PIXEL AS THE COMMON LANGUAGE: UNIFIED MULTIMODAL EMBEDDING VIA VISUAL RENDERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal embedding models map heterogeneous inputs into a shared representation space, yet text-only, image-only, and composed inputs are still presented in distinct formats and follow different routes through the model. This fragmentation prevents heterogeneous tasks from jointly optimizing the same end-to-end encoding process. To address this issue, we propose *unification before encoding* and instantiate this paradigm in **RenderEmbed**. The framework uses *Visual Substrate Rendering* to convert heterogeneous inputs into a common visual substrate and then encodes the resulting visual-token sequence with a VLM. To retain both broader context and fine-grained cues in compact retrieval representations, it applies *Progressive Hierarchical Anchor Encoding* by interleaving anchors throughout the visual-token sequence and fusing their representations across layers. It further employs *Interventional Rendering Learning* to distinguish retrieval semantics from presentation variations through complementary content and rendering interventions. Extensive experiments on MMEB and ViDoRe v2 show that RenderEmbed outperforms prior state-of-the-art models at the same scale and exhibits strong out-of-domain generalization. Compared with MetaEmbed, RenderEmbed improves MMEB Overall Precision@1 by 1.7 points, with a larger 2.1-point gain on out-of-domain tasks.

1 INTRODUCTION

Multimodal embedding models map heterogeneous inputs containing text, images, or combinations of both into a unified representation space, providing a common similarity interface for diverse tasks such as cross-modal search, visual question answering, and multimodal retrieval-augmented generation (Radford et al., 2021; Jiang et al., 2024; Yu et al., 2025). Existing embedding methods achieve this unification through two main paradigms. Early approaches use modality-specific encoders and align their outputs in a shared embedding space for cross-modal comparison (Radford et al., 2021; Jia et al., 2021; Li et al., 2022). Recent universal embedding models advance unification beyond aligned outputs by using a vision-language model (VLM) to support diverse modality configurations (Jiang et al., 2024; Lin et al., 2025; Li et al., 2026). Despite this progression from output-space unification to encoder unification, text-only, image-only, and composed inputs retain distinct formats and follow different pathways through the model, as illustrated in Figure 1(a) and (b), preventing heterogeneous tasks from jointly optimizing the same end-to-end encoding process.

To establish such a process, we explore *unification before encoding* by expressing heterogeneous inputs in a common format before they enter the model. Recent work suggests that visual form can serve this role. Vision-based document retrievers encode pages directly as images, preserving textual, visual, and layout elements within a single visual input (Ma et al., 2024; Faysse et al., 2025). In parallel, several studies render text into images for context compression (Wei et al., 2025; Cheng et al., 2026) and visual latent reasoning (Wang et al., 2026; Jiang et al., 2026), demonstrating that linguistic content can also be processed in visual form. Together, they suggest that different modalities can share a common visual substrate. This leads to our central question: *can unified multimodal embedding be realized by adopting visual substrate as a common input format before encoding?*

To answer this question, we introduce RenderEmbed, a unified multimodal embedding framework built on the paradigm of *unification before encoding*. As illustrated in Figure 1(c), RenderEmbed renders each input into a common visual substrate and encodes the resulting visual-token sequence

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.

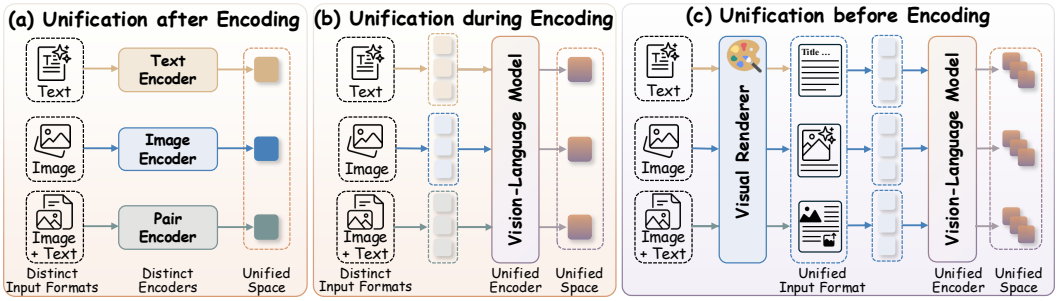


Figure 1: Progression of unification in multimodal embedding models. (a) Unification after encoding retains distinct input formats and encoders, aligning their outputs in a shared embedding space. (b) Unification during encoding introduces a shared VLM while retaining distinct input formats. (c) Unification before encoding (Ours), further unifies the input format by converting all inputs into a common visual substrate. The stacked outputs in (c) denote compact multi-vector embeddings.

into retrieval representations with a VLM. By standardizing the input format, it enables training examples across modality configurations to jointly optimize the same visual encoding pathway.

Implementing *unification before encoding* requires a coherent path from visual construction to retrieval encoding and representation learning. Figure 2 illustrates how RenderEmbed builds this path. RenderEmbed constructs a common visual substrate from heterogeneous inputs through *Visual Substrate Rendering*, using a canonical configuration selected by retrieval-guided policy search. After rendering, the framework faces two challenges. First, it must encode retrieval evidence distributed across textual, visual, and layout structures into compact representations. It therefore introduces *Progressive Hierarchical Anchor Encoding*, which uses progressively interleaved anchors to aggregate increasingly broad contexts and cross-layer fusion to combine multi-level representations. Second, it must learn robust retrieval representations that remain sensitive to content changes while remaining robust across different renderings. RenderEmbed promotes this robustness through *Interventional Rendering Learning*, which combines complementary semantic and rendering interventions.

To evaluate the effectiveness of RenderEmbed, we conduct comprehensive experiments on the MMEB (Jiang et al., 2024) and ViDoRe v2 (Macé et al., 2025) benchmarks, spanning retrieval tasks over text, images, and visually rich documents. The results show that RenderEmbed outperforms prior state-of-the-art models at the same scale, with particularly strong generalization to out-of-domain tasks. Moreover, its advantage over the native-input baseline grows as training incorporates more data, indicating favorable scaling with heterogeneous supervision.

Our contributions are threefold: (1) we propose *unification before encoding* paradigm for unified multimodal embedding, enabling diverse modality configurations to optimize the same end-to-end visual encoding pathway; (2) we develop RenderEmbed, combining progressive hierarchical anchor encoding and interventional rendering learning to produce compact and robust retrieval representations from a common visual substrate; and (3) RenderEmbed achieves state-of-the-art performance, strong out-of-domain generalization, and increasing gains from heterogeneous training supervision.

