{101.6\%}$	.714$_{99.3\%}$	.698$_{96.7\%}$
LLaMA3 / LLaDA-1.5 backbone				
RepLLaMA (Ma et al., 2024)	dense	**.412**	.715	.690
DiffEmbed (Zhang et al., 2025b)	dense	.398	.695	.676
DiffRetriever (Wang et al., 2026)	dense	.385	**.728**	.724
	sparse	.303	.627	.609
	hybrid	.365	.710	.710
LIGHT-DIFFRETRIEVER	dense	.388$_{100.9\%}$	.719$_{98.7\%}$	**.731**$_{101.0\%}$
	sparse	.306$_{101.1\%}$	.628$_{100.1\%}$	.619$_{101.6\%}$
	hybrid	.370$_{101.4\%}$	.703$_{98.9\%}$	.701$_{98.8\%}$

Table 2: BEIR-7 full-budget nDCG@10 (Avg: unweighted). Notation follows Table 1.

Method	Mode	NQ	HotpotQA	SciFact	TREC-COVID	FiQA	ArguAna	Quora	Avg
BM25	lexical	.243	.567	.664	.530	.236	.275	.789	.472
Qwen2.5 / Dream backbone									
DiffEmbed (Zhang et al., 2025b)	dense	.625	.668	.735	.758	.478	.375	.831	.638
DiffRetriever (Wang et al., 2026)	dense	.629	.669	.740	**.860**	.480	**.402**	.870	**.664**
	sparse	.440	.589	.641	.677	.312	.276	.694	.518
	hybrid	.576	**.695**	.726	.819	.423	.384	.831	.636
LIGHT-DIFFRETRIEVER	dense	**.632**$_{100.6\%}$	.676$_{101.0\%}$	**.750**$_{101.4\%}$	.811$_{94.3\%}$	**.484**$_{100.9\%}$	.390$_{96.9\%}$	**.874**$_{100.5\%}$	.660$_{99.3\%}$
	sparse	.435$_{98.8\%}$	.575$_{97.7\%}$	.650$_{101.5\%}$	.640$_{94.5\%}$	.318$_{101.8\%}$	.304$_{110.2\%}$	.701$_{101.0\%}$	.517$_{99.8\%}$
	hybrid	.575$_{99.7\%}$	.688$_{99.1\%}$	.729$_{100.5\%}$	.792$_{96.7\%}$	.430$_{101.6\%}$	.370$_{96.5\%}$	.834$_{100.4\%}$	.631$_{99.2\%}$
LLaMA3 / LLaDA-1.5 backbone									
RepLLaMA (Ma et al., 2024)	dense	.622	.503	.653	.564	.383	**.411**	.821	.565
DiffEmbed (Zhang et al., 2025b)	dense	.587	.601	.714	.578	.455	.387	.842	.595
DiffRetriever (Wang et al., 2026)	dense	.629	.674	**.758**	.826	.453	.385	**.845**	**.653**
	sparse	.450	.602	.681	.647	.306	.304	.629	.517
	hybrid	.584	**.698**	.738	.797	.402	.367	.799	.626
LIGHT-DIFFRETRIEVER	dense	**.631**$_{100.4\%}$	.675$_{100.0\%}$	.748$_{98.7\%}$	**.832**$_{100.7\%}$	**.459**$_{101.3\%}$	.377$_{98.1\%}$	.822$_{97.2\%}$	.649$_{99.4\%}$
	sparse	.441$_{97.9\%}$	.585$_{97.1\%}$	.672$_{98.6\%}$	.689$_{106.5\%}$	.311$_{101.5\%}$	.315$_{103.5\%}$	.620$_{98.6\%}$	.519$_{100.3\%}$
	hybrid	.578$_{99.0\%}$	.689$_{98.7\%}$	.739$_{100.1\%}$	.802$_{100.6\%}$	.407$_{101.3\%}$	.362$_{98.7\%}$	.776$_{97.2\%}$	.622$_{99.3\%}$

5.2 MAIN RESULTS

Budget-flexible training preserves full-capacity effectiveness while improving several in-domain metrics (Table 1). Dream's dense MS MARCO MRR@10 rises from .395 to .405 (102.7% of DiffRetriever), and DL19 from .748 to .752. LLaDA-1.5 improves from .385/.724 to .388/.731 on MS MARCO/DL20. The remaining dense TREC comparisons retain 98.4% on Dream DL20 and 98.7% on LLaDA-1.5 DL19. These results support supervising reduced budgets alongside the full-capacity objective without systematically sacrificing full-budget quality.

Flexibility also largely preserves out-of-domain generalization (Table 2). Dense BEIR-7 averages reach .660/.649 on Dream/LLaDA-1.5, retaining 99.3%/99.4% of DiffRetriever and exceeding DiffEmbed's .638/.595. Dream improves on five datasets, including SciFact (.740 to .750) and HotpotQA (.669 to .676); LLaDA-1.5 improves on NQ, TREC-COVID, and FiQA. Sparse/hybrid averages retain 99.2–100.3% of matched baselines. Budget-adaptive banks therefore preserve aggregate transfer, although gains are not uniform across datasets.

Smaller banks retain near-full performance without retraining (Figure 4). At $(4, 4)$, Dream retains 98.2–99.7% across all four benchmarks with one quarter of the passage-vector storage; MS MARCO/BEIR-7 reach .398/.649. At $(4, 8)$, retention is 99.3–99.7% despite halving storage. Even $(2, 2)$ retains 95.6–97.6% with one eighth of the passage vectors; $(1, 1)$ retains 90.5–94.2% and

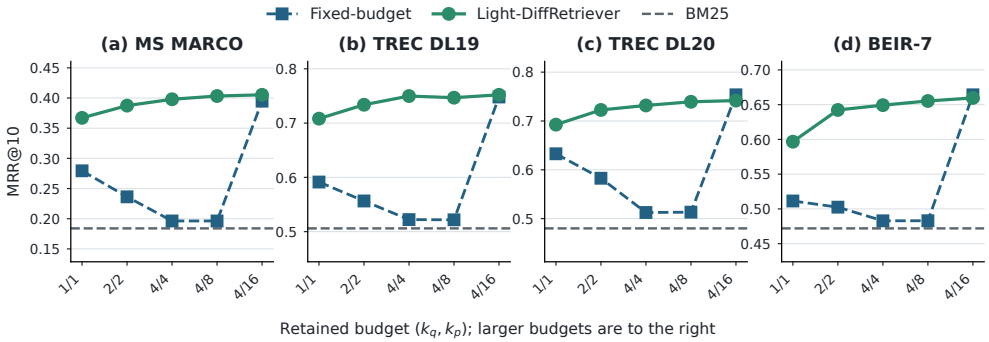


Figure 4: Dream dense prefix retrieval without retraining; gray dashed lines denote BM25.

surpasses matched fixed-budget prefixes on every benchmark. This nearly flat region supports concentrating retrieval evidence in reusable prefixes rather than separate budget-specific encoders.

Fixed-budget truncation is non-monotonic, further motivating prefix supervision. MS MARCO falls from .279/.236 at $(1,1)/(2,2)$ to .196 at $(4,4)$; BEIR-7 falls from .511/.503 to .483. Our $(4,4)$ scores, .398/.649, improve by .202/.166. Additional vectors can change query contributions or inflate irrelevant-document maxima (Appendix C.3); this is a possible mechanism, not a causal attribution. The unclipped $(1,1)$ rule (Appendix B.1) cannot explain the $(2,2)$-to-$(4,4)$ decline.

5.3 Hyperparameter Analysis

Initialization and the joint recipe. Initialization interacts with the training objective (Figure 5(a)). Endpoint $c_{K_{\max}} = .5$ reaches .665/.398 on BEIR-7/MS MARCO, versus .660/.390 at .3 and .658/.395 at .1. With prefix supervision and warm-up, .1 reaches .660/.405. The joint recipe leads on MS MARCO and is within .005 of the best BEIR-7 score.

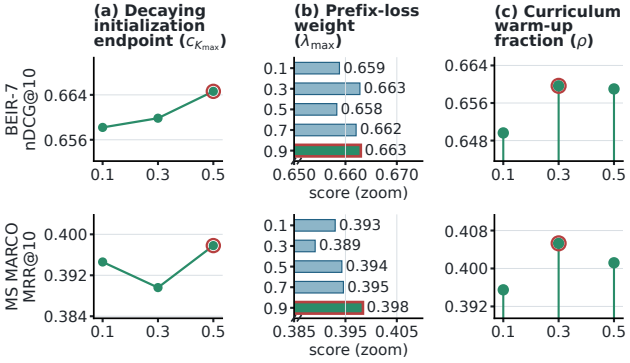


Figure 5: Full-budget Dream sweeps on BEIR-7 (top) and MS MARCO (bottom); outlines mark panel maxima.

Prefix-loss weight. Full-budget performance is stable across prefix-loss weights (Figure 5(b)). Varying $\lambda_{\max}$ from .1 to .9 yields BEIR-7 scores of .658–.663 and MS MARCO scores of .389–.398. Scores peak at .9 (.663/.398); .5 gives .658/.394, within .005 on both benchmarks. This sweep varies prefix-loss weighting without slot offsets or a curriculum. Within this setting, limited full-budget sensitivity across the sweep suggests that prefix constraints can coexist with full-capacity learning. The full-budget loss remains active, so increasing $\lambda_{\max}$ strengthens prefix supervision without replacing the objective for the complete bank.

Curriculum warm-up. An intermediate warm-up performs best in the matched comparison (Figure 5(c)). Increasing ρ from .1 to .3 raises BEIR-7 from .650 to .660 and MS MARCO from .396 to .405; extending it to .5 yields .659/.401. These estimates favor $\rho = .3$, suggesting a balance between initial full-capacity learning and subsequent prefix organization. A shorter warm-up introduces prefix constraints earlier, whereas a longer one reduces their training window. At fixed $\lambda_{\max} = .5$, integration over normalized training progress gives cumulative prefix weights of .225, .175, and .125, respectively. Thus, the schedule jointly adjusts supervision timing and cumulative weight (Appendix C.5); $\rho = .3$ balances early full-bank adaptation with subsequent prefix learning.

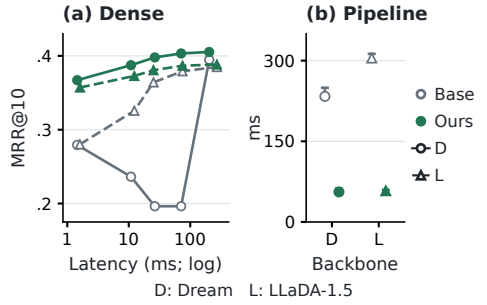


Figure 6: (a) Dense quality–latency curves; (b) warm-pipeline p50–p95: baseline $(4, 16)$ versus ours $(4, 4)$.

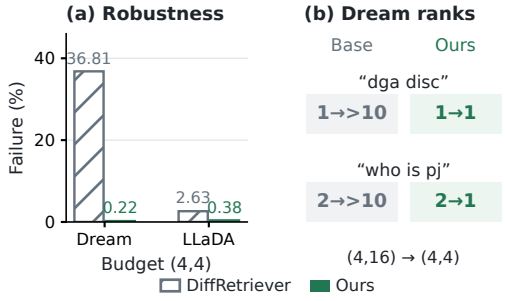


Figure 7: (a) Matched-cohort failures; (b) Dream ranks under truncation from $(4, 16)$ to $(4, 4)$.

5.4 Efficiency Analysis

Budget flexibility translates into substantial retrieval and storage savings with limited accuracy loss (Figure 6). We time exact search over 100K MS MARCO passages on one A100-80GB per run (20 warm-up, 100 measured queries), evaluating full-corpus accuracy separately. The curves cover five retained budgets, $(1, 1)$, $(2, 2)$, $(4, 4)$, $(4, 8)$, and $(4, 16)$, without retraining or re-encoding the corpus. Compared with full-budget DiffRetriever, our $(4, 4)$ prefix reduces median dense latency from 203.82 to 26.52 ms on Dream and from 271.28 to 25.20 ms on LLaDA-1.5 ($7.69\times$/$10.77\times$ faster). Full-corpus MRR@10 remains .3979/.3806, retaining 98.2%/98.0% of our own full-budget scores; Dream also exceeds the baseline's .3948. Representative-vector storage falls fourfold; including the auxiliary quotation vector, serialized files shrink from 24.37/27.85 to 7.17/8.19 GB ($3.40\times$). With query encoding, sparse search, and fusion included, median warm-pipeline latency drops from 233.50/304.88 to 55.86/58.09 ms ($4.18\times$/$5.25\times$), and p95 from 249.56/312.62 to 61.92/59.71 ms. This auxiliary hybrid pipeline measures timing, not hybrid accuracy. Same-budget dense timings differ by less than 0.5% at $(4, 4)$ and $(4, 16)$: savings arise from effective smaller banks, while unchanged full-capacity encoding imposes a latency floor (Appendix D.2).

5.5 Case Study

Prefix supervision preserves individual retrieval successes under truncation (Figure 7). Among 55,578 MS MARCO queries, we compare the same queries with full-budget RR@10 $\geq .5$ for both methods: 18,775 on Dream and 18,691 on LLaDA-1.5. At $(4, 4)$, the fraction losing every judged-relevant passage from the top ten falls from 36.81% to 0.22% on Dream (6,911 to 41 queries) and from 2.63% to 0.38% on LLaDA-1.5 (492 to 71). This matched-cohort comparison excludes differing full-budget success sets as the explanation. Across all queries, Dream's fraction with lower RR@10 after truncation also decreases from 49.3% to 13.9%, extending the stability pattern beyond initially strong queries. For Dream's "dga disc," truncation moves passage 7751426 from rank one beyond the baseline's top ten, while our prefix preserves rank one; for "who is pj," passage 7285839 falls from rank two beyond the baseline's top ten, whereas ours improves from second to first. These selected examples illustrate, rather than prove universally, that a smaller bank can preserve or improve relevant ranks. The pattern is consistent with directly supervising retained prefixes instead of relying on the full-budget objective to organize early slots. Appendix E provides all case records and diagnostics.