2 Related Work

Universal Multimodal Embedding. Multimodal embedding methods differ in where they unify heterogeneous inputs along the encoding pipeline. Classical models use modality-specific encoders and align their outputs in a shared embedding space after encoding (Radford et al., 2021; Jia et al., 2021; Li et al., 2022; Zhou et al., 2024). More recent universal retrievers, such as VLM2Vec (Jiang et al., 2024), GME (Zhang et al., 2024b), MM-EMBED (Lin et al., 2025), and MoCa (Chen et al., 2026), adapt vision-language models (Liu et al., 2023; Wang et al., 2024; Abdin et al., 2024; Yuan et al., 2025) to encode these input types within a shared backbone and training objective. Despite this architectural unification, their encoder inputs remain modality dependent, taking the form of text tokens, visual tokens, or interleaved multimodal sequences. RenderEmbed moves unification to the input stage by converting every retrieval unit into a common visual substrate before encoding.

Visualized Information Retrieval. Prior work connects visual representations with retrieval along two lines. CLIPPO (Tschannen et al., 2023) and Pixel Linguist II (Yuan et al., 2026a) render language

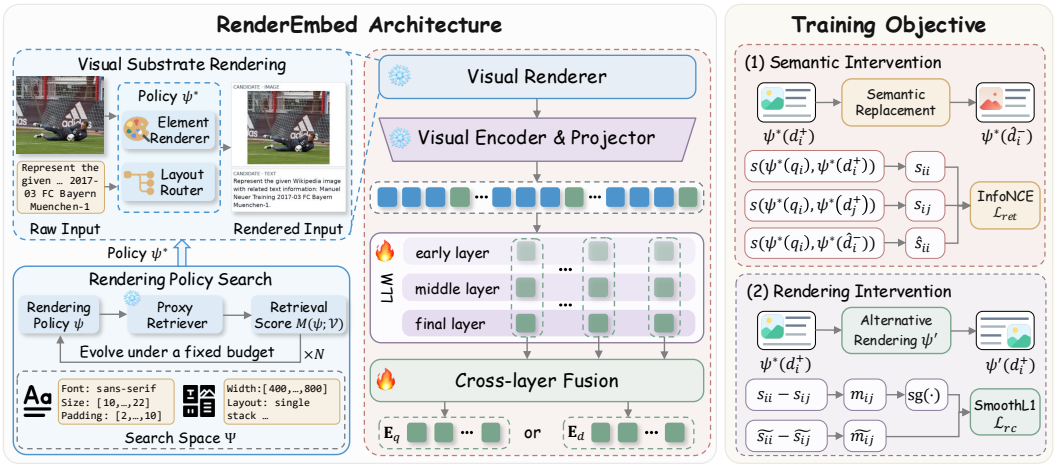


Figure 2: Overview of RenderEmbed. **Left**: Visual Substrate Rendering constructs a common visual substrate with a policy selected by retrieval-guided policy search. **Middle**: Progressive Hierarchical Anchor Encoding interleaves anchors with visual tokens and fuses their cross-layer states to produce compact retrieval vectors. **Right**: Interventional Rendering Learning combines semantic and rendering interventions to learn content-sensitive and rendering-robust retrieval representations.

as pixels for vision-based encoding, while VisRet (Wu et al., 2026) synthesizes images from textual queries before image retrieval. Screenshot-based methods instead treat page images as retrieval objects: DSE (Ma et al., 2024) represents pages without content extraction, ColPali (Faysse et al., 2025) preserves patch-level evidence through late interaction, and UniSE (Liu et al., 2025) extends screenshot retrieval across multiple data formats. Together, these methods establish visual encoding for rendered language and document screenshots in specific retrieval settings.

Multi-Vector Retrieval. Multi-vector retrievers based on the late-interaction schema preserve fine-grained token- or patch-level evidence but increase index size and scoring cost (Khattab & Zaharia, 2020; Santhanam et al., 2022; Faysse et al., 2025). Retrieval-oriented compression reduces this overhead through token pooling, sequence compression, and patch pruning or merging (Clavié et al., 2024; Ma et al., 2025; Yan et al., 2026; Yuan et al., 2026b). More recent approaches learn a small set of tokens to summarize inputs into compact multi-vector representations, as exemplified by AGC (Qin et al., 2026) and MetaEmbed (Xiao et al., 2026). Within this paradigm, RenderEmbed uses Progressive Hierarchical Anchor Encoding to aggregate retrieval-relevant evidence from rendered inputs via progressive anchor interleaving and cross-layer fusion.

3 Methodology

To realize *unification before encoding* for unified multimodal embedding, we propose RenderEmbed, which transforms modality-dependent input formats into a shared visual substrate. This design enables training examples with diverse modality configurations to supervise a single end-to-end visual encoding pathway. As illustrated in Figure 2, RenderEmbed comprises three components: *Visual Substrate Rendering* constructs unified visual inputs, *Progressive Hierarchical Anchor Encoding* produces compact multi-vector representations, and *Interventional Rendering Learning* learns robust representations for retrieval. We describe each component in the following sections.

3.1 Problem Formulation

We consider a general multimodal retrieval setting, where a query q is matched against a candidate collection $\mathcal{C}$. Both queries and candidates may contain text, images, or ordered compositions of both. Let $s(q, d)$ denote the relevance score between q and a candidate $d \in \mathcal{C}$. Thus, the top-1 candidate selected by the retrieval model is determined by the highest relevance score: $d^\star = \arg\max_{d \in \mathcal{C}} s(q, d)$. For subsequent operations shared by the query and candidate sides, we use x as a generic retrieval unit to represent either query or candidate input.

3.2 VISUAL SUBSTRATE RENDERING

To establish a common input format before encoding, we introduce *Visual Substrate Rendering*, which maps each retrieval unit x to a common visual substrate $\psi(x)$ under a rendering policy ψ. The policy specifies the visual style, layout geometry, and element-routing rules. The rendering process consists of two stages. First, an *Element Renderer* rasterizes text using the typography and style specified by ψ and proportionally fits images into the designated visual regions. An *Adaptive Layout Router* then assembles the rendered elements on a single pixel canvas while preserving their original order. In this way, these operations express retrieval units with different modality configurations in the same visual input format while preserving their content structure.

The common visual substrate is not unique: the same retrieval unit may admit multiple valid renderings that organize retrieval-relevant evidence differently and incur different visual-token costs. Since these choices affect both retrieval quality and encoding efficiency, we select a canonical policy ψ^* from the candidate space Ψ using validation data $\mathcal{V}$, subject to an average visual-token budget $\bar{\ell}$:

$$\psi^* = \arg\max_{\psi \in \Psi} \mathcal{M}(\psi; \mathcal{V}), \qquad \text{s.t.} \quad \mathbb{E}_{x \sim \mathcal{V}} \left[\ell\big(\psi(x)\big) \right] \leq \bar{\ell}. \tag{1}$$

Here, $\mathcal{M}$ measures retrieval performance, and $\ell(\cdot)$ counts visual tokens. We solve this discrete problem through evolutionary search (Cheng et al., 2026), using MoCa-3B (Chen et al., 2026) as a frozen proxy retriever, thereby decoupling rendering-policy selection from RenderEmbed optimization. Once selected, ψ^* is fixed as the canonical policy throughout training and inference processes. Further details are provided in Appendix A.