6 Conclusion

We presented Light-DiffRetriever, combining learnable slot offsets, decaying-scale initialization, and curriculum prefix supervision for budget-flexible DLM retrieval. Across Dream and LLaDA-1.5, retaining four of sixteen passage vectors preserves approximately 98% of full-budget MS MARCO and BEIR-7 aggregate performance with fourfold lower dense-vector storage. The savings concern storage and matching, not encoding, and dataset-level trade-offs remain. A shared model and corpus encoding can thus serve heterogeneous budgets for resource-adaptive retrieval. Appendices F–I discuss limitations and outlook, AI use, ethics, and reproducibility.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. In *International Conference on Learning Representations*, 2025. URL https://dblp.org/rec/conf/iclr/ArriolaGCYQHSK25.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces. In *Advances in Neural Information Processing Systems*, pp. 17981–17993, 2021. URL https://dblp.org/rec/conf/nips/AustinJHTB21.

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. doi: 10.48550/arXiv.2404.05961. URL https://dblp.org/rec/journals/corr/abs-2404-05961.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling Up Diffusion Language Models to 100B. *arXiv preprint arXiv:2512.15745*, 2025. doi: 10.48550/ARXIV.2512.15745. URL https://dblp.org/rec/journals/corr/abs-2512-15745.

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuocheng Gong, Yanmei Gu, Jian Guan, Kaiyuan Guan, Hongliang He, Zenan Huang, Juyong Jiang, Zhonghui Jiang, Zhenzhong Lan, Chengxi Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Yuan Lu, Yuxin Ma, Xingyu Mou, Zhenxuan Pan, Kaida Qiu, Yuji Ren, Jianfeng Tan, Yiding Tian, Zian Wang, Lanning Wei, Tao Wu, Yipeng Xing, Wentao Ye, Liangyu Zha, Tianze Zhang, Xiaolu Zhang, Junbo Zhao, Da Zheng, Hao Zhong, Wanli Zhong, Jun Zhou, Liwang Zhu, Muzhi Zhu, and Yihong Zhuang. LLaDA2.1: Speeding Up Text Diffusion via Token Editing. *arXiv preprint arXiv:2602.08676*, 2026. doi: 10.48550/ARXIV.2602.08676. URL https://dblp.org/rec/journals/corr/abs-2602-08676.

Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka Multimodal Models. In *International Conference on Learning Representations*, 2025. URL https://dblp.org/rec/conf/iclr/CaiY0L25.

Antoine Chaffin, Luca Arnaboldi, Amélie Chatelain, and Florent Krzakala. ColBERT-Zero: To Pre-train Or Not To Pre-train ColBERT models. *arXiv preprint arXiv:2602.16609*, 2026. doi: 10.48550/ARXIV.2602.16609. URL https://dblp.org/rec/journals/corr/abs-2602-16609.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, 2024. doi: 10.18653/v1/2024.findings-acl.137.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the TREC 2020 deep learning track. In *Proceedings of the 29th Text REtrieval Conference*, 2020a. URL https://trec.nist.gov/pubs/trec29/papers/OVERVIEW.DL.pdf.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the TREC 2019 deep learning track. *arXiv preprint arXiv:2003.07820*, 2020b. URL https://dblp.org/rec/journals/corr/abs-2003-07820.

Gabriel de Souza Pereira Moreira, Ronay Ak, Mengyao Xu, Oliver Holworthy, Benedikt Schifferer, Zhiding Yu, Yauhen Babakhin, Radek Osmulski, Jiarui Cai, Ryan Chesler, Bo Liu, and Even Oldridge. Nemotron ColEmbed V2: Top-Performing Late Interaction embedding models for Visual Document Retrieval. *arXiv preprint arXiv:2602.03992*, 2026. doi: 10.48550/ARXIV.2602.03992. URL https://dblp.org/rec/journals/corr/abs-2602-03992.

Laxman Dhulipala, Majid Hadian, Rajesh Jayaram, Jason Lee, and Vahab Mirrokni. MUVERA: Multi-vector retrieval via fixed dimensional encoding. In *Advances in Neural Information Processing Systems*, volume 37, pp. 101042–101073, 2024. doi: 10.52202/079017-3204. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b71cfefae46909178603b5bc6c11d3ae-Abstract-Conference.html.

Sedigheh Eslami, Maksim Gaiduk, Markus Krimmel, Louis Mark Milliken, Bo Wang, and Denis Bykov. Diffusion-pretrained dense and contextual embeddings. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 990–1004, 2026. doi: 10.18653/v1/2026.acl-industry.69. URL https://aclanthology.org/2026.acl-industry.69/.

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*, 2025. URL https://dblp.org/rec/conf/iclr/FaysseSWOVHC25.

Luyu Gao, Zhuyun Dai, and Jamie Callan. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3030–3042, 2021a. doi: 10.18653/v1/2021.naacl-main.241.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Tevatron: An efficient and flexible toolkit for neural retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3120–3124, 2023. doi: 10.1145/3539618.3591805.

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, 2021b. doi: 10.18653/v1/2021.emnlp-main.552.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Wenbo Hu, Zi-Yi Dou, Liunian Harold Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Matryoshka Query Transformer for Large Vision-Language Models. In *Advances in Neural Information Processing Systems*, 2024. URL https://dblp.org/rec/conf/nips/0006DLK0C24.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkxgnnNFvH.

Phung Gia Huy, Hai An Vu, Minh-Phuc Truong, Thang Duc Tran, Linh Ngo Van, Thanh Hong Nguyen, and Trung Le. MIPIC: Matryoshka Representation Learning via Self-Distilled Intra-Relational and Progressive Information Chaining. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13824–13838, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.676. URL https://dblp.org/rec/conf/acl/HuyVTTVNL26.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=jKN1pXi7b0.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 39–48, 2020. doi: 10.1145/3397271.3401075.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham M. Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30233–30249, 2022. doi: 10.52202/068431-2192. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c32319f4868da7613d78af9993100e42-Abstract-Conference.html.

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*, 2025. URL https://openreview.net/forum?id=lgsyLSsDRe.

Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Zhao. Rethinking the role of token retrieval in multi-vector retrieval. In *Advances in Neural Information Processing Systems*, volume 36, pp. 15384–15405, 2023. doi: 10.52202/075280-0677. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/31d997278ee9069d6721bc194174bb4c-Abstract-Conference.html.

Xianming Li, Zongxi Li, Jing Li, Haoran Xie, and Qing Li. 2D matryoshka sentence embeddings. *arXiv preprint arXiv:2402.14776*, 2024. doi: 10.48550/arXiv.2402.14776. URL https://dblp.org/rec/journals/corr/abs-2402-14776.

Xianming Li, Zongxi Li, Jing Li, Haoran Xie, and Qing Li. ESE: Espresso Sentence Embeddings. In *International Conference on Learning Representations*, 2025. URL https://dblp.org/rec/conf/iclr/LiLL0L25.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023. doi: 10.48550/arXiv.2308.03281. URL https://dblp.org/rec/journals/corr/abs-2308-03281.

Qi Liu, Kun Ai, Jiaxin Mao, Yanzhao Zhang, Mingxin Li, Dingkun Long, Pengjun Xie, Fengbin Zhu, and Ji-Rong Wen. DiffuRank: Effective document reranking with diffusion language models. *arXiv preprint arXiv:2602.12528*, 2026. doi: 10.48550/arXiv.2602.12528. URL https://dblp.org/rec/journals/corr/abs-2602-12528.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32819–32848, 2024. URL https://dblp.org/rec/conf/icml/LouME24.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dKV-Cache: The Cache for Diffusion Language Models. In *Advances in Neural Information Processing Systems*, 2025. URL https://dblp.org/rec/conf/nips/MaYFW25.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning LLaMA for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2421–2425, 2024. doi: 10.1145/3626772.3657951.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BC4lIvfSzv.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016. URL https://dblp.org/rec/journals/corr/NguyenRSGTMD16.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 50608–50646, 2025. doi: 10.52202/085713-1689. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/48b383b24230e0e6e649d9c98dae4d8c-Abstract-Conference.html.

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*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

Trishan Singha Roy, Arkadeep Acharya, Vishwajeet Kumar, Jaydeep Sen, and Sachindra Joshi. Spatial Matryoshka Training for Multi-Granularity Visual Document Retrieval. *arXiv preprint arXiv:2608.29951*, 2026. doi: 10.48550/ARXIV.2608.29951. URL https://dblp.org/rec/journals/corr/abs-2608-29951.

Subham S. Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models. In *Advances in Neural Information Processing Systems*, 2024. URL https://dblp.org/rec/conf/nips/SahooASGMCRK24.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. PLAID: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pp. 1747–1756, 2022a. doi: 10.1145/3511808.3557325.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, 2022b. doi: 10.18653/v1/2022.naacl-main.272.

Jan Luca Scheerer, Matei Zaharia, Christopher Potts, Gustavo Alonso, and Omar Khattab. WARP: An Efficient Engine for Multi-Vector Retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2504–2512, 2025. doi: 10.1145/3726302.3729904. URL https://dblp.org/rec/conf/sigir/ScheererZPAK25.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. Repetition improves language model embeddings. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ahlrf2HGJR.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract-round2.html.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022. doi: 10.48550/arXiv.2212.03533. URL https://dblp.org/rec/journals/corr/abs-2212-03533.

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. doi: 10.18653/v1/2024.acl-long.642.

Shuai Wang, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. 2D matryoshka training for information retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3125–3134, 2025. doi: 10.1145/3726302.3730330. URL https://dblp.org/rec/conf/sigir/0032ZKZ25.

Shuai Wang, Yu Yin, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. DiffRetriever: Parallel representative tokens for retrieval with diffusion language models. *arXiv preprint arXiv:2605.07210*, 2026. doi: 10.48550/arXiv.2605.07210. URL https://dblp.org/rec/journals/corr/abs-2605-07210.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo O. Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient Block-Diffusion LLM. *arXiv preprint arXiv:2509.26328*, 2025. doi: 10.48550/ARXIV.2509.26328. URL https://dblp.org/rec/journals/corr/abs-2509-26328.

Haowen Xiang, Yibo Yan, Jiahao Huo, Yu Huang, Yi Cao, Mingdong Ou, and Xuming Hu. MM-Matryoshka: Towards Budget-Elastic Visual Document Retrieval via a 2D Multimodal Matryoshka Training Framework. *arXiv preprint arXiv:2606.07654*, 2026. doi: 10.48550/ARXIV.2606.07654. URL https://dblp.org/rec/journals/corr/abs-2606-07654.

Zilin Xiao, Qi Ma, Mengting Gu, Chun cheng Jason Chen, Xintao Chen, Vicente Ordonez, and Vijai Mohan. MetaEmbed: Scaling Multimodal Retrieval at Test-Time with Flexible Late Interaction. *arXiv preprint arXiv:2509.18095*, 2025. doi: 10.48550/ARXIV.2509.18095. URL https://dblp.org/rec/journals/corr/abs-2509-18095.

Yibo Yan, Mingdong Ou, Yi Cao, Xin Zou, Jiahao Huo, Shuliang Liu, James T. Kwok, and Xuming Hu. Sculpting the Vector Space: Towards Efficient Multi-Vector Visual Document Retrieval via Prune-then-Merge Framework. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24883–24925, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.1247. URL https://dblp.org/rec/conf/acl/YanOCZHLKH26.

Eugene Yang, Andrew Yates, Dawn J. Lawrie, James Mayfield, Saron Samuel, and Rohan Jha. ColBERTSaR: Sparsified ColBERT Index via Product Quantization. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 4305–4310, 2026. doi: 10.1145/3805712.3809920. URL https://dblp.org/rec/conf/sigir/YangYLMSJ26.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025. doi: 10.48550/arXiv.2508.15487. URL https://dblp.org/rec/journals/corr/abs-2508-15487.

Jinsung Yoon, Rajarishi Sinha, Sercan O Arik, and Tomas Pfister. Matryoshka-Adaptor: Unsupervised and supervised tuning for smaller embedding dimensions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 10318–10336, 2024. doi: 10.18653/v1/2024.emnlp-main.576. URL https://aclanthology.org/2024.emnlp-main.576/.

Biao Zhang, Lixin Chen, Tong Liu, and Bo Zheng. SMEC:rethinking matryoshka representation learning for retrieval embedding compression. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26209–26222, 2025a. doi: 10.18653/v1/2025.emnlp-main.1332. URL https://aclanthology.org/2025.emnlp-main.1332/.

Siyue Zhang, Yilun Zhao, Liyuan Geng, Arman Cohan, Anh Tuan Luu, and Chen Zhao. Diffusion vs. autoregressive language models: A text embedding perspective. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4273–4303, 2025b. doi: 10.18653/v1/2025.emnlp-main.213.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025c. doi: 10.48550/arXiv.2506.05176. URL https://dblp.org/rec/journals/corr/abs-2506-05176.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11425–11460, 2026. doi: 10.18653/v1/2026.acl-long.524. URL https://aclanthology.org/2026.acl-long.524/.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. PromptReps: Prompting large language models to generate dense and sparse representations for zero-shot document retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4375–4391, 2024. doi: 10.18653/v1/2024.emnlp-main.250.