3.3 PROGRESSIVE HIERARCHICAL ANCHOR ENCODING

Rendered inputs distribute retrieval evidence across spatially separated text and image regions. Single-vector encoding can obscure local correspondences, whereas patch-level retrieval incurs high encoding, indexing, and scoring costs (Faysse et al., 2025). We therefore seek a compact multi-vector representation (Xiao et al., 2026) that maintains broad coverage under a fixed retrieval-vector budget. To this end, we propose *Progressive Hierarchical Anchor Encoding*, which uses a small set of learnable anchors to aggregate progressively expanding contexts across the rendered visual sequence and applies cross-layer fusion to combine their multi-level representations.

RenderEmbed uses a pretrained Vision-Language Model (VLM) to encode each rendered retrieval unit. Its vision encoder and projector map the rendered substrate $\psi(x)$ to a visual-token sequence:

$$\mathbf{E}_x = \text{Projector}\left(\text{VisionEncoder}\big(\psi(x)\big)\right) \in \mathbb{R}^{P_x \times H}, \tag{2}$$

where P_x is the number of visual tokens and H is the hidden dimension of the VLM.

To obtain role-specific retrieval representations, we introduce separate learnable anchor banks for the query and candidate sides:

$$\mathbf{A}_q = [\mathbf{a}_{q,1}, \ldots, \mathbf{a}_{q,M}] \in \mathbb{R}^{M \times H}, \qquad \mathbf{A}_d = [\mathbf{a}_{d,1}, \ldots, \mathbf{a}_{d,M}] \in \mathbb{R}^{M \times H}, \tag{3}$$

where M is the number of anchors. The two banks are learned independently, while all remaining encoding modules are shared. In what follows, $\mathbf{A}_x$ denotes the role-specific bank associated with x.

Appending all anchors after $\mathbf{E}_x$, as in MetaEmbed (Xiao et al., 2026), would give them the same complete visual prefix without explicitly differentiating their visual coverage. We instead partition $\mathbf{E}_x$ as evenly as possible into M consecutive segments, $\mathbf{E}_x = \mathbf{E}_{x,1} \| \cdots \| \mathbf{E}_{x,M}$, and interleave one anchor after each segment

$$\mathbf{Z}_x = \mathbf{E}_{x,1} \| \mathbf{a}_{x,1} \| \cdots \| \mathbf{E}_{x,M} \| \mathbf{a}_{x,M}, \tag{4}$$

where $\|$ denotes sequence concatenation. The language backbone of VLM jointly processes the visual tokens and anchors in $\mathbf{Z}_x$. Under causal attention, the i-th anchor can attend to the first i visual segments and all preceding anchors, so successive anchors aggregate nested and progressively expanding visual contexts.

Progressive interleaving expands each anchor's spatial scope, but its final-layer state reflects only one representational depth. Because fine-grained visual cues and contextual semantics are emphasized at different depths of the VLM, we fuse each anchor's hidden states across multiple layers. Concretely, let $\mathbf{h}_{x,i}^{(l)}$ denote the hidden state of anchor $\mathbf{a}_{x,i}$ at layer l. We collect its states from a set of K

selected layers, $\mathcal{I}_{\text{fuse}} = \{l_1, \ldots, l_K\}$, which always includes the final layer l_K. After applying RMSNorm (Zhang & Sennrich, 2019) to each state, we concatenate them and use a two-layer MLP with GELU activation to project the resulting KH-dimensional representation back to the H-dimensional retrieval space:

$$\mathbf{f}_i\big(\psi(x)\big) = \mathrm{MLP}\left(\left[\mathrm{RMSNorm}\left(\mathbf{h}_{x,i}^{(l_1)}\right); \ldots; \mathrm{RMSNorm}\left(\mathbf{h}_{x,i}^{(l_K)}\right)\right]\right). \tag{5}$$

We then apply ℓ_2 normalization to obtain the retrieval vector $\mathbf{e}_i(\psi(x)) = \mathbf{f}_i(\psi(x))/\|\mathbf{f}_i(\psi(x))\|_2$.

Under the canonical rendering policy ψ^*, the VLM encodes $\psi^*(q)$ and $\psi^*(d)$ in separate forward passes, producing M retrieval vectors for each. Their relevance score is computed using the MaxSim operation in late interaction (Khattab & Zaharia, 2020):

$$s\big(\psi^*(q), \psi^*(d)\big) = \sum_{i=1}^{M} \max_{1 \le j \le M} \mathbf{e}_i\big(\psi^*(q)\big)^\top \mathbf{e}_j\big(\psi^*(d)\big). \tag{6}$$

3.4 Interventional Rendering Learning

Rendering introduces presentation variations in addition to semantic content. A retrieval model should respond to changes in retrieval-relevant content while remaining robust when the same content is presented under different renderings. We therefore propose *Interventional Rendering Learning*, which targets these complementary behaviors through a rendering-controlled semantic intervention and a semantics-preserving rendering intervention.

Rendering-Controlled Semantic Intervention. For a training batch $\mathcal{B} = \{(q_i, d_i^+, d_i^-)\}_{i=1}^B$, each query q_i is paired with a labeled positive d_i^+ and negative d_i^-. After visual rendering, the negative d_i^- may differ from d_i^+ in both retrieval-relevant content and visual organization, entangling semantic and rendering changes. To isolate the semantic difference, we construct a rendering-controlled negative $\hat{d}_i^-$ by replacing one retrieval-relevant element in d_i^+ with a same-type element from d_i^- that contains different content. This minimal, single-element intervention preserves the original layout geometry and all unchanged regions, keeping $\hat{d}_i^-$ close to d_i^+ and focusing learning on fine-grained semantic differences. Under the canonical policy ψ^*, let $s_{ij} = s\big(\psi^*(q_i), \psi^*(d_j^+)\big)$ denote the score between query q_i and candidate d_j^+, and let $\hat{s}_{ii} = s\big(\psi^*(q_i), \psi^*(\hat{d}_i^-)\big)$ denote the score of the rendering-controlled negative. Standard InfoNCE contrasts the positive score s_{ii} against $\{s_{ij}\}_{j \neq i}$, where the positives of other queries serve as in-batch negatives (Jiang et al., 2024). To encourage the model to distinguish retrieval-relevant content changes under matched rendering conditions, we add the controlled-negative score $\hat{s}_{ii}$ to the same denominator:

$$\mathcal{L}_{\text{ret}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(s_{ii}/\tau)}{\sum_{j=1}^{B} \exp(s_{ij}/\tau) + \exp(\hat{s}_{ii}/\tau)}, \tag{7}$$

where τ denotes the temperature.

Semantics-Preserving Rendering Intervention. The semantic intervention encourages sensitivity to changes in retrieval-relevant content, while the complementary objective is to preserve retrieval decisions across alternative renderings of the same content. We sample a policy $\psi' \neq \psi^*$ and use it to render each positive candidate d_j^+, varying its presentation while preserving its content and element order. Pairing the alternative candidate rendering $\psi'(d_j^+)$ with the canonical query rendering $\psi^*(q_i)$ yields the alternative-rendering score $\widetilde{s}_{ij} = s\big(\psi^*(q_i), \psi'(d_j^+)\big)$.