Shengyao Zhuang, Shuai Wang, Fabio Zheng, Bevan Koopman, and Guido Zuccon. Starbucks: Improved training for 2D matryoshka embeddings. In *Advances in Information Retrieval: 48th European Conference on Information Retrieval, Part I*, pp. 67–82, 2026. doi: 10.1007/978-3-032-21289-4_5. URL https://dblp.org/rec/conf/ecir/ZhuangWZKZ26.

APPENDIX CONTENTS

Extended related work appears in Appendix A; implementation details and pseudocode in Appendix B; complete proofs in Appendix C; additional experimental analysis in Appendix D; case studies in Appendix E; and outlook and research statements in Appendices F–I. Section titles and page numbers below link directly to their destinations.

A More Related Work **16**
- A.1 Diffusion LLMs . . . 16
- A.2 Multi-Vector Retrieval . . . 16
- A.3 Matryoshka Representation Learning . . . 17

B Implementation and Reproducibility Details **18**
- B.1 Scoring Conventions . . . 18
- B.2 Training and Evaluation Defaults . . . 19
- B.3 Retrieval Prompts and Input Format . . . 20
- B.4 Training and Deployment Algorithm . . . 20

C Theoretical Foundations and Proofs **22**
- C.1 Notation and Scope . . . 22
- C.2 Full-budget Symmetry Does Not Identify Prefix Utility . . . 22
- C.3 Geometric Conditions for Stable Truncation . . . 23
- C.4 From Prefix Contrastive Loss to Ranking Error . . . 24
- C.5 Initialization and Curriculum Properties . . . 25
- C.6 Evaluation Conventions and Limits of the Guarantees . . . 26

D More Experimental Analysis **26**
- D.1 Complete Budget Results . . . 26
- D.2 Efficiency Protocol and Measurement Scope . . . 28
- D.3 Complete Efficiency Measurements . . . 29
- D.4 Loading and First-query Behavior . . . 31

E Case Study **32**
- E.1 Per-query Robustness and Cohort Definitions . . . 32
- E.2 Retention Across Datasets and Query Groups . . . 33
- E.3 Complete Qualitative Ranking Records . . . 34
- E.4 Positional-token Diagnostic . . . 35

F Limitations and Outlook **36**

G AI Use Statement **36**

H Ethics Statement **36**

I Reproducibility Statement **37**

A More Related Work

A.1 Diffusion LLMs

Diffusion language modeling replaces a strictly left-to-right factorization with learning to reverse a corruption process, allowing masked positions to use context from both directions. D3PM established discrete-state diffusion with structured transition kernels, including absorbing states (Austin et al., 2021); SEDD introduced score-entropy learning of discrete probability ratios (Lou et al., 2024); and MDLM simplified masked-diffusion training through a Rao–Blackwellized objective related to weighted masked-language-modeling losses (Sahoo et al., 2024). Subsequent work has pursued scale, alignment, and decoding efficiency rather than a single architectural trajectory. Block Diffusion combines autoregressive ordering across blocks with diffusion within blocks, supporting flexible-length generation and caching (Arriola et al., 2025); dKV-Cache studies delayed, conditioned reuse of key–value states during denoising (Ma et al., 2025). Fast-dLLM v2 further explores efficient conversion of autoregressive models into block-diffusion models with hierarchical caching (Wu et al., 2025). More recent scaling efforts include the 100B-parameter mixture-of-experts variant of LLaDA2.0 and the mask-to-token plus token-to-token editing formulation of LLaDA2.1 (Bie et al., 2025; 2026). These developments concern generative modeling and decoding; they do not by themselves establish that truncated retrieval representations remain discriminative.

Dream and LLaDA-1.5 provide two representative, complementary foundations for studying this representation question. Dream adapts autoregressive pretrained weights, using the Qwen2.5-7B configuration and diffusion-oriented continued training (Ye et al., 2025). The original LLaDA instead learns an 8B masked-diffusion model from scratch, while LLaDA-1.5 builds on that family through variance-reduced preference optimization (Nie et al., 2025; Zhu et al., 2026). Both expose bidirectionally contextualized hidden states at masked positions, permitting parallel representative extraction without invoking their iterative generation procedures. Our use of these backbones therefore covers autoregressive-to-diffusion adaptation and a natively diffusion-pretrained family, rather than two sizes of the same pretrained model; their distinct training histories make them informative test cases for budget-flexible DLM retrieval.

A.2 Multi-Vector Retrieval

Textual multi-vector retrieval retains local matching evidence that a single pooled embedding can discard. ColBERT operationalized this principle with independently encoded token representations and late `MaxSim` interaction (Khattab & Zaharia, 2020). ColBERTv2 and PLAID subsequently improved storage and search through residual compression and centroid-based candidate filtering (Santhanam et al., 2022b;a), whereas COIL restricted contextualized interactions to lexical matches (Gao et al., 2021a). XTR aligns training more closely with token retrieval, and WARP supplies an efficient execution engine for the resulting scoring pipeline (Lee et al., 2023; Scheerer et al., 2025). MUVERA takes a complementary route by constructing fixed-dimensional encodings for multi-vector similarity search (Dhulipala et al., 2024). Recent directions include multi-vector-specific pretraining in ColBERT-Zero and conversion of quantized ColBERT representations into a sparse inverted index in ColBERTSaR (Chaffin et al., 2026; Yang et al., 2026). Collectively, these approaches improve representation quality, compression, or search execution, but do not make arbitrary retained prefixes of an existing representative bank equally effective.

The same trade-off becomes particularly acute in multimodal retrieval, where document pages can contain hundreds or thousands of visual patches. ColPali applies late interaction directly to page-image representations, retaining text, layout, and visual evidence without requiring a separate OCR-based indexing pipeline (Faysse et al., 2025). Nemotron ColEmbed V2 extends this paradigm to stronger vision–language backbones and emphasizes training-data, negative-mining, and representation-design choices (de Souza Pereira Moreira et al., 2026). For storage reduction, Prune-then-Merge removes low-information patches before hierarchical merging (Yan et al., 2026); MetaEmbed instead learns compact retrieval tokens with nested multi-vector supervision (Xiao et al., 2025). The recent spatial Matryoshka approach, ColSNAP, jointly trains pooled patch-grid tiers that can be selected after a single encoding (Roy et al., 2026). These methods reinforce the importance of controlling the number of stored vectors, while also showing that token-budget learning is not unique to diffusion models. Our focus is the distinct setting of parallel DLM representatives: we learn ordered query and passage

prefixes through slot-aware encoding and curriculum contrastive supervision, targeting post-encoding storage and matching budgets.

A.3 Matryoshka Representation Learning

Embedding coordinates. The original MRL jointly supervises nested coordinate prefixes of an embedding, so a vector $\mathbf{z} \in \mathbb{R}^h$ can be deployed as $\mathbf{z}[1:d] \in \mathbb{R}^d$ at a selected dimension $d \leq h$ (Kusupati et al., 2022). Matryoshka-Adaptor applies learned transformations to existing embeddings, while SMEC studies sequential Matryoshka optimization, adaptive dimension selection, and cross-batch supervision for retrieval compression (Yoon et al., 2024; Zhang et al., 2025a). Here the nested object is the *coordinate axis*: reducing dimension shortens each stored vector, without necessarily reducing the number of vectors or the encoder depth (Figure 8(a)).

Encoder depth and coordinates. 2D Matryoshka sentence embeddings extend adaptive representations to intermediate encoder layers as well as coordinate prefixes (Li et al., 2024). ESE combines shallow-layer representation learning with PCA-guided feature compression (Li et al., 2025); 2D Matryoshka training for information retrieval and Starbucks develop retrieval-oriented depth–dimension training (Wang et al., 2025; Zhuang et al., 2026). MIPIC further couples cross-dimensional relational alignment with progressive transfer of semantic information across depth (Huy et al., 2026). In visual document retrieval, MM-Matryoshka applies joint width and layer supervision to ColPali-style multi-vector encoders (Xiang et al., 2026). These approaches select an output layer and a vector width; reducing encoder depth can affect encoding computation, a different intervention from retaining fewer vectors after the final layer (Figure 8(b)).

Token counts and spatial granularities. The Matryoshka principle also operates beyond embedding coordinates. MQT trains a query transformer with prefixes of latent visual queries, whereas Matryoshka Multimodal Models (M3) use spatial pooling to learn visual-token sets at multiple granularities (Hu et al., 2024; Cai et al., 2025). Both vary how much visual information is passed to a downstream language model, but pooled coarse tokens are not generally literal row prefixes of finer tokens (Figure 8(c)). For retrieval, MetaEmbed supervises nested groups of learned Meta Tokens, and ColSNAP constructs jointly trained spatial pooling tiers from page-patch embeddings (Xiao et al., 2025; Roy et al., 2026). These are closely related precedents for flexible interaction budgets, with different token construction and nesting operators.

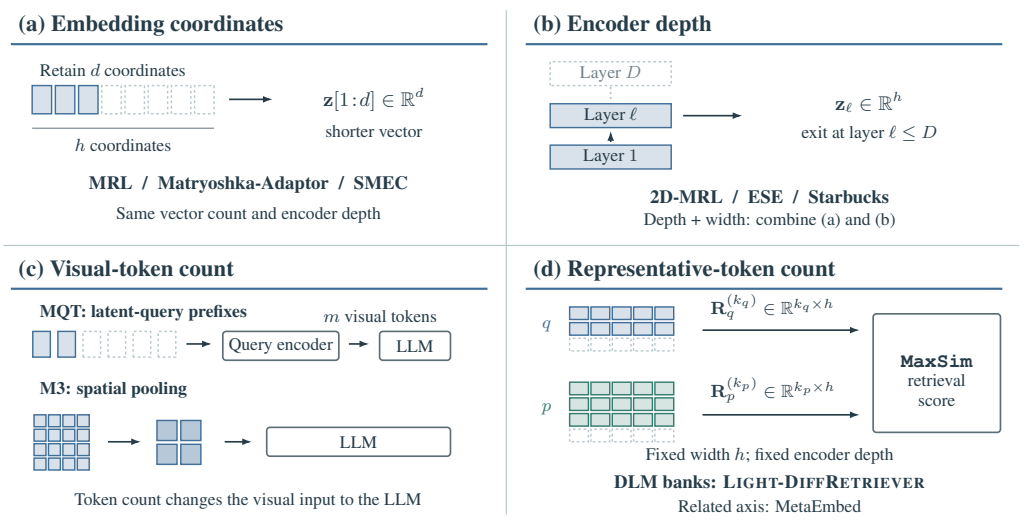


Figure 8: Different objects of Matryoshka nesting. **(a)** Coordinate prefixes shorten each embedding. **(b)** Layer selection changes encoder depth from D to ℓ; 2D approaches combine (a) and (b). **(c)** Visual-token budgets use latent-query prefixes (MQT) or pooled spatial grids (M3), which are distinct operators. **(d)** Retrieval-vector prefixes retain complete h-dimensional rows for late interaction. MetaEmbed is a related precedent on this axis; LIGHT-DIFFRETRIEVER studies ordered DLM query/passage banks with fixed encoding capacity. Filled cells are retained and dashed cells are omitted; counts are schematic.

Our representative-token axis. LIGHT-DIFFRETRIEVER retains $\mathbf{R}_q[1:k_q,:]$ and $\mathbf{R}_p[1:k_p,:]$, preserving the hidden dimension h and full encoder depth (Figure 8(d)). The distinction is therefore not a claim to originate token-count nesting: it is its formulation for bidirectional DLM representative banks, coupled with learnable slot offsets, decaying initialization, and curriculum prefix-level retrieval losses. Unlike width truncation, our budgets reduce the number of vectors; unlike depth selection or visual-token reduction before an LLM, they are applied after full-capacity encoding. This isolates the storage–late-interaction trade-off and permits multiple deployment budgets to share one trained encoder and cached corpus encoding.

B IMPLEMENTATION AND REPRODUCIBILITY DETAILS

B.1 SCORING CONVENTIONS

The training objective uses the query-averaged `MaxSim` in equation (4) without clamping similarities. The supplied multi-vector evaluator instead ranks with

$$s^+_{a,b}(q,p) = \sum_{i=1}^{a} \max\left\{0, \max_{1\leq j\leq b} \mathbf{R}_q[i,:]\mathbf{R}_p[j,:]^\top\right\}. \tag{13}$$

The missing factor $1/a$ does not change a ranking at a fixed query budget or a min–max-normalized hybrid score, whereas flooring negative maxima at zero can change rankings. The reported evaluation follows this implementation distinction. At $(1,1)$ the evaluator uses the unclipped dot product of the first query and passage vectors.

The mode named ‘single_dense‘ in a full-capacity summary averages normalized representative vectors before taking a dot product; it is not a model trained with one mask and is not the $(1,1)$ prefix. Prefix curves read $(1,1)$ from ‘single_dense_kq1_kp1‘ and larger prefixes from the corresponding ‘multi_dense_kq..._kp...‘ entries. Sparse retrieval extracts lowercase alphabetic non-stopwords, max-pools representative logits after $\log(1+\mathrm{ReLU}(\cdot))$, retains the evaluator’s top-128 entries, and combines dense and sparse top-1,000 lists after independent per-query min–max normalization.

B.2 Training and Evaluation Defaults

Table 3: Training and evaluation defaults for both DLM backbones.

Setting	Value
Backbones	Dream-v0-Instruct-7B; LLaDA-1.5
Maximum query/passage masks	$K_q = 4$, $K_p = 16$ (both backbones)
Representative passes	$S = 1$
Position-aware offsets	Shared $16 \times h$ table, tail-aligned
Offset initialization	Descending linear scale, $\sigma = .02$, $c_{K_{\max}} = .1$
MRL prefixes	$(1, 1), (2, 2), (4, 4), (4, 8)$
MRL target / warmup	$\lambda_{\max} = .5$, $\rho = .3$
Dense / sparse loss weights	$1/1$; dense temperature .01
Training data	Tevatron MS MARCO augmented triples
Passages per query	1 positive + 15 hard negatives
Query / passage text limit	32 / 156 tokens in training
LoRA	Rank 16, scaling 64, dropout .05
LoRA target matrices	Attention Q/K/V/O and feed-forward projections
Optimizer	AdamW, 10^{-4} learning rate, .01 weight decay
Schedule / clipping	Cosine; .06 warmup; gradient norm 1.0
Epochs	2 (DiffRetriever paper: 1)
Batch / accumulation	8 devices $\times$ 4 queries $\times$ 4 steps = 128
Precision / distributed	bf16 / DeepSpeed ZeRO-2
Seed	42; data seed 42
Evaluation text limit	512 query / 512 passage tokens
Retrieval / metric cutoffs	Top 1,000 / top 10
Sparse top-k / fusion	128 / .5 dense + .5 sparse, normalized

B.3 Retrieval Prompts and Input Format

The default training recipe selects the `few` prompt variant for both Dream and LLaDA-1.5. Its system and user instructions match the multi-representation wording in Table 1 of Wang et al. (2026). The match is not character-for-character: the code uses straight quotation marks and the assistant prefix `The words are: "`, including a colon absent from that table. The boxes below reproduce the implementation's instruction text; `{query}` and `{passage}` denote substituted inputs, and the role labels are explanatory rather than literal prompt tokens.

Shared system message

```
You are an AI assistant that can understand human language.
```

Query retrieval template

User message:
```
Query: "{query}". Use a few words to represent the query in a retrieval task. Make sure your words are in lowercase.
```
Assistant prefix:
```
The words are: "
```

Passage retrieval template

User message:
```
Passage: "{passage}". Use a few words to represent the passage in a retrieval task. Make sure your words are in lowercase.
```
Assistant prefix:
```
The words are: "
```

Backbone-specific serialization. The system and user messages are rendered with the backbone tokenizer's native chat template and `add_generation_prompt=True`, after which the assistant prefix is appended. For role $r \in \{q, p\}$, the implementation inserts K_r mask tokens followed by a structural tail comprising a closing quote, a turn-end token, and EOS:

$$\texttt{The words are: "}\underbrace{\texttt{[MASK]}_1 \cdots \texttt{[MASK]}_{K_r}}_{K_r \text{ representative positions}}\texttt{" <turn-end> <eos>.}$$

Here `[MASK]` is schematic: the adapters specify Dream's `<|mask|>` (ID 151666) and LLaDA-1.5's `<|mdm_mask|>` (ID 126336). Training saves the query/passage prefix and suffix token IDs in the checkpoint configuration; the trained-model evaluation path restores these IDs rather than rebuilding the prompt from a requested retained budget.

Encoding capacity versus retained budget. The word-count helper rewrites explicit number words such as `one` or `three`, but leaves `a few words` unchanged. Thus, $K_q = 4$ and $K_p = 16$ determine the inserted mask counts, not numerical requests in the instruction. All retained budgets (k_q, k_p) reuse these full-capacity encodings and differ only in output slicing. In particular, $(1, 1)$ truncation retains this prompt; using the `one word` variant requires a separate prompt selection.

B.4 Training and Deployment Algorithm

Algorithm 1 summarizes the single-pass configuration studied in this paper, with $K_q > 1$ as in both main models. It uses the symbols of section 4: ϕ denotes the trainable LoRA parameters within F_θ, and $\mathbf{A}$ is the shared offset table. The backbone weights and vocabulary head are frozen. Gradient accumulation and distributed sharding are implementation details of the optimizer update, not separate budget-specific encoders.

Algorithm 1: LIGHT-DIFFRETRIEVER: shared encoding, curriculum prefix training, and budget-flexible deployment.

Input : Training triples; DLM F_θ with LoRA parameters $\boldsymbol{\phi}$; capacities K_q, K_p; smaller budgets $\mathcal{B}$; scales $\{c_j\}_{j=1}^{K_{\max}}$, $\sigma > 0$; temperature $\tau > 0$; weights $\lambda_{\mathrm{sp}}, \lambda_{\max} \geq 0$; warm-up fraction $0 < \rho < 1$; T optimizer updates

Output : One trained encoder and offset table, supporting retained dense budgets (k_q, k_p) and full-capacity sparse/hybrid retrieval

▷ `Shared encoder: one forward pass per input`

Function `Encode`(x, r), *where* $r \in \{q, p\}$**:**
- Construct the prompted embeddings $\mathbf{U}_x$ with K_r masks at $\mathcal{G}_x$
- **for** $i = 1, \ldots, K_r$ **do**
  - $\mathbf{U}_x[g_{x,i}, :] \leftarrow \mathbf{e}_{\mathrm{mask}}^\top + \mathbf{A}[K_{\max} - K_r + i, :]$
- $\mathbf{H}_x \leftarrow F_\theta(\mathbf{U}_x)$; extract representative logits $\mathbf{Z}_x$ from the same pass
- Normalize $\mathbf{H}_x[g_{x,i}, :]$ rowwise to form $\mathbf{R}_x \in \mathbb{R}^{K_r \times h}$ (1)
- Form $\mathbf{v}_x \in \mathbb{R}^V$ from $\mathbf{Z}_x$ and content filter $\mathbf{c}_x$ (7)
- **return** $(\mathbf{R}_x, \mathbf{v}_x)$

▷ `Training: full-capacity anchor and smaller-prefix supervision`

Freeze pretrained weights; initialize LoRA and $\mathbf{A}[j, :] \leftarrow c_j \boldsymbol{\epsilon}_j^\top$ as in (3)

for $t = 0, \ldots, T-1$ **do**
- Set $u \leftarrow t/T$ and $\lambda \leftarrow \lambda_{\mathrm{mrl}}(u)$ using (8)
- Sample a global batch of B queries, designated positives, and hard negatives
- Encode local queries with `Encode`$(\cdot, q)$ and passages with `Encode`$(\cdot, p)$
- Gather cross-device banks, sparse vectors, and positive indices to form $\mathcal{P}$
- Compute $\mathcal{L}_{K_q, K_p}$ by query-averaged `MaxSim` and temperature τ (5)
- Compute $\mathcal{L}_{\mathrm{sp}}$ from full-capacity sparse dot products, without temperature scaling
- $\mathcal{L}_{\mathrm{prefix}} \leftarrow 0$
- **if** $\lambda > 0$ **then**
  - **foreach** $(a, b) \in \mathcal{B}$ **do**
    - Slice $\mathbf{R}_q[1\!:\!a, :]$ and $\mathbf{R}_p[1\!:\!b, :]$; compute $\mathcal{L}_{a,b}$ via (5)
    - $\mathcal{L}_{\mathrm{prefix}} \leftarrow \mathcal{L}_{\mathrm{prefix}} + \mathcal{L}_{a,b}/|\mathcal{B}|$
- $\mathcal{L} \leftarrow \mathcal{L}_{K_q, K_p} + \lambda_{\mathrm{sp}} \mathcal{L}_{\mathrm{sp}} + \lambda \mathcal{L}_{\mathrm{prefix}}$
- Update only $(\boldsymbol{\phi}, \mathbf{A})$ with AdamW using $\nabla_{(\boldsymbol{\phi}, \mathbf{A})} \mathcal{L}$

▷ `Deployment: retain prefixes after full-capacity encoding`

Encode and cache $(\mathbf{R}_p, \mathbf{v}_p)$ for all $p \in \mathcal{C}$ once with the trained encoder

foreach *query* q *and requested mode* **do**
- $(\mathbf{R}_q, \mathbf{v}_q) \leftarrow$ `Encode`(q, q)
- **if** *mode is dense* **then**
  - Select (k_q, k_p); rank with $\widehat{s}_{k_q, k_p}(q, p)$ on retained rows
- **else if** *mode is sparse* **then**
  - Rank with $\mathbf{v}_q^\top \mathbf{v}_p$ using the full-capacity lexical representations
- **else**
  - Retrieve full-capacity dense/sparse lists; independently min–max normalize their scores
  - Fuse their candidate union with weight α; rank by the hybrid score in section 4.2

Budget construction and schedule. The default code constructs smaller pairs $(a, b) = (\min(k, K_q), \min(k, K_p))$ from configured positive prefix lengths $k < K_{\max}$. These pairs form $\mathcal{B}$; the full pair is optimized separately, not counted again in its average. A configuration explicitly containing the full pair would instead reweight the full loss and must be reported as such. For $K_{\max} > 1$, the descending linear initializer uses $c_j = 1 - (1 - c_{K_{\max}})(j - 1)/(K_{\max} - 1)$; with one slot, its scale is specified directly. Optimizer progress is written to the underlying model even when wrapped by distributed training. Setting the implementation's warm-up option to zero disables the curriculum and uses the constant weight $\lambda_{\max}$; it does *not* mean substituting $\rho = 0$ into the ramp in (8).

Scoring and caching. In the algorithm, $\widehat{s}_{1,1} = s_{1,1}$ and $\widehat{s}_{a,b} = s^{+}_{a,b}$ otherwise, following Appendix B.1. Sparse deployment additionally keeps the largest vocabulary weights according to the evaluation setting. Hybrid retrieval uses the union of the two candidate lists, with an absent list score assigned zero after per-list normalization; a list with zero score range receives zero normalized scores. Sparse and hybrid retrieval use full-capacity representative banks. Dense-only deployment may omit vocabulary-logit extraction and sparse storage. Retaining the full corpus banks permits future budget increases without re-encoding; storing only a shortened bank does not. No step re-encodes an input once per retained budget.

C Theoretical Foundations and Proofs

This section gives deterministic statements about the representation and objective in section 4. The results require neither a probabilistic model of relevance nor an assumption that optimization reaches a global minimum. They explain why full-budget training leaves prefix quality unconstrained, identify sufficient conditions for ranking stability, and connect prefix contrastive loss to finite-candidate ranking errors. They do not establish a population generalization bound or an unconditional guarantee of lossless compression.

C.1 Notation and Scope

Fix a trained encoder, a query q, and its full-capacity encoded banks. Write $\mathbf{r}_{x,i} = \mathbf{R}_x[i, :]^\top \in \mathbb{R}^h$, and assume $\|\mathbf{r}_{x,i}\|_2 = 1$ for all representative rows. This is the idealized normalization in (1); the score bounds also hold for norms at most one. All prefixes are selected *after* encoding, so the retained rows are exactly the corresponding full-bank rows. The results need not hold if the input mask count is changed and the sequence is re-encoded.

The full query and passage capacities are K_q and K_p; (a, b) denotes a retained pair, and (k_q, k_p) denotes a deployment request. The nonempty training set $\mathcal{B}$ contains smaller paired budgets, whereas $\mathcal{B} \cup \{(K_q, K_p)\}$ contains all supervised budgets. The score $s_{a,b}$ is the query-averaged, unclipped score in (4). Unless stated otherwise, comparisons use fixed representations and a fixed candidate collection, not separately trained checkpoints. All logarithms are natural.

C.2 Full-budget Symmetry Does Not Identify Prefix Utility

Proposition C.1 (Representation-level permutation invariance)**.** *Let $\mathbf{\Pi}_q \in \{0,1\}^{K_q \times K_q}$ and $\mathbf{\Pi}_p \in \{0,1\}^{K_p \times K_p}$ be permutation matrices. Replacing $\mathbf{R}_q$ by $\mathbf{\Pi}_q \mathbf{R}_q$ and $\mathbf{R}_p$ by $\mathbf{\Pi}_p \mathbf{R}_p$ leaves $s_{K_q,K_p}(q, p)$ unchanged. Consequently, the full dense contrastive loss is unchanged under such row permutations. The sparse max-pooling representation is also unchanged if the corresponding logit rows are permuted. A proper-prefix loss is not invariant in general to permutations moving rows across its truncation boundary.*

Proof. Let π_q, π_p denote the bijections represented by the matrices. Then

$$\frac{1}{K_q} \sum_{i=1}^{K_q} \max_{1 \le j \le K_p} \mathbf{r}^\top_{q,\pi_q(i)} \mathbf{r}_{p,\pi_p(j)} = \frac{1}{K_q} \sum_{i=1}^{K_q} \max_{1 \le j \le K_p} \mathbf{r}^\top_{q,i} \mathbf{r}_{p,j}. \tag{14}$$

The maximum and sum are invariant to bijective reindexing. Every score entering (5) is therefore unchanged. Likewise, the maximum over rows in (7) is unchanged, and the content filter is independent of row order.

For a counterexample to prefix invariance, take $h = 2$, $K_q = 1$, $K_p = 2$, and orthonormal vectors $\mathbf{e}_1, \mathbf{e}_2$. Let the query row be $\mathbf{e}_1$, the positive passage rows be $(-\mathbf{e}_1, \mathbf{e}_1)$, and both negative passage rows be $\mathbf{e}_2$. Full-capacity scores are 1 and 0, while $(1, 1)$ scores are -1 and 0: the prefix reverses the full ranking. Swapping the two passage rows leaves all full scores unchanged but restores the correct prefix order. All rows are unit norm. □

This is a property of the scoring objective on encoded banks, not a claim that the DLM is permutation equivariant or that its positional encoding makes masks indistinguishable. Prefix supervision removes the objective's general indifference to exchanges across supervised boundaries. It still permits permutations within blocks that lie wholly inside or outside every supervised prefix, and it does not uniquely identify individual vector semantics.

C.3 Geometric Conditions for Stable Truncation

Two different geometric quantities arise because `MaxSim` *averages* query contributions but *maximizes* over passage rows. Query coverage must preserve the mass of the averaged contributions, whereas passage coverage must preserve potential maximizers.