Alternative renderings can produce different visual-token sequences and retrieval vectors for the same content, making direct embedding alignment overly restrictive. Thus, we align the relative retrieval decisions induced by the canonical and alternative renderings. For $i \neq j$, their positive-to-negative margins are defined as $m_{ij} = s_{ii} - s_{ij}$ and $\widetilde{m}_{ij} = \widetilde{s}_{ii} - \widetilde{s}_{ij}$. Their consistency is then optimized by

$$\mathcal{L}_{\text{rc}} = \frac{1}{B(B-1)} \sum_{i=1}^{B} \sum_{j \neq i} \mathrm{SmoothL1}\left(\widetilde{m}_{ij} - \mathrm{sg}\left(m_{ij}\right)\right), \tag{8}$$

Table 1: Main results on MMEB. Precision@1 is reported for each model. IND and OOD denote in-domain and out-of-domain averages.

Model	Size	Per Meta-Task Score				Average Score		
		Classification	VQA	Retrieval	Grounding	IND	OOD	Overall
CLIP (Radford et al., 2021)	428M	42.8	9.1	53.0	51.8	37.1	38.7	37.8
BLIP2 (Li et al., 2023)	428M	27.0	4.2	33.9	47.0	25.3	25.1	25.2
OpenCLIP (Cherti et al., 2023)	428M	47.8	10.9	52.3	53.3	39.3	40.2	39.7
SigLIP (Zhai et al., 2023)	652M	40.3	8.4	31.6	59.5	32.3	38.0	34.8
Magiclens (Zhang et al., 2024a)	613M	38.8	8.3	35.4	26.0	31.0	23.7	27.8
UniIR (Wei et al., 2024)	428M	42.1	15.0	60.1	62.2	44.7	40.4	42.8
GME (Zhang et al., 2024b)	7B	58.6	35.8	73.1	67.3	55.8	60.9	58.0
VLM2Vec (Jiang et al., 2024)	7B	61.2	49.9	67.4	86.1	67.5	57.1	62.9
VLM2Vec-V2 (Meng et al., 2026)	2B	62.9	56.3	69.5	77.3	–	–	64.9
MM-Embed (Lin et al., 2025)	7B	48.1	32.2	63.8	57.8	–	–	50.0
MMRet (Zhou et al., 2025)	7B	56.0	57.4	69.9	83.6	68.0	59.1	64.1
UniME-V2-2B (Gu et al., 2026)	2B	49.5	52.0	58.0	65.9	59.2	49.5	54.9
UniME-V2-7B (Gu et al., 2026)	7B	51.9	56.4	62.8	74.3	62.3	55.5	59.3
MoCa-3B (Chen et al., 2026)	3B	59.8	62.9	70.6	**88.6**	72.3	61.5	67.5
MetaEmbed (Xiao et al., 2026)	3B	62.7	68.1	71.9	79.6	73.5	63.8	69.1
RenderEmbed (Ours)	3B	**64.3**	**68.9**	**73.6**	83.2	**74.7**	**65.9**	**70.8**

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. The canonical margin serves as the reference, allowing the retrieval vectors to vary across rendering settings while preserving the relative retrieval structure. At each training step, ψ' is resampled from a pool of alternative policies, as described in Appendix B.2.

Training Objective. The final objective combines the intervention-augmented contrastive loss with rendering-consistency regularization: $\mathcal{L} = \mathcal{L}_{\mathrm{ret}} + \lambda_{\mathrm{rc}}\mathcal{L}_{\mathrm{rc}}$, where λ_{rc} balances the two objectives. Both interventions are confined to training, while indexing and inference use the canonical policy ψ^* without controlled negatives or alternative rendering views.

4 Experiments

4.1 Experimental Setup

Training Data and Benchmarks. Following the experimental setting of MetaEmbed (Xiao et al., 2026), we train RenderEmbed on MoCa-CL-Pairs (Chen et al., 2026), which combines the training set of MMEB (Jiang et al., 2024) and ViDoRe (Faysse et al., 2025). MMEB contains 20 datasets spanning four meta-tasks: classification, visual question answering (VQA), multimodal retrieval, and visual grounding (Jiang et al., 2024). ViDoRe focuses on visually rich document retrieval, covering documents from public academic datasets and web-crawled PDFs. We evaluate general multimodal embedding on MMEB using Precision@1 and visual document retrieval on ViDoRe v2 (Faysse et al., 2025) using average NDCG@5. The MMEB comparison includes multimodal embedding models, whereas ViDoRe v2 additionally includes multi-vector models designed for visual document retrieval.

Implementation Details. RenderEmbed is initialized from Qwen2.5-VL-3B (Bai et al., 2025). We train the role-specific anchor banks and the cross-layer fusion module. Every LLM layer is adapted with LoRA (Hu et al., 2022) using rank 32 and scaling factor $\alpha = 32$, while all remaining parameters are frozen. The final configuration uses $M = 16$ anchors and $\mathcal{I}_{\mathrm{fuse}} = \{18, 24, 30, 36\}$. Training runs for 1 epoch with a global batch size of 128, temperature $\tau = 0.02$, and rendering-consistency weight $\lambda_{\mathrm{rc}} = 0.1$, while each query-positive pair contributes one rendering-controlled semantic negative and one alternative rendering view. For RenderEmbed, all training and evaluation inputs pass through Visual Substrate Rendering before encoding. More experimental details are provided in Appendix B.

4.2 Main Results

Tables 1 and 2 summarize the main results on general multimodal embedding and visual document retrieval, respectively. RenderEmbed, MoCa-3B, ColQwen2.5, and MetaEmbed share the same Qwen2.5-VL-3B backbone, enabling capacity-matched comparisons across retrieval representation types. Specifically, MoCa-3B uses a single-vector representation, ColQwen2.5 retains patch-level

Table 2: Main results on ViDoRe v2. The metric is nDCG@5 (%). "EN" is the English subset, "Multi. Avg." denotes the average over multilingual queries, and "Human" denotes the human-labeled subset.