Query transport discrepancy. Define the feasible couplings

$$\mathcal{T}_a = \left\{ \mathbf{\Gamma} \in \mathbb{R}_{\geq 0}^{K_q \times a} : \sum_{\ell=1}^{a} \Gamma_{i\ell} = \frac{1}{K_q}, \quad \sum_{i=1}^{K_q} \Gamma_{i\ell} = \frac{1}{a} \right\},$$
$$\epsilon_q(a; q) = \min_{\mathbf{\Gamma} \in \mathcal{T}_a} \sum_{i=1}^{K_q} \sum_{\ell=1}^{a} \Gamma_{i\ell} \|\mathbf{r}_{q,i} - \mathbf{r}_{q,\ell}\|_2. \tag{15}$$

Here $\Gamma_{i\ell}$ is a scalar entry of the nonnegative transport matrix. The row and column constraints couple the uniform full query bank with the uniform retained bank. A feasible coupling always exists, and the minimum is attained because the feasible set is compact. This is a definition for analysis; the algorithm does not solve a transport problem.

Passage covering radius. Define

$$\epsilon_p(b; p) = \max_{1 \leq j \leq K_p} \min_{1 \leq \ell \leq b} \|\mathbf{r}_{p,j} - \mathbf{r}_{p,\ell}\|_2. \tag{16}$$

It measures how far any full passage vector can lie from the retained bank. The universal bounds are $\epsilon_q(a; q) \leq 2(1 - a/K_q)$ and $\epsilon_p(b; p) \leq 2$, and both quantities vanish at their respective full capacities. To obtain the first bound, match mass $1/K_q$ on each retained row to itself and transport only the remaining mass $1 - a/K_q$, at distance at most two. The bounds can be loose; effective small prefixes require more structure than normalization alone.

Theorem C.2 (Stability of query-averaged late interaction)**.** *For the fixed, unit-norm banks above and any retained pair (a, b),*

$$-\epsilon_q(a; q) \leq s_{K_q,K_p}(q, p) - s_{a,b}(q, p) \leq \epsilon_q(a; q) + \epsilon_p(b; p). \tag{17}$$

In particular, (10) holds, and its right-hand side may be capped at two because every averaged score lies in $[-1, 1]$.

Proof. For fixed p, define $f_p^{(b)}(\mathbf{z}) = \max_{j \leq b} \mathbf{z}^\top \mathbf{r}_{p,j}$. Each function is 1-Lipschitz in the Euclidean norm:

$$|f_p^{(b)}(\mathbf{z}) - f_p^{(b)}(\mathbf{z}')| \leq \max_{j \leq b} |(\mathbf{z} - \mathbf{z}')^\top \mathbf{r}_{p,j}| \leq \|\mathbf{z} - \mathbf{z}'\|_2. \tag{18}$$

For any $\mathbf{\Gamma} \in \mathcal{T}_a$, its marginal constraints give

$$s_{K_q,K_p} - s_{a,K_p} = \sum_{i=1}^{K_q} \sum_{\ell=1}^{a} \Gamma_{i\ell} \left(f_p^{(K_p)}(\mathbf{r}_{q,i}) - f_p^{(K_p)}(\mathbf{r}_{q,\ell}) \right), \tag{19}$$

$$|s_{K_q,K_p} - s_{a,K_p}| \leq \sum_{i,\ell} \Gamma_{i\ell} \|\mathbf{r}_{q,i} - \mathbf{r}_{q,\ell}\|_2. \tag{20}$$

Minimizing over feasible couplings bounds this difference by $\epsilon_q(a;q)$.

For any unit vector $\mathbf{z}$, choose a full-bank maximizing row $j^\star$. A nearest retained row $\ell^\star$ satisfies

$$0 \le f_p^{(K_p)}(\mathbf{z}) - f_p^{(b)}(\mathbf{z}) \le \mathbf{z}^\top(\mathbf{r}_{p,j^\star} - \mathbf{r}_{p,\ell^\star}) \le \epsilon_p(b;p). \tag{21}$$

The first inequality follows from set inclusion; the last follows from Cauchy–Schwarz. Averaging over the retained query rows yields $0 \le s_{a,K_p} - s_{a,b} \le \epsilon_p(b;p)$. Adding this nonnegative difference to the query discrepancy proves both sides of (17). □

Corollary C.3 (Pairwise and top-J ranking preservation)**.** *Let* $\gamma(q;p^+,p^-) = s_{K_q,K_p}(q,p^+) - s_{K_q,K_p}(q,p^-) > 0$. *If*

$$\gamma(q;p^+,p^-) > \epsilon_{a,b}(q,p^+) + \epsilon_{a,b}(q,p^-), \tag{22}$$

then $s_{a,b}(q,p^+) > s_{a,b}(q,p^-)$. *A sharper sufficient condition is*

$$\gamma(q;p^+,p^-) > 2\epsilon_q(a;q) + \epsilon_p(b;p^+). \tag{23}$$

For a finite candidate collection, let $\bar{\epsilon}_{a,b}(q)$ *be the maximum of* $\epsilon_{a,b}(q,p)$ *over that collection. If the full-budget gap between ranks* J *and* $J+1$ *exceeds* $2\bar{\epsilon}_{a,b}(q)$*, then the top-*J set *is unchanged.*

Proof. Subtract the two bounds on score perturbation in (10) to obtain a prefix margin of at least $\gamma - \epsilon_{a,b}(q,p^+) - \epsilon_{a,b}(q,p^-)$. For the sharper version, (17) gives $s_{a,b}(q,p^+) \ge s_{K_q,K_p}(q,p^+) - \epsilon_q - \epsilon_p(b;p^+)$ and $s_{a,b}(q,p^-) \le s_{K_q,K_p}(q,p^-) + \epsilon_q$. Their difference yields (11) and proves (23). For the top-J statement, every full top-J candidate exceeds every outside candidate by at least the boundary gap, so no such pair can reverse. □

Top-J set preservation alone does not imply identical within-set order or identical nDCG. If the margin condition holds for every ordered pair in a strict full ranking, then the entire order, and hence any metric depending only on that order and fixed relevance labels, is preserved. These are sufficient conditions rather than necessary ones: small prefixes can work well even when the uniform geometric bound is loose. Conversely, prefix contrastive training is not asserted to minimize either geometric discrepancy explicitly.

Non-monotonic rankings across budgets. At fixed query budget, enlarging the passage bank cannot decrease an individual `MaxSim` score, but it can increase an irrelevant score more than a relevant one. For example, let the query be $\mathbf{e}_1$, the positive passage contain two copies of $\frac{1}{2}\mathbf{e}_1 + \frac{\sqrt{3}}{2}\mathbf{e}_2$, and the negative passage contain $(\mathbf{e}_2, \mathbf{e}_1)$. Increasing the passage budget from one to two changes the positive/negative scores from $(1/2, 0)$ to $(1/2, 1)$, reversing the ranking. Increasing the query budget changes the average itself: for query rows $(\mathbf{e}_1, \mathbf{e}_2)$, a one-row positive $\mathbf{e}_1$ and negative $\frac{3}{5}\mathbf{e}_1 + \frac{4}{5}\mathbf{e}_2$ score $(1, 3/5)$ with one query row but $(1/2, 7/10)$ with both. These exact counterexamples establish that budget monotonicity is not a property of the scoring rule. They illustrate possible mechanisms, not a causal diagnosis of any particular experimental run.

C.4 From Prefix Contrastive Loss to Ranking Error

For query q_n and its designated positive p_n^+, write

$$\ell_{a,b,n} = \log\left(1 + \sum_{p \in \mathcal{P}\setminus\{p_n^+\}} \exp\left(\frac{s_{a,b}(q_n,p) - s_{a,b}(q_n,p_n^+)}{\tau}\right)\right), \qquad \mathcal{L}_{a,b} = \frac{1}{B}\sum_{n=1}^{B} \ell_{a,b,n}. \tag{24}$$

Here $\mathcal{P}$ is the indexed candidate collection used in the loss, with at least one non-designated candidate. "Negative" refers to these training candidates, not a claim that every unlabelled passage is truly irrelevant. For a desired margin $\gamma \ge 0$, define

$$E_{a,b}(\gamma) = \frac{1}{B}\sum_{n=1}^{B} \mathbf{1}\left\{ s_{a,b}(q_n,p_n^+) - \max_{p \in \mathcal{P}\setminus\{p_n^+\}} s_{a,b}(q_n,p) \le \gamma \right\}. \tag{25}$$

The quantity $E_{a,b}$ in section 4.4 is $E_{a,b}(0)$, counting ties as failures.

Proposition C.4 (Finite-candidate, budget-wise surrogate control)**.** *For $\tau > 0$ and $\gamma \geq 0$, put $d_\gamma = \log(1 + e^{-\gamma/\tau}) > 0$. Then*

$$E_{a,b}(\gamma) \leq \min\{1, \mathcal{L}_{a,b}/d_\gamma\}, \tag{26}$$

$$\frac{1}{|\mathcal{B}|} \sum_{(a,b)\in\mathcal{B}} E_{a,b}(\gamma) \leq \min\{1, \mathcal{L}_{\text{prefix}}/d_\gamma\}, \qquad \max_{(a,b)\in\mathcal{B}} E_{a,b}(\gamma) \leq \min\{1, |\mathcal{B}|\mathcal{L}_{\text{prefix}}/d_\gamma\}. \tag{27}$$

The full-budget bound has the same form with $(a, b) = (K_q, K_p)$, yielding (12) at $\gamma = 0$.

Proof. If a query violates the requested margin, at least one negative has $s_{a,b}(q_n, p) - s_{a,b}(q_n, p_n^+) \geq -\gamma$. Its term alone makes $\ell_{a,b,n} \geq \log(1 + e^{-\gamma/\tau}) = d_\gamma$. Thus the indicator in (25) is at most $\ell_{a,b,n}/d_\gamma$. Average over queries and then budgets. Finally, nonnegativity gives $\mathcal{L}_{a,b} \leq |\mathcal{B}|\mathcal{L}_{\text{prefix}}$ for each trained smaller budget. Each error fraction is also at most one. □

There is also an individual-query margin certificate. For any non-designated candidate,

$$s_{a,b}(q_n, p_n^+) - s_{a,b}(q_n, p) \geq -\tau \log\left(e^{\ell_{a,b,n}} - 1\right). \tag{28}$$

Indeed, that negative's exponential term is at most $e^{\ell_{a,b,n}} - 1$. A per-query loss below $\log 2$ therefore implies a strictly positive margin against every candidate in that loss. A small *average* loss does not certify every individual query; (27) is the corresponding aggregate statement.

Curriculum as a budget mixture. For fixed progress u, abbreviate $\lambda = \lambda_{\text{mrl}}(u) \geq 0$ and define

$$\nu_u(K_q, K_p) = \frac{1}{1+\lambda}, \qquad \nu_u(a, b) = \frac{\lambda}{(1+\lambda)|\mathcal{B}|} \quad ((a, b) \in \mathcal{B}). \tag{29}$$

Since the smaller budgets exclude the full pair, these masses sum to one. The dense part of (8) equals $(1 + \lambda)\mathbb{E}_{(a,b)\sim\nu_u}\mathcal{L}_{a,b}$. Hence, if $\lambda_{\text{sp}} \geq 0$,

$$\mathbb{E}_{(a,b)\sim\nu_u} E_{a,b}(\gamma) \leq \frac{\mathcal{L}_{K_q,K_p} + \lambda\mathcal{L}_{\text{prefix}}}{(1+\lambda)d_\gamma} \leq \frac{\mathcal{L}(u)}{(1+\lambda)d_\gamma}. \tag{30}$$

At warm-up, all mixture mass is on the full budget; subsequently, smaller budgets gain mass without removing its loss. This identity characterizes which ranking errors are supervised. It does not ensure that nonconvex training decreases all constituent losses, nor does it transfer a sampled-negative bound to the full corpus, an unseen domain, or a budget outside $\mathcal{B}$.

C.5 Initialization and Curriculum Properties

Proposition C.5 (Offset moments and distinguishability at initialization)**.** *For independent $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_h)$ with $\sigma > 0$ and $c_j > 0$, the initializer (3) satisfies, for $j \neq \ell$,*

$$\mathbb{E}\|\mathbf{A}[j, :]\|_2^2 = h\sigma^2 c_j^2, \qquad \mathbb{E}\|\mathbf{A}[j, :] - \mathbf{A}[\ell, :]\|_2^2 = h\sigma^2(c_j^2 + c_\ell^2). \tag{31}$$

Distinct rows are unequal with probability one in exact arithmetic. Descending c_j imposes an order only on the expected initial squared norms, including within either tail-aligned role subsequence $\pi_r(i)$.

Proof. The covariance of row j is $\sigma^2 c_j^2\mathbf{I}_h$, whose trace gives the first identity. Independence and zero means eliminate the cross term in the second. The row difference has nondegenerate Gaussian covariance $\sigma^2(c_j^2 + c_\ell^2)\mathbf{I}_h$, so its probability of being exactly zero is zero. The map $i \mapsto \pi_r(i)$ is increasing, preserving the expected-norm ordering within each role. □

The proposition concerns the additive offsets, not the mask embedding plus offset, the trained offset norms, or semantic content. A nonlinear encoder may still map distinct inputs to identical or redundant outputs. Thus, initialization supplies slot-specific parameters and a scale bias; discriminative prefix behavior must be learned from the retrieval objectives.

Controlled introduction of the auxiliary gradient. Let $\boldsymbol{\omega} = (\boldsymbol{\phi}, \mathrm{vec}(\mathbf{A}))$ collect the trainable parameters, and set $\mathcal{L}_0 = \mathcal{L}_{K_q,K_p} + \lambda_{\mathrm{sp}}\mathcal{L}_{\mathrm{sp}}$. At a differentiable point, holding u fixed,

$$\nabla_{\boldsymbol{\omega}}\mathcal{L}(u) - \nabla_{\boldsymbol{\omega}}\mathcal{L}_0 = \lambda_{\mathrm{mrl}}(u)\nabla_{\boldsymbol{\omega}}\mathcal{L}_{\mathrm{prefix}}. \tag{32}$$

If the prefix-gradient norm at this point is at most G, the added raw-gradient norm is at most $\lambda_{\mathrm{mrl}}(u)G$. The same algebra holds for consistently selected backpropagation subgradients at maximum ties. For $0 < \rho < 1$, the schedule is continuous, zero before ρ, and increases with slope $\lambda_{\max}/(1-\rho)$ afterward. Its integrated weight is exactly

$$\int_0^1 \lambda_{\mathrm{mrl}}(u)\, du = \frac{\lambda_{\max}(1-\rho)}{2}. \tag{33}$$

This supports describing the schedule as a gradual introduction of prefix pressure. It does not prove gradient alignment, convergence, a bound on AdamW parameter updates, or superiority over constant weighting. Changing ρ also changes the integrated loss weight, so the warm-up sweep does not isolate timing from total auxiliary supervision.

C.6 Evaluation Conventions and Limits of the Guarantees

For a common nondecreasing 1-Lipschitz scalar function ψ, define

$$s^{\psi}_{a,b}(q,p) = \frac{1}{a}\sum_{i=1}^{a} \psi\left(\max_{j\le b} \mathbf{r}_{q,i}^{\top}\mathbf{r}_{p,j}\right). \tag{34}$$

Both $\psi(z) = z$ and $\psi(z) = \max(0, z)$ meet these conditions. Theorem C.2 and Corollary C.3 extend to s^{ψ}: composition preserves the Lipschitz inequality, while monotonicity preserves the nonnegative passage-truncation difference. The evaluator's $s^{+}_{a,b} = a\, s^{\psi}_{a,b}$ for the clipped choice has the same ranking as $s^{\psi}_{a,b}$ at a fixed query budget. Bounds comparing different budgets must use these averaged scores, not raw sums with different multipliers.

The $(1,1)$ evaluation branch instead uses the unclipped score. A comparison that crosses this convention change requires an extra term. With $[z]_+ = \max(0, z)$, define

$$\chi_{a,b}(q,p) = \frac{1}{a}\sum_{i=1}^{a}\left[-\max_{j\le b}\mathbf{r}_{q,i}^{\top}\mathbf{r}_{p,j}\right]_+. \tag{35}$$

It is exactly the difference between clipped and unclipped averaged scores. Therefore,

$$|s_{1,1}(q,p) - s^{\psi}_{K_q,K_p}(q,p)| \le \epsilon_{1,1}(q,p) + \chi_{1,1}(q,p), \qquad \psi(z) = [z]_+. \tag{36}$$

This follows by inserting the clipped $(1,1)$ score and applying the triangle inequality. When its dot product is nonnegative, the additional term is zero. The loss-to-error bounds concern the training score; they must not be applied unchanged to a different clipping rule or to hybrid candidate fusion.

Finally, the storage and exhaustive matching formulas in (9) follow directly from storing k_p rows of width h per passage and computing $k_q k_p$ inner products per query–passage pair. At the same precision their ratios to full capacity are k_p/K_p and $k_q k_p/(K_q K_p)$, respectively. These are representation payload and arithmetic-count statements, not measurements of index overhead, approximate-search complexity, encoding cost, or end-to-end latency. Together, the analysis supports the design of supervised, reusable prefixes while leaving their achieved geometric coverage, held-out accuracy, and system efficiency to empirical evaluation.

D More Experimental Analysis

D.1 Complete Budget Results

Table 4 reports dense retrieval at five post-encoding budgets from one Light-DiffRetriever checkpoint per backbone. At $(4,4)$, dense-vector storage and pairwise interactions are both 25% of full capacity; MS MARCO/BEIR-7 scores are .398/.649 for Dream and .381/.635 for LLaDA-1.5, compared with .405/.660 and .388/.649 at $(4,16)$. At $(2,2)$, storage falls to 12.5% and interactions

to 6.25%, while BEIR-7 reaches .642/.615 on Dream/LLaDA-1.5. The $(1, 1)$ budget yields .597 on BEIR-7 for both backbones, indicating a larger accuracy trade-off at extreme compression. Performance need not increase monotonically with vector count: Dream DL19 changes from .750 at $(4, 4)$ to .747 at $(4, 8)$. These point estimates characterize budget-dependent trade-offs; the resource fractions concern storage and matching, not encoding costs or measured end-to-end latency.

Table 4: Dense retrieval from each main checkpoint across evaluated prefix budgets. Storage and interactions are analytical fractions of the $(4, 16)$ dense representation, distinct from measured end-to-end speedups.

Backbone	(k_q, k_p)	Storage	Interactions	MS MARCO	DL19	DL20	BEIR-7
Dream	$(1, 1)$	6.25%	1.5625%	.367	.708	.693	.597
Dream	$(2, 2)$	12.5%	6.25%	.387	.734	.723	.642
Dream	$(4, 4)$	25%	25%	.398	.750	.732	.649
Dream	$(4, 8)$	50%	50%	.403	.747	.739	.655
Dream	$(4, 16)$	100%	100%	.405	.752	.742	.660
LLaDA-1.5	$(1, 1)$	6.25%	1.5625%	.357	.662	.702	.597
LLaDA-1.5	$(2, 2)$	12.5%	6.25%	.373	.699	.713	.615
LLaDA-1.5	$(4, 4)$	25%	25%	.381	.700	.718	.635
LLaDA-1.5	$(4, 8)$	50%	50%	.387	.702	.730	.643
LLaDA-1.5	$(4, 16)$	100%	100%	.388	.719	.731	.649

Table 5: Full-capacity encoding: batch-1 query latency, batch-32 query throughput, and batch-64 passage throughput. GPU peaks cover passage encoding. Tokenization and input transfer precede the encoding timer.

Method	Q p50 (ms)	Q p95 (ms)	Q/s	P/s	Alloc. (GB)	Reserved (GB)
Dream backbone						
DiffRetriever	26.41	30.66	176.63	82.67	19.45	22.42
LIGHT-DIFFRETRIEVER	25.08	29.03	178.13	83.12	19.45	22.42
LLaDA-1.5 backbone						
DiffRetriever	30.24	36.73	124.83	58.22	19.59	21.40
LIGHT-DIFFRETRIEVER	28.61	31.76	124.41	58.19	19.59	21.65

D.2 EFFICIENCY PROTOCOL AND MEASUREMENT SCOPE

Hardware and workloads. The measurements use an eight-GPU NVIDIA A100-SXM4-80GB host with CUDA 12.6, PyTorch 2.7.1, Transformers 4.55.0, and FlashAttention 2.8.0.post2. Each timed configuration uses one GPU; independent configurations are distributed across the host, not jointly parallelized within a query. The passage and query pools are sampled with seed 42: 100,000 passages from 8,841,823 MS MARCO passages and 500 queries from the development query file. Single-query matching and pipeline measurements use the first 100 sampled queries after 20 warm-up iterations. Batch-32 matching uses 5 warm-up and 30 measured batches. CPU resources are shared across concurrent jobs. Latency percentiles summarize within-sequence query variation.

Encoding and matching. All checkpoints encode at $(K_q, K_p) = (4, 16)$ in one pass with the saved retrieval prompt. The encoder uses bf16, whereas exported representative banks and dense matching use fp32, with $h = 3584$ for Dream and $h = 4096$ for LLaDA-1.5. Budgets $(1, 1)$, $(2, 2)$, $(4, 4)$, $(4, 8)$, and $(4, 16)$ select output prefixes without re-encoding. Dense search exhaustively scores all 100,000 passages in chunks of 8,192; CUDA-event timing includes top-1,000 selection. Sparse matching uses a CPU inverted index with up to 128 retained terms per input. This is an exact-search workload, not ANN retrieval or reranking only 1,000 preselected candidates. Table 5 separates full-capacity encoding from subsequent matching.

Auxiliary serving pipeline. Wall-clock timing starts before query tokenization and ends after returning the top ten from equal-weight, independently min–max-normalized dense and sparse scores. Model and index loading are excluded from warm measurements. The timed pipeline is a deployment-cost probe, not the main-table hybrid evaluator: it fuses scores over the entire sampled corpus rather than the union of two top-1,000 lists, and its online query encoding does not supply the content-token filter used for cached sparse encodings. The shared dense timing kernel sums un-clamped maxima, whereas the main multi-vector evaluator floors negative maxima at zero (Appendix B.1). Sparse vectors remain pooled over full capacity as dense prefixes change. These distinctions are shared by both compared methods. We therefore pair dense full-corpus effectiveness with dense timing, and report hybrid-pipeline latency separately without attributing a measured hybrid accuracy to it.

Interpretation. The large single-query speedups need not equal the fourfold reduction in pairwise interactions: chunked tensor contractions and memory access depend on shape, and batch-32 timings exhibit different ratios. The reported ratios characterize workload-level performance rather than identifying a specific kernel-level cause. At full capacity, passage-encoding throughput is 82.67/83.12 passages/s for Dream baseline/ours and 58.22/58.19 for LLaDA-1.5, consistent with unchanged encoding work. The measured speedups apply to exact search over the specified 100K-passage corpus on an A100; corpus size, search engine, and hardware determine their deployment context.

Table 6: All timed budgets on 100K passages (ms). Dense/hybrid matching use batch size 1; B32 is dense time per query in a batch of 32. Pipeline includes online query preparation, encoding, matching, and fusion.

Method	Budget	Dense p50	Hybrid p50	B32 p50	Pipeline p50	p95	p99
Dream backbone							
DiffRetriever	(1, 1)	1.45	3.08	0.08	30.68	34.45	52.33
	(2, 2)	10.86	12.60	0.49	39.06	42.08	45.09
	(4, 4)	26.54	28.94	1.37	56.14	62.94	66.17
	(4, 8)	71.06	73.14	3.27	100.43	107.25	119.62
	(4, 16)	203.82	206.08	8.40	233.50	249.56	263.15
LIGHT-DIFFRETRIEVER	(1, 1)	1.45	2.98	0.08	30.61	37.64	38.39
	(2, 2)	10.87	12.56	0.49	39.04	41.56	44.37
	(4, 4)	26.52	28.42	1.37	55.86	61.92	70.20
	(4, 8)	71.14	73.35	3.28	104.77	122.01	139.17
	(4, 16)	203.28	205.39	8.37	233.88	245.88	268.61
LLaDA-1.5 backbone							
DiffRetriever	(1, 1)	1.60	3.24	0.09	38.24	50.71	87.94
	(2, 2)	12.32	14.05	0.55	44.78	50.94	54.45
	(4, 4)	25.24	27.51	1.40	58.80	64.35	67.77
	(4, 8)	74.86	77.22	3.54	107.99	114.87	118.27
	(4, 16)	271.28	273.66	10.78	304.88	312.62	329.89
LIGHT-DIFFRETRIEVER	(1, 1)	1.61	3.18	0.08	35.37	45.01	86.24
	(2, 2)	12.31	14.10	0.55	45.25	51.13	51.90
	(4, 4)	25.20	27.19	1.40	58.09	59.71	60.37
	(4, 8)	74.82	76.99	3.54	108.07	111.09	112.84
	(4, 16)	271.12	273.77	10.78	305.86	309.56	311.50

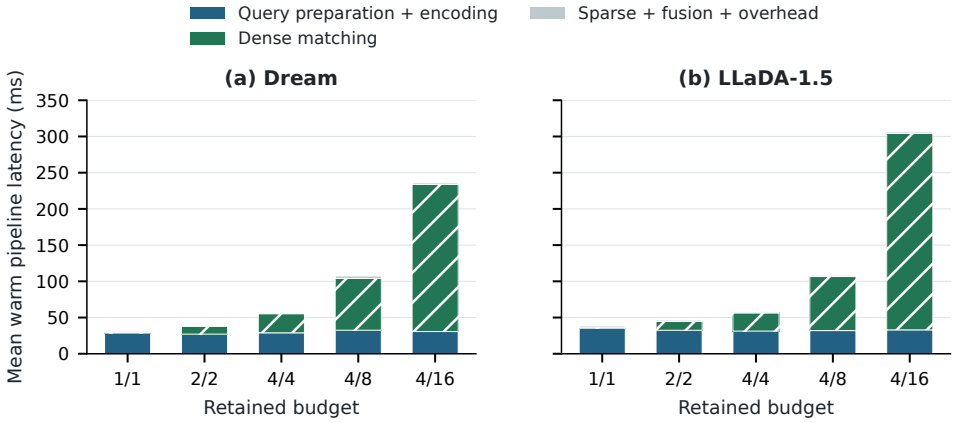


Figure 9: Mean warm-pipeline components for LIGHT-DIFFRETRIEVER. Full-capacity query encoding becomes the dominant cost as the retained dense budget shrinks.

D.3 COMPLETE EFFICIENCY MEASUREMENTS

Table 6 reports all configurations, including batched matching and latency tails. At $(4, 4)$, baseline/ours dense latency is 26.54/26.52 ms on Dream and 25.24/25.20 ms on LLaDA-1.5, confirming comparable same-budget computation. At $(1, 1)$, our matching falls to 1.45/1.61 ms, but pipeline latency remains 30.61/35.37 ms because query preparation and encoding dominate. LLaDA-1.5's $(1, 1)$ p99 reaches 86.24 ms: a small median does not imply an equally small service tail.

Figure 9 uses arithmetic means so components add to the measured mean total; marginal medians would not. The residual includes sparse matching, fusion, and synchronization/timing overhead. At $(4, 4)$, query preparation and encoding account for 50.8%/53.8% of Dream/LLaDA-1.5 pipeline time, locating the remaining bottleneck without implying a shorter backbone forward pass.