Method	Size	Biomedical Lectures		Economics Reports		ESG Reports			Avg.
		EN	Multi. Avg.	EN	Multi. Avg.	EN	Multi. Avg.	Human	
Single-Vector Retrieval									
SigLIP (Zhai et al., 2023)	652M	33.8	18.2	29.8	14.0	19.8	21.9	28.8	23.8
VLM2Vec (Jiang et al., 2024)	7B	38.8	29.7	51.4	42.0	36.7	38.4	33.9	38.7
VisRAG-Ret (Yu et al., 2025)	3B	54.8	47.7	59.6	48.7	45.9	46.4	53.7	51.0
GME (Zhang et al., 2024b)	7B	64.0	55.1	**62.9**	56.2	54.3	56.7	**65.8**	59.3
MoCa-3B (Chen et al., 2026)	3B	62.5	59.8	62.8	**57.3**	58.3	54.8	63.3	59.8
Multi-Vector Retrieval									
ColPali (Faysse et al., 2025)	3B	59.7	56.5	51.6	49.9	57.0	55.7	51.1	54.5
ColQwen2.5 (Faysse et al., 2025)	3B	63.2	60.3	61.0	57.2	60.6	56.2	64.6	60.4
Jina-v4 (Günther et al., 2025)	4B	65.1	62.0	57.6	54.5	50.9	52.7	60.9	57.7
AGC (Qin et al., 2026)	3B	60.2	57.7	62.8	56.3	54.5	53.1	54.5	57.0
MAGIC (Yuan et al., 2026b)	3B	60.6	57.8	60.3	56.8	57.2	53.2	55.4	57.3
MetaEmbed (Xiao et al., 2026)	3B	61.7	58.7	62.3	55.5	**62.6**	57.4	63.7	60.3
RenderEmbed (Ours)	3B	**66.2**	**62.1**	59.9	55.6	60.6	**58.2**	63.6	**60.9**

vectors, and MetaEmbed learns a compact multi-vector representation. Our RenderEmbed instead constructs compact multi-vector embeddings from a common visual substrate, using $M = 16$ vectors on both the query and candidate sides.

General Multimodal Embedding. Across MMEB, RenderEmbed achieves the strongest overall performance, leading three of the four meta-task categories as well as both the in-domain and out-of-domain averages. It outperforms the state-of-the-art 3B baselines MetaEmbed and MoCa-3B under the same backbone, while also surpassing every 7B model listed in Table 1. The advantage over both 3B baselines is larger on out-of-domain tasks than on in-domain tasks, reaching 2.1 points over MetaEmbed and 4.4 points over MoCa-3B. This pattern is consistent with the *unification before encoding* paradigm of RenderEmbed: rendering queries and candidates into a common visual substrate allows heterogeneous tasks to supervise the same end-to-end visual encoding pathway rather than distinct modality-conditioned pathways in previous *unification after encoding* and *unification during encoding* approaches. Supervision from different task and modality configurations can therefore update the same representation pathway, supporting transfer to held-out tasks.

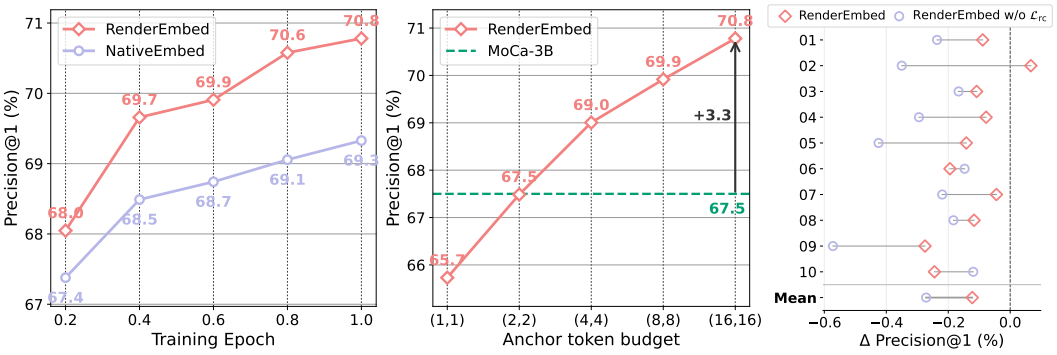
Visual Document Retrieval. The performance advantage also extends to visual document retrieval: on ViDoRe v2, RenderEmbed obtains the highest average NDCG@5 among the compared single-vector and multi-vector methods and outperforms all three capacity-matched baselines. Its strongest results appear on multilingual retrieval, where it ranks first on the multilingual Biomedical and ESG subsets despite the limited share of multilingual examples in the training corpus. By rendering multilingual inputs into the same visual substrate as the rest of the training corpus, RenderEmbed allows limited multilingual supervision to update a common visual encoding pathway, which may facilitate transfer across languages.

4.3 Ablation Studies

Core Design. Table 3 reports controlled ablations on MMEB. Due to computational constraints, all variants are trained on the same 25% subset of the training corpus, with shared backbone and optimization settings. NativeEmbed in Table 3 replaces Visual Substrate Rendering with native multimodal inputs. Since interventional rendering learning relies on rendered input format, it is not applicable to NativeEmbed, which therefore uses the standard InfoNCE objective. The *w/o Interventional Rendering Learning* variant retains the common visual substrate but removes both interventions and uses the same InfoNCE objective. Comparing these variants measures the benefit of introducing a common visual substrate under standard contrastive learning,

Table 3: Ablation study on the core designs of RenderEmbed. Δ is the change in MMEB Overall Precision@1 relative to our RenderEmbed.

Model	Overall ↑	Δ
RenderEmbed	68.6	–
NativeEmbed	67.5	−1.1
w Appended Anchors	68.2	−0.4
w Final Layer Only	67.9	−0.7
w/o Semantic Intervention	68.1	−0.5
w/o Rendering Intervention	68.4	−0.2
w/o Interventional Rendering Learning	68.1	−0.5



(a) Training-Data Scaling (b) Test-Time Anchor Budget (c) Rendering Sensitivity

Figure 3: Analysis on MMEB. (a) Overall Precision@1 of RenderEmbed and NativeEmbed with various training epochs, representing increasing training budgets. (b) Overall Precision@1 under different anchor budgets. The dashed line marks the MoCa-3B baseline. (c) Change in Overall Precision@1 for RenderEmbed and *w/o Rendering Intervention* variant under 10 different non-canonical rendering policies. Changes are measured relative to the canonical rendering policy, with positive and negative values indicating improvements and degradations, respectively. The vertical dashed line denotes no change, and the *Mean* row gives the average across policies.

while comparing the latter with RenderEmbed measures the additional contribution of interventional rendering learning. *w Appended Anchors* places all anchors after the input sequence instead of progressively interleaving them, whereas *w Final Layer Only* constructs retrieval vectors from final-layer anchor states without cross-layer fusion. The remaining variants individually remove the rendering-controlled semantic intervention or the rendering-consistency objective.

The controlled comparison from NativeEmbed through *w/o Interventional Rendering Learning* to RenderEmbed separates two sources of improvement: the common visual substrate contributes 0.6 points under the same InfoNCE objective, and interventional rendering learning adds another 0.5 points. Among the encoding designs, the 0.7-point drop without cross-layer fusion highlights the importance of aggregating anchor representations across different depths. Notably, removing only the semantic intervention produces the same 0.5-point drop as removing both interventions, identifying semantic discrimination as the main driver of accuracy under canonical rendering. The rendering intervention is designed to stabilize retrieval across presentation changes, which we examine directly under alternative rendering policies in Section 4.4.

Fusion Layers. We further analyze which backbone depths contribute most effectively to cross-layer fusion. For the 36-layer Qwen2.5-VL-3B backbone, we compare early fusion with $\mathcal{I}_{\text{fuse}} = \{6, 12, 18, 36\}$, middle fusion with $\{18, 24, 30, 36\}$, and late fusion with $\{33, 34, 35, 36\}$. All settings fuse four-layer states and include the final layer, while the no-fusion baseline uses only $\{36\}$. Figure 4 shows that all cross-layer configurations outperform using the final layer alone, with middle fusion performing best. This pattern suggests that effective fusion requires representations that are sufficiently deep to capture retrieval-relevant semantics while remaining separated enough to provide complementary information.

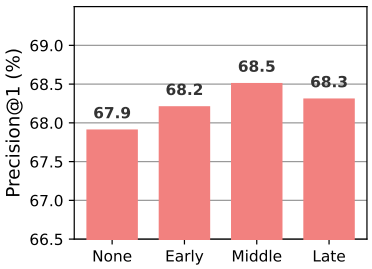


Figure 4: Overall Precision@1 under different fusion-layer settings.

4.4 Analysis

Training-Data Scaling. Figure 3(a) evaluates MMEB Overall Precision@1 after training for 0.2, 0.4, 0.6, 0.8, and 1.0 epochs. Both methods improve with larger training budgets, while RenderEmbed consistently outperforms NativeEmbed at every budget. The advantage also increases from 0.6 points at 0.2 epochs to 1.5 points at 1.0 epochs, indicating that RenderEmbed benefits increasingly from additional training data. We attribute this trend to the proposed *unification before encoding* paradigm,

Table 4: Representation storage and scoring efficiency of representative multi-vector retrievers under a controlled workload of 1,000 queries and 100,000 candidates. "Q/C Vectors" is the average number of query and candidate vectors. Per-query scoring latency is reported as $\mathrm{mean}_{\pm\mathrm{std}}$ with 10 runs.

Model	Q/C Vectors	Embedding Dim.	Index Size (GiB)	Scoring FLOPs (G)	Scoring Latency (ms)
ColPali	31 / 1031	128	24.58	826.17	$13.06_{\pm 7.0\times10^{-4}}$
ColQwen2.5	35 / 751	128	17.90	666.05	$11.58_{\pm 8.3\times10^{-4}}$
MetaEmbed-3B	16 / 64	2048	24.41	419.43	$4.18_{\pm 1.5\times10^{-4}}$
RenderEmbed	16 / 16	2048	**6.10**	**104.85**	$\mathbf{1.30_{\pm 1.0\times10^{-4}}}$

which allows the heterogeneous training corpus to jointly optimize the same end-to-end visual encoding pathway and thereby improves the utilization of additional training data.

Test-Time Anchor Budget. Figure 3(b) evaluates RenderEmbed under symmetric test-time anchor budgets, retaining the same number of query and candidate anchors. MetaEmbed learns prefix-truncatable representations through the Matryoshka Multi-Vector Retrieval objective (Xiao et al., 2026), whereas RenderEmbed supports test-time budget adaptation without an additional budget-specific training objective. For a target budget, we uniformly sample anchors across their progressively interleaved positions and always retain the final anchor. Because successive anchors aggregate progressively broader visual contexts, this strategy preserves multiple context scales while maintaining coverage of the complete input. Performance increases monotonically with the anchor budget: using two anchors per side already matches MoCa-3B while retaining only $1/8$ of the full-budget candidate vectors and requiring $1/64$ of the anchor-pair comparisons. With the full $(16, 16)$ budget, RenderEmbed exceeds MoCa-3B by 3.3 points. This monotonic curve provides a flexible quality–efficiency trade-off from a single trained model. Appendix C.1 further shows that, once the query retains several anchors, candidate-heavy allocations achieve higher accuracy at matched scoring cost.

Rendering Sensitivity. To test robustness to presentation changes, we compare RenderEmbed with the *w/o Rendering Intervention* under 10 non-canonical policies sampled from the policy candidate space, as shown in Figure 3(c). Under each policy, both queries and candidates are rendered accordingly, and we report the change in Overall Precision@1 relative to canonical rendering. Both variants remain relatively robust across these policies, with all performance changes bounded within 0.6 points, indicating that the underlying RenderEmbed framework is already tolerant to rendering variation. Beyond this baseline robustness, the full model exhibits a more favorable relative change under 8/10 policies, together with a smaller mean degradation and a less severe worst-case drop. By aligning positive-to-negative margins between canonical and alternative renderings, the rendering-consistency objective further stabilizes retrieval decisions across changes in layout and visual style.

Efficiency. We compare RenderEmbed with representative multi-vector retrievers in terms of representation storage and late-interaction scoring efficiency (Table 4). RenderEmbed performs best across all three efficiency metrics. MetaEmbed-3B provides the most direct comparison because it uses the same embedding dimension and query-vector budget as RenderEmbed. Compared with its 64 candidate vectors, the 16-vector candidate representation of RenderEmbed reduces index storage and scoring FLOPs by a factor of four, while lowering scoring latency from 4.18 to 1.30 ms, corresponding to a $3.2\times$ speedup. Full measurement details are provided in Appendix B.3.

5 Conclusion

RenderEmbed demonstrates that multimodal unification can be extended to the input level through *unification before encoding*. It uses *Visual Substrate Rendering* to convert text, images, and their combinations into a common visual substrate, enabling heterogeneous supervision to optimize a shared visual encoding pathway. The framework further integrates *Progressive Hierarchical Anchor Encoding* to retain distributed evidence in compact retrieval representations and *Interventional Rendering Learning* to distinguish retrieval semantics from presentation variations. Across MMEB and ViDoRe v2, RenderEmbed attains state-of-the-art performance at the same scale, with strong out-of-domain generalization and growing gains from heterogeneous supervision.

AI USE STATEMENT

In this work, we used AI tools to support the retrieval and discovery of related literature, assist with the implementation of auxiliary training and testing scripts, and aid in drafting and polishing the manuscript. Generative AI was not used to design or implement the core method and experiments. The authors reviewed all AI-assisted work and take responsibility for the final content of this work, including its text, claims, code, and other artifacts produced with the aid of generative AI.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, and Johan Bjorck. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Haonan Chen, Hong Liu, Yuping Luo, Liang Wang, Nan Yang, Furu Wei, and Zhicheng Dou. Moca: Modality-aware continual pre-training makes better bidirectional multimodal embeddings. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 810–823, 2026.

Jiale Cheng, Yusen Liu, Xinyu Zhang, Yulin Fei, Wenyi Hong, Ruiliang Lyu, Weihan Wang, Zhe Su, Xiaotao Gu, Xiao Liu, et al. Glyph: Scaling context windows via visual-text compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37145–37158, 2026.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2829. IEEE, 2023.

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

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

Tiancheng Gu, Kaicheng Yang, Kaichen Zhang, Xiang An, Ziyong Feng, Yueyi Zhang, Weidong Cai, Jiankang Deng, and Lidong Bing. Unime-v2: Mllm-as-a-judge for universal multimodal embedding learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 21378–21386, 2026.