Table 7: Storage and index measurements; GB uses decimal units. File includes one quotation vector per passage; directory includes query caches and sparse files. Peak is scoring-process GPU allocation, including batch-32 trials, not full-service memory.

Method	Budget	Repr. (GB)	File (GB)	Directory (GB)	Peak (GB)	Build (s)
Dream backbone						
DiffRetriever	$(1, 1)$	1.43	2.87	3.00	1.47	3.27
	$(2, 2)$	2.87	4.30	4.44	3.15	3.43
	$(4, 4)$	5.73	7.17	7.32	6.29	3.32
	$(4, 8)$	11.47	12.90	13.05	12.53	4.35
	$(4, 16)$	22.94	24.37	24.52	25.00	3.12
LIGHT-DIFFRETRIEVER	$(1, 1)$	1.43	2.87	3.00	1.47	3.16
	$(2, 2)$	2.87	4.30	4.44	3.15	3.08
	$(4, 4)$	5.73	7.17	7.32	6.29	3.07
	$(4, 8)$	11.47	12.90	13.05	12.53	2.83
	$(4, 16)$	22.94	24.37	24.52	25.00	2.83
LLaDA-1.5 backbone						
DiffRetriever	$(1, 1)$	1.64	3.28	3.41	1.67	2.99
	$(2, 2)$	3.28	4.92	5.06	3.59	3.10
	$(4, 4)$	6.55	8.19	8.35	7.18	37.58
	$(4, 8)$	13.11	14.75	14.90	14.31	3.34
	$(4, 16)$	26.21	27.85	28.01	28.55	3.07
LIGHT-DIFFRETRIEVER	$(1, 1)$	1.64	3.28	3.41	1.67	3.01
	$(2, 2)$	3.28	4.92	5.06	3.59	3.15
	$(4, 4)$	6.55	8.19	8.35	7.18	3.38
	$(4, 8)$	13.11	14.75	14.90	14.31	3.27
	$(4, 16)$	26.21	27.85	28.01	28.55	3.08

Storage accounting. Equation 9 counts only representative vectors. The serialized dense passage file additionally stores one fp32 quotation vector per passage, giving approximately $N(k_p + 1)h\, b_{\text{elt}}$ bytes, with $b_{\text{elt}} = 4$, plus a small serialization header. Consequently, reducing k_p from 16 to 4 yields a file-size ratio of $17/5 = 3.40$, rather than the representative-payload ratio of $16/4 = 4$. The quotation vector is not read by the timed `MaxSim` kernel. The directory totals in Table 7 additionally include cached query banks, passage/query IDs, sparse top-128 tensors, and the compressed inverted index; they are not minimal passage-only deployment requirements. The sparse passage tensor occupies 102.40 MB per configuration, while compressed inverted files occupy 11.20–12.15 MB and do not shrink with the dense budget.

GPU allocation and index construction. The reported scoring peak includes resident query/passage banks and the largest temporary allocation observed across single-query and batch-32 trials; it excludes model weights because scoring is measured in a separate process. It therefore cannot be interpreted as total serving memory. Our $(4, 4)$ scoring peaks are 6.29/7.18 GB, compared with 25.00/28.55 GB for the full-budget baseline. The recorded build time covers construction and compression of the sparse inverted index, not complete end-to-end indexing. Most observations lie between 2.83 and 4.35 s; the 37.58 s LLaDA-1.5 baseline observation is retained rather than removed. Raw timing summaries, including means, standard deviations, and maxima, accompany these tables in the derived data files.

Table 8: Loading and first-query measurements by configuration. Model/index loading is in seconds; first-query latency and the warm-pipeline median are in milliseconds.

Method	Budget	Model load (s)	Index load (s)	First query (ms)	Warm p50 (ms)
Dream backbone					
DiffRetriever	$(1, 1)$	47.51	0.42	513.02	30.68
	$(2, 2)$	49.46	0.70	390.16	39.06
	$(4, 4)$	49.54	1.28	396.07	56.14
	$(4, 8)$	49.45	2.06	425.83	100.43
	$(4, 16)$	49.08	3.68	549.95	233.50
LIGHT-DIFFRETRIEVER	$(1, 1)$	47.51	0.48	505.47	30.61
	$(2, 2)$	49.57	0.74	404.28	39.04
	$(4, 4)$	49.59	1.23	434.10	55.86
	$(4, 8)$	34.79	2.62	619.73	104.77
	$(4, 16)$	34.82	4.30	1067.25	233.88
LLaDA-1.5 backbone					
DiffRetriever	$(1, 1)$	46.89	0.41	398.28	38.24
	$(2, 2)$	39.92	0.72	372.01	44.78
	$(4, 4)$	18.91	1.13	407.75	58.80
	$(4, 8)$	49.86	2.51	524.06	107.99
	$(4, 16)$	49.06	4.74	716.77	304.88
LIGHT-DIFFRETRIEVER	$(1, 1)$	46.90	0.44	438.99	35.37
	$(2, 2)$	24.10	0.67	393.26	45.25
	$(4, 4)$	13.35	1.06	364.14	58.09
	$(4, 8)$	23.99	2.12	421.44	108.07
	$(4, 16)$	21.37	4.40	668.07	305.86

D.4 LOADING AND FIRST-QUERY BEHAVIOR

The field called `cold_start` in the measurement output times only the first query *after* loading the model and index. Table 8 therefore separates model loading, index loading, first-query latency, and warmed latency. For our Dream checkpoint, the recorded index-load time decreases from 4.30 s at $(4, 16)$ to 1.23 s at $(4, 4)$; for LLaDA-1.5 it decreases from 4.40 to 1.06 s. First-query latency is substantially larger than the warmed median, but neither the first-query measurement nor model loading is monotonic across every budget. Model loading varies from 13.35 to 49.86 s across configurations and is unrelated to the retained output prefix. Startup behavior depends on loading, cache state, and concurrent I/O, whereas retained-prefix savings act on matching. Main-text speedups therefore compare warmed, repeated-query timings under matched workloads.

Table 9: MS MARCO retention and top-10 collapse rates. Agg. is the ratio of aggregate MRR; Cond. is the mean per-query ratio over nonzero full-budget scores. Own uses method-specific strong queries; matched uses the common strong cohort.

Method	Budget	Agg.	Cond.	Own collapse	Matched count	Matched rate
Dream backbone						
DiffRetriever	$(1, 1)$	70.8%	0.757	18.65%	3,030/18,775	16.14%
	$(2, 2)$	59.8%	0.661	28.84%	4,893/18,775	26.06%
	$(4, 4)$	49.7%	0.554	39.27%	6,911/18,775	36.81%
	$(4, 8)$	49.7%	0.554	39.26%	6,908/18,775	36.79%
LIGHT-DIFFRETRIEVER	$(1, 1)$	90.6%	0.944	4.31%	533/18,775	2.84%
	$(2, 2)$	95.6%	0.989	1.48%	146/18,775	0.78%
	$(4, 4)$	98.2%	1.006	0.55%	41/18,775	0.22%
	$(4, 8)$	99.5%	1.005	0.21%	13/18,775	0.07%
LLaDA-1.5 backbone						
DiffRetriever	$(1, 1)$	72.8%	0.749	17.35%	2,927/18,691	15.66%
	$(2, 2)$	84.6%	0.873	8.61%	1,400/18,691	7.49%
	$(4, 4)$	94.6%	0.989	3.34%	492/18,691	2.63%
	$(4, 8)$	98.5%	1.005	1.20%	168/18,691	0.90%
LIGHT-DIFFRETRIEVER	$(1, 1)$	92.0%	0.949	3.36%	476/18,691	2.55%
	$(2, 2)$	96.0%	0.986	1.59%	216/18,691	1.16%
	$(4, 4)$	98.0%	0.999	0.59%	71/18,691	0.38%
	$(4, 8)$	99.6%	1.007	0.26%	26/18,691	0.14%

E CASE STUDY

E.1 PER-QUERY ROBUSTNESS AND COHORT DEFINITIONS

We analyze dense per-query evaluation records, independently of the 100K-passage timing workload. The analysis comprises 1,262,588 query–budget records across two methods, two backbones, and four reduced budgets. Each MS MARCO configuration contains 55,578 queries. Per-query comparisons cover MS MARCO and the seven BEIR tasks for both backbones, with DL19/20 additionally included for Dream. LLaDA-1.5 DL19/20 comparisons use the aggregate metrics in Tables 1 and 4; Table 10 also lists our model's per-query diagnostics.

Let $\mathrm{RR}_m^{(a,b)}(q)$ denote RR@10 for method $m \in \{\mathrm{base}, \mathrm{light}\}$ at budget (a, b). Separately for each backbone, the matched strong-query cohort is

$$\mathcal{Q}_* = \left\{ q : \mathrm{RR}_{\mathrm{base}}^{(K_q,K_p)}(q) \geq \tfrac{1}{2},\ \mathrm{RR}_{\mathrm{light}}^{(K_q,K_p)}(q) \geq \tfrac{1}{2} \right\}. \tag{37}$$

The collapse rate for method m at a reduced budget is

$$\frac{1}{|\mathcal{Q}_*|} \sum_{q \in \mathcal{Q}_*} \mathbf{1}\left\{ \mathrm{RR}_m^{(a,b)}(q) = 0 \right\}. \tag{38}$$

Here "collapse" means no judged-relevant result in the first ten, not proof that all retrieved passages are irrelevant. The common cohort contains 18,775 Dream queries and 18,691 LLaDA-1.5 queries. Figure 7 uses this denominator for both methods, whereas the original method-specific diagnostic conditions only on that method's full-budget RR@10. Table 9 reports both definitions explicitly.

Under the method-specific definition, Dream's $(4, 4)$ collapse rate is 8,633/21,983 = 39.27% for the baseline and 124/22,572 = 0.55% for ours. These denominators differ; the matched-cohort result, 6,911/18,775 versus 41/18,775, is the controlled comparison in the main text. LLaDA-1.5 shows a smaller but consistent improvement under either definition. Across all MS MARCO queries at $(4, 4)$, Dream baseline/ours have 27,425/7,751 decreases, 22,014/40,483 ties, and 6,139/7,344 increases relative to their own full-budget RR@10. LLaDA-1.5 has 9,702/7,548 decreases, 37,599/41,175 ties, and 8,277/6,855 increases. Ties include queries scored zero at both budgets, and are not all retrieval successes.

Table 10: Conditional per-query retention at $(4, 4)$, with eligible query counts in parentheses. Each method conditions on its own nonzero full-budget scores; dashes denote aggregate-only evaluation.

Dataset	Dream / base	Dream / ours	LLaDA-1.5 / base	LLaDA-1.5 / ours
MS MARCO	0.554 (38,275)	1.006 (39,029)	0.989 (37,444)	0.999 (37,713)
DL19	0.699 (43)	1.001 (43)	—	0.980 (43)
DL20	0.682 (52)	0.984 (52)	—	0.983 (52)
NQ	0.384 (2,975)	0.975 (2,997)	0.837 (2,963)	0.950 (2,995)
HotpotQA	0.413 (7,007)	0.943 (7,048)	0.867 (7,043)	0.944 (7,019)
SciFact	0.815 (261)	0.994 (263)	0.872 (264)	1.000 (259)
TREC-COVID	0.852 (50)	1.038 (49)	1.064 (49)	1.052 (50)
FiQA	0.697 (483)	0.980 (492)	0.936 (480)	0.949 (473)
ArguAna	0.846 (1,152)	0.977 (1,124)	0.888 (1,113)	0.975 (1,090)
Quora	0.949 (9,775)	0.996 (9,775)	0.909 (9,674)	0.970 (9,579)

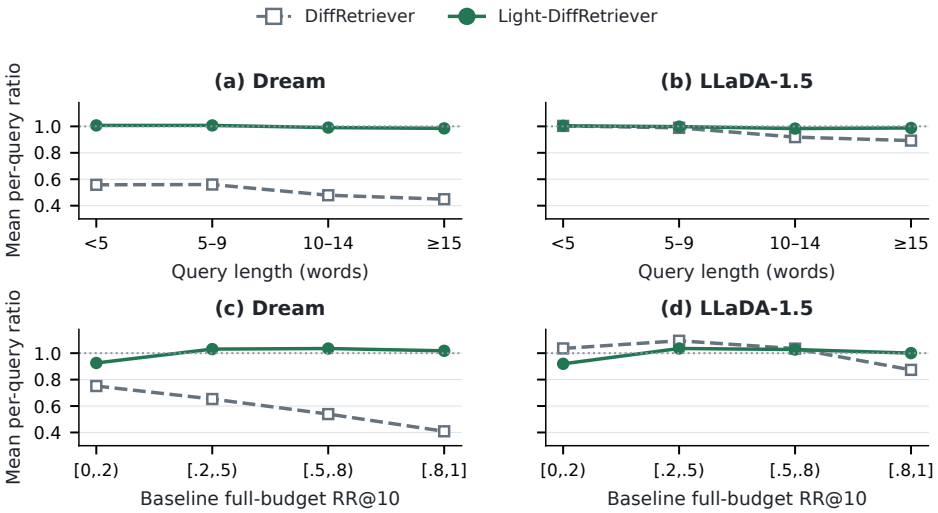


Figure 10: MS MARCO conditional per-query retention at $(4, 4)$ by query length and baseline full-budget difficulty. Dotted lines mark a ratio of one; eligible sets are method-specific.