Michael Günther, Saba Sturua, Mohammad Kalim Akram, Isabelle Mohr, Andrei Ungureanu, Bo Wang, Sedigheh Eslami, Scott Martens, Maximilian Werk, Nan Wang, et al. jina-embeddings-v4: Universal embeddings for multimodal multilingual retrieval. In *Proceedings of the 5th Workshop on Multilingual Representation Learning (MRL 2025)*, pp. 531–550, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pp. 4904–4916. PMLR, 2021.

Houcheng Jiang, Jiajun Fu, Junfeng Fang, Chen Gao, Xiang Wang, Xiangnan He, and Yong Li. Univlr: Unifying text and vision in visual latent reasoning for multimodal llms. *arXiv preprint arXiv:2605.11856*, 2026.

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

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.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Zheng Liu, Ze Liu, Zhengyang Liang, Junjie Zhou, Shitao Xiao, Chao Gao, Chen Jason Zhang, and Defu Lian. Any information is just worth one single screenshot: Unifying search with visualized information retrieval. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19238–19261, 2025.

Xueguang Ma, Sheng-Chieh Lin, Minghan Li, Wenhu Chen, and Jimmy Lin. Unifying multimodal retrieval via document screenshot embedding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6492–6505, 2024.

Yubo Ma, Jinsong Li, Yuhang Zang, Xiaobao Wu, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Jiaqi Wang, Yixin Cao, et al. Towards storage-efficient visual document retrieval: An empirical study on reducing patch-level embeddings. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19568–19580, 2025.

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

Rui Meng, Ziyan Jiang, Ye Liu, Mingyi Su, Xinyi Yang, Yuepeng Fu, Can Qin, Raghuveer Thirukovalluru, Xuan Zhang, Zeyuan Chen, Ran Xu, Caiming Xiong, Yingbo Zhou, Wenhu Chen, and Semih Yavuz. VLM2vec-v2: Advancing multimodal embedding for videos, images, and visual documents. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Hanxiang Qin, Alexander Martin, Rohan Jha, Chunsheng Zuo, Reno Kriz, and Benjamin Van Durme. Multi-vector index compression in any modality. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1519–1530, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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, 2022.

Michael Tschannen, Basil Mustafa, and Neil Houlsby. Clippo: Image-and-language understanding from pixels only. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11006–11017. IEEE, 2023.

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

Yifan Wang, Shiyu Li, Peiming Li, Xiaochen Yang, Zheng Wei, and Yang Tang. Render-of-thought: Rendering textual chain-of-thought as images for visual latent reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45236–45253, 2026.

Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. Uniir: Training and benchmarking universal multimodal information retrievers. In *European Conference on Computer Vision*, pp. 387–404. Springer, 2024.

Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025.

Di Wu, Yixin Wan, and Kai-Wei Chang. Visret: Visualization improves knowledge-intensive text-to-image retrieval. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25982–26001, 2026.

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

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

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

Chaohao Yuan, Ruifeng Yuan, Zhuoxu Huang, Yu Rong, Hong Cheng, Hou Pong Chan, and Chenghao Xiao. On the design fundamentals of pixel text representation learning. *arXiv preprint arXiv:2609.01147*, 2026a.

Xu Yuan, Li Zhou, Zenghui Sun, Zikun Zhou, and Jingsong Lan. Instruction-guided multi-granularity segmentation and captioning with large multimodal model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Xu Yuan, Hua Liu, Wenqi Fan, and Qing Li. Magic: Marginal-guided compression with optimal transport for efficient visual document retrieval, 2026b. URL `https://arxiv.org/abs/2609.21018`.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Kai Zhang, Yi Luan, Hexiang Hu, Kenton Lee, Siyuan Qiao, Wenhu Chen, Yu Su, and Ming-Wei Chang. Magiclens: Self-supervised image retrieval with open-ended instructions. In *Forty-first International Conference on Machine Learning*, 2024a.

Xin Zhang, Yanzhao Zhang, Wen Xie, Mingxin Li, Ziqi Dai, Dingkun Long, Pengjun Xie, Meishan Zhang, Wenjie Li, and Min Zhang. Gme: Improving universal multimodal retrieval by multimodal llms. *arXiv preprint arXiv:2412.16855*, 2024b.

Junjie Zhou, Zheng Liu, Shitao Xiao, Bo Zhao, and Yongping Xiong. Vista: Visualized text embedding for universal multi-modal retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3185–3200, 2024.

Junjie Zhou, Yongping Xiong, Zheng Liu, Ze Liu, Shitao Xiao, Yueze Wang, Bo Zhao, Chen Jason Zhang, and Defu Lian. Megapairs: Massive data synthesis for universal multimodal retrieval. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19076–19095, 2025.

A Rendering Implementation Details

This section provides the implementation and selection details of Visual Substrate Rendering. Section A.1 describes how heterogeneous inputs, including text, image, and composed elements, are rendered into visual form and routed among the supported layout templates. Section A.2 defines the rendering-policy space and presents the retrieval-guided policy search used to select a canonical rendering policy. Finally, Section A.3 analyzes the search outcomes and specifies the selected canonical rendering policy.

A.1 Element Rendering and Adaptive Layout Routing

Visual Substrate Rendering maps each retrieval unit x to a visual substrate $\psi(x)$ under a rendering policy ψ. A rendering policy is a parameterized specification of the complete rendering process: it determines how the text and image elements of x are visually instantiated and how the rendered elements are spatially composed into a common substrate. This process is implemented by two components. The Element Renderer converts individual elements into visual cards, while the Adaptive Layout Router selects among the supported layouts according to the input composition and rendered geometry.

Element Renderer. For a retrieval unit x, the Element Renderer retains its non-empty text and image elements in their original order and renders each element as a typed visual card. Each card contains a header that identifies its retrieval role (query or candidate) and element type (text or image), together with a content region, background, and border. The parameters in ψ determine the typography, color theme, and internal padding. Text is wrapped within the available content width, whereas an image is fitted proportionally to its assigned region. Given the assigned card width, the header, rendered content, and internal padding determine the final card height.

Adaptive Layout Router. Given the rendered element cards, the Adaptive Layout Router constructs the layout plans applicable to the input composition. Every composition admits an order-preserving vertical plan, denoted by π_{stack}. A single-text input additionally admits a two-column plan π_{text}, while an input containing one text element and one image element admits a side-by-side plan π_{mix}. Each composition-specific plan is considered feasible only when every column or lane retains sufficient content width after padding.

For a single-text input, π_{text} wraps the text at the equal-column width and divides the resulting lines into two contiguous columns. Let W_{stack} and H_{stack} denote the dimensions of π_{stack}. The router selects π_{text} when it is feasible and $H_{\text{stack}}/W_{\text{stack}} \geq \eta_{\text{text}}$, where η_{text} specifies the height-to-width ratio at which the two-column layout is activated. For an input containing one text element and one image element, π_{mix} places the two cards in equal-width lanes while preserving their input order. Let h_{T} and h_{I} denote the resulting card heights. The router selects π_{mix} when it is feasible, $\max(h_{\text{T}}, h_{\text{I}})/\min(h_{\text{T}}, h_{\text{I}}) \leq \eta_{\text{mix}}$, and $\ell(\pi_{\text{mix}}) < \ell(\pi_{\text{stack}})$. Here, η_{mix} limits the imbalance between the two card heights, and $\ell(\pi)$ denotes the visual-token count produced by rasterizing plan π. Inputs without an applicable composition-specific plan, or whose plan fails its routing criteria, use π_{stack}. Figures 5 and 6 show representative renderings produced under the selected canonical rendering policy ψ^*, specified in Section A.3, while Table 5 summarizes their applicable input compositions and routing criteria.

Algorithm 1 formalizes these routing rules and rasterizes the selected plan into the visual substrate $\psi(x)$.

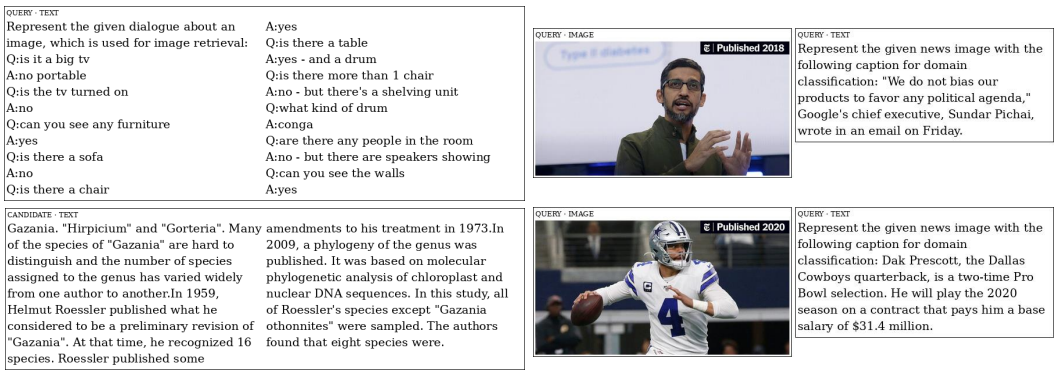
A.2 Retrieval-Guided Policy Search

Retrieval-guided policy search selects ψ^* under a visual-token budget. We construct a discrete policy space Ψ by varying the rendering and routing parameters while preserving the substrate-construction procedure in Section A.1. After filtering geometrically invalid configurations, each remaining policy is evaluated on a common policy-search set $\mathcal{V}$ with a frozen proxy retriever. The search returns the highest-scoring policy whose mean visual-token count does not exceed $\bar{\ell} = 480$, breaking score ties by lower visual-token cost.



(a) Example of classification (b) Example of visual grounding (c) Example of retrieval

Figure 5: Single-stack rendering examples produced under the selected canonical rendering policy ψ^* across multimodal embedding tasks: (a) classification, (b) visual grounding, and (c) retrieval.



(a) Examples of two-column text layout (b) Examples of side-by-side layout

Figure 6: Composition-specific layout examples produced under the selected canonical rendering policy ψ^*: (a) two-column text and (b) side-by-side text–image.

Table 5: Candidate layout templates. The schematics show the relative placement of element content. Exact dimensions are determined by the rendering policy and the input content.

Template	Schematic	Applicable unit	Routing decision
Single stack	Image / Text	Any supported composition	Baseline for every input. Retained when no applicable alternative satisfies its routing criteria.
Two-column text	Text | Text	A single text element	Selected when both columns are feasible, and the stacked layout is sufficiently tall.
Side-by-side	Image | Text	One text and one image element	Selected when both lanes are feasible, their card heights are balanced, and the plan reduces the visual-token count.

Candidate policies and feasibility. We instantiate each candidate policy by assigning one value to every parameter in Table 6. In particular, the routing thresholds allow different policies to select different layouts for the same input. Let p, $w_{\min}$, and $w_{\max}$ denote the card padding, minimum substrate width, and maximum substrate width of a policy, respectively. We retain the policy-level feasible subset

$$\Psi_{\text{feas}} = \{\psi \in \Psi : \; 2p < w_{\min} \leq w_{\max}\}. \tag{9}$$

Algorithm 1 Visual Substrate Rendering

Require: Retrieval unit x, rendering policy ψ
Ensure: Common visual substrate $\psi(x)$

Element Renderer
Let $\mathcal{U}(x) = (u_1, \ldots, u_{J_x})$ be the non-empty text and image elements of x in input order, where J_x is their count
Render each u_j as a typed visual card under ψ, with its header determined by the retrieval role of x and the element type
Adaptive Layout Router
Arrange the cards from top to bottom, yielding the order-preserving vertical plan π_{stack} with dimensions $(W_{\text{stack}}, H_{\text{stack}})$
$\pi \leftarrow \pi_{\text{stack}}$
if $\mathcal{U}(x)$ consists of a single text element **then**
    Wrap the text at the equal-column width and divide the resulting lines into two contiguous columns, yielding π_{text}
    if π_{text} is feasible and $H_{\text{stack}}/W_{\text{stack}} \geq \eta_{\text{text}}$ **then**
        $\pi \leftarrow \pi_{\text{text}}$
    end if
else if $\mathcal{U}(x)$ consists of one text element and one image element **then**
    Place the two cards in equal-width lanes in input order, yielding π_{mix} with card heights $h_{\text{T}}, h_{\text{I}}$
    if π_{mix} is feasible, $\max(h_{\text{T}}, h_{\text{I}})/\min(h_{\text{T}}, h_{\text{I}}) \leq \eta_{\text{mix}}$, and $\ell(\pi_{\text{mix}}) < \ell(\pi_{\text{stack}})$ **then**
        $\pi \leftarrow \pi_{\text{mix}}$
    end if
end if
Determine the canvas extent from π and place each rendered card in its assigned region
return the assembled visual substrate $\psi(x)$

This screening is applied whenever a policy is initialized or generated, while the feasibility of a composition-specific layout remains input dependent as defined in Section A.1.

Policy-search data and proxy scorer. We construct $\mathcal{V}$ from the training subsets of MoCa-CL-Pairs (Chen et al., 2026)[1]. Using random seed 42, we sample 100 query–positive pairs from each subset. The sampled queries form the query set, and their distinct positive candidates form the retrieval pool for that subset. The resulting policy-search data come only from training subsets and do not include final evaluation examples.

Candidate policies are evaluated with a frozen MoCa-3B encoder. Queries and candidates are encoded from their rendered substrates using the role-specific instructions "Represent this rendered query for retrieving a relevant document." and "Represent this rendered document for retrieval." The scorer applies last-token pooling and ℓ_2 normalization and computes cosine similarities. We compute Precision@1 within each subset and macro-average it across subsets to obtain $\mathcal{M}(\psi; \mathcal{V})$. The same sampled queries and distinct candidates are used to measure $\mathbb{E}_{x\sim\mathcal{V}}[\ell(\psi(x))]$, and policies exceeding $\bar{\ell} = 480$ are excluded