E.2 Retention Across Datasets and Query Groups

Aggregate retention divides the summed reduced-budget scores by the summed full-budget scores. Conditional per-query retention instead averages individual ratios, excluding zero full-budget scores; the metric is RR@10 on MS MARCO and nDCG@10 elsewhere. This gives greater relative influence to small denominators and can exceed one even when aggregate effectiveness falls. Dream at $(4, 4)$ illustrates this distinction: conditional retention is 1.006, but aggregate MRR retention is 98.2%. Table 10 reports conditional ratios and eligible counts; Table 4 reports aggregate effectiveness.

Dream improves on NQ (.384 to .975) and HotpotQA (.413 to .943), whereas LLaDA-1.5 is not uniformly better: TREC-COVID changes from 1.064 to 1.052. Method-specific eligible subsets make these descriptive, not matched-cohort comparisons. Per-query means differ from the aggregate archive by less than .00024; the main performance tables retain their original aggregate scores.

Figure 10 shows stable retention across length groups: Dream baseline/ours span .449–.560/.985–1.007; LLaDA-1.5 spans .892–1.003/.983–1.004. The longest queries have the lowest baseline ratios, not the shortest. Difficulty groups use baseline full-budget scores but retain method-specific nonzero denominators. Dream's largest gain occurs in the easiest bucket (.409 to 1.018); LLaDA-1.5 does not improve in every bucket. These observations do not establish that the hardest queries benefit most.

Table 11: All five distinct queries in the supplied qualitative case list (Dream, MS MARCO). Passage IDs are ordered from rank 1 to 5; bold denotes qrel relevance. > 10 follows from zero RR@10, not merely absence from the displayed top five.

Method	Budget	Top-five passage IDs	First rel.
dga disc (qid 1101587)			
DiffRetriever	$(4, 16)$	**7751426**, 7751434, 2723187, 7751427, 3469468	1
DiffRetriever	$(4, 4)$	7751427, 3469468, 5426519, 2736335, 2723187	> 10
LIGHT-DIFFRETRIEVER	$(4, 16)$	**7751426**, 7751434, 2723187, 3469468, 7751435	1
LIGHT-DIFFRETRIEVER	$(4, 4)$	**7751426**, 7751434, 2723187, 7751435, 3469468	1
xenon gas (qid 1080909)			
DiffRetriever	$(4, 16)$	**7734933**, 205519, 7734934, 4906181, 2989222	1
DiffRetriever	$(4, 4)$	6540295, 6540294, 4906185, 7671265, 205527	> 10
LIGHT-DIFFRETRIEVER	$(4, 16)$	**7734933**, 7734934, 3853448, 205519, 7734935	1
LIGHT-DIFFRETRIEVER	$(4, 4)$	**7734933**, 7734934, 3853448, 205519, 7734935	1
who is pj (qid 989596)			
DiffRetriever	$(4, 16)$	7285835, **7285839**, 3257356, 5330093, 3257355	2
DiffRetriever	$(4, 4)$	7285835, 3297041, 7285840, 8821307, 8480021	> 10
LIGHT-DIFFRETRIEVER	$(4, 16)$	7285835, **7285839**, 5330093, 5717498, 8821307	2
LIGHT-DIFFRETRIEVER	$(4, 4)$	**7285839**, 8821307, 7285835, 5330093, 5717498	1
why sqlite (qid 1074184)			
DiffRetriever	$(4, 16)$	**7886803**, 4021775, 7886804, 4021768, 4548316	1
DiffRetriever	$(4, 4)$	4548313, 6290825, 7886806, 4021774, 1656859	> 10
LIGHT-DIFFRETRIEVER	$(4, 16)$	**7886803**, 4548316, 4021770, 4021768, 4548311	1
LIGHT-DIFFRETRIEVER	$(4, 4)$	**7886803**, 4548316, 4021768, 4021770, 7886804	1
what tw (qid 909658)			
DiffRetriever	$(4, 16)$	**7778491**, 3167752, 654430, 1202031, 437805	1
DiffRetriever	$(4, 4)$	1050188, 1202032, 437805, 654430, 1252845	> 10
LIGHT-DIFFRETRIEVER	$(4, 16)$	3167752, **7778491**, 654430, 4855026, 7778488	2
LIGHT-DIFFRETRIEVER	$(4, 4)$	3167752, **7778491**, **7778487**, 1252849, 1202031	2

E.3 COMPLETE QUALITATIVE RANKING RECORDS

The supplied qualitative report contains eight entries but only five distinct queries, all from Dream on MS MARCO; Table 11 removes repeated entries without dropping a distinct case. The source selection favors queries whose baseline ranking deteriorates under compression, so these are illustrative successes rather than a representative sample. For "xenon gas" and "why sqlite," our $(4, 4)$ prefix preserves rank one of the judged-relevant passage, as in the "dga disc" example. For "what tw," our relevant passage remains second at both budgets while the baseline loses its full-budget first-ranked relevant result from the top ten. "Who is pj" additionally illustrates a within-model improvement from rank two to one.

The cases are analyzed at the ranking level using query strings, document IDs, retrieval scores, and qrel markers. This comparison tracks the positions of judged-relevant passages across retained budgets, without assigning document topics or semantic roles from identifiers. Unmarked documents are not labeled relevant by the supplied qrels; this is not equivalent to an exhaustive irrelevance judgment. The relevant-rank comparisons therefore illustrate resilience to budget reduction under the benchmark judgments, rather than a semantic explanation of individual retrieval errors.

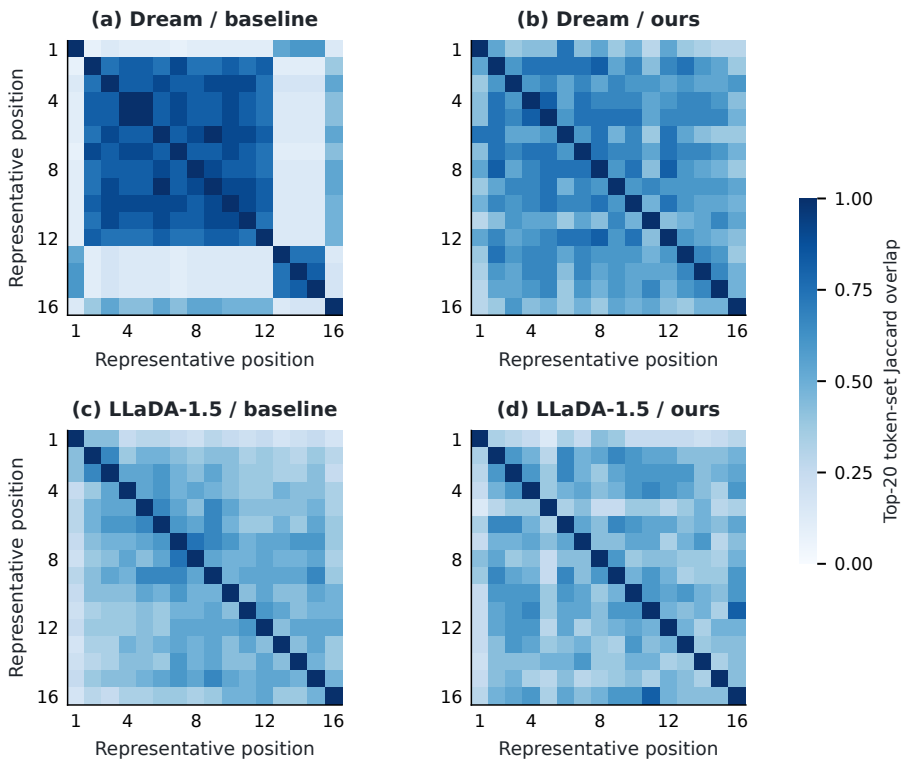


Figure 11: Overlap of the twenty most frequent decoded sparse tokens at each representative position. All panels use the same Jaccard scale; token overlap is not dense-vector similarity.

E.4 Positional-token Diagnostic

To examine whether representative positions exhibit an interpretable division of vocabulary, the diagnostic encodes 200 passages selected at intervals of 500 from the sampled passage bank. For each position and passage, it considers the top fifteen positive, content-filtered sparse activations, then retains the twenty decoded tokens with the highest occurrence counts across passages (activation sum breaks ties). Figure 11 shows pairwise Jaccard overlap between these position-specific token sets. Positions are displayed one-based, consistently with the methodology; no dense-vector similarity or retrieval score enters this visualization.

Mean off-diagonal overlap changes from .514 to .576 on Dream and from .448 to .450 on LLaDA-1.5. Frequently observed items include short subword fragments, and the probe does not establish a systematic progression from global concepts to fine-grained details. Nor does overlap of frequent sparse tokens establish redundancy of contextual dense representatives. Thus, we do not interpret the learnable offsets as guaranteed semantic roles or attribute their effect causally to semantic specialization. The supported mechanism remains the learning objective: retained prefixes receive retrieval supervision, while the offsets provide distinguishable, trainable input slots.

F LIMITATIONS AND OUTLOOK

From representation budgets to serving budgets. LIGHT-DIFFRETRIEVER makes the retained representative bank flexible, but does not reduce the backbone computation used to produce that bank. Query encoding consequently becomes a larger fraction of latency as matching costs decrease (Appendix D.3). Extending this flexibility to the entire retrieval pipeline calls for jointly designed adaptive encoders, compressed vector indexes, and serving policies. In particular, reducing the number of input masks is not equivalent to slicing a fully contextualized output bank. Future work could jointly train encoding and matching budgets with compatible corpus representations, accounting for approximate search, index updates, and service-level constraints.

Adaptive allocation across queries and retrieval modes. The present interface exposes a budget selected at deployment; it does not learn which budget an individual query needs. Aggregate retention also does not imply equally small changes for every query. A budget-selection policy could allocate additional representatives to queries with uncertain rankings, but would need calibrated quality estimates and an accounting of the policy's own overhead. Moreover, prefix supervision currently applies to dense banks, whereas sparse and hybrid retrieval retain full-capacity representations. Jointly nesting dense and lexical evidence, and calibrating their fusion as budgets change, could extend budget flexibility to sparse and hybrid retrieval.

Geometry-aware learning and broader representations. Our theoretical analysis gives conditional score and ranking guarantees for fixed encoded banks; it does not establish that optimization necessarily produces the required prefix coverage or ranking margins. An important direction is to connect these quantities to practical training objectives, including margin-aware or coverage-aware prefix supervision, without imposing prohibitive corpus-level computation. Beyond the English text setting studied here, multilingual and visually rich documents introduce additional allocation choices across languages, text segments, and image patches. Extending nested representatives to such inputs will require jointly modeling these heterogeneous information units, rather than assuming that the present ordering or budget schedule transfers unchanged. These directions extend the shared-bank interface toward one retriever that adapts to heterogeneous deployment requirements.

G AI USE STATEMENT

Following the ICLR 2027 AI Policy for Authors,[1] we disclose the use of GPT-6-Astra for language refinement and manuscript preparation, including literature and code analysis, mathematical formulation and proof drafting, interpretation of supplied results, and visualization-script preparation. AI assistance also informed the efficiency-measurement plan. The authors developed the core retrieval method and implementation and conducted the reported model training and evaluation; AI-assisted analysis operated on the supplied code and results rather than replacing the experiments or supplying experimental observations. Research decisions and responsibility for the accuracy, originality, and integrity of the final text, claims, and artifacts remain with the authors.

H ETHICS STATEMENT

This study uses existing retrieval benchmarks and pretrained models; it does not recruit human participants or collect new personal data. The planned code archive excludes third-party corpora and pretrained weights, which should be obtained from their original providers under the applicable licenses and access conditions. Public availability does not imply that every underlying query or document is free of sensitive information or harmful content. More efficient retrieval can also make such content easier to surface, so deployment should retain corpus-level access controls, privacy safeguards, and application-specific review. Benchmark relevance judgments and pretrained representations may encode social or linguistic biases, and aggregate accuracy retention is not evidence of equal treatment across user groups or topics. Budget allocation therefore warrants subgroup-aware assessment before consequential use. The experiments do not evaluate surveillance, demographic inference, or automated high-stakes decisions, and do not establish suitability for these

[1] https://iclr.cc/Conferences/2027/AIPolicyForAuthors

applications. For research integrity, we distinguish measured latency from analytical storage savings, separate full-corpus quality from sampled-corpus timing, and avoid assigning passage semantics from document identifiers alone. Any broader social or environmental benefit depends on the deployment context, not solely on reduced vector counts.

I Reproducibility Statement

An anonymized code archive will accompany the submission. Section 4, Algorithm 1, and Appendix C specify the method, training procedure, and theoretical assumptions and proofs. The implementation in `src/models/trainable_diff_retriever.py` contains the slot offsets, decaying initialization, and prefix losses; `scripts/train_retriever.py` supplies optimizer progress to the curriculum, and `scripts/run_train.sh` launches training. Table 3 specifies the main configuration, including $(K_q, K_p) = (4, 16)$ for *both* backbones, descending initialization with endpoint .1, prefix weight .5, and warm-up fraction .3; these settings must be selected explicitly rather than inferred from generic launcher defaults. Data preparation uses `scripts/prepare_msmarco.py`, `scripts/preprocess_msmarco_aug.py`, and `scripts/download_data.sh`; dependencies and prompts are recorded in `requirements.txt` and `prompts/default/`. For the experiments in Section 5, `scripts/encode_promptreps.py` encodes saved checkpoints, `scripts/evaluate_sweep.py` implements scoring and metrics, and `scripts/run_eval_all.sh` orchestrates datasets and retained budgets through `MRL_PAIRS`. The `results/` directory provides full-capacity `eval/summary.json` and prefix `eval_mrl/summary.json` records for numerical cross-checks; Appendix B.1 explains their scoring conventions. Data and trained checkpoints must be obtained or regenerated separately. The independent `efficiency_test/` timing and case-study harness is excluded from this archive; its protocols and diagnostics are documented in Appendices D.2–D.4 and E, rather than represented as included executable artifacts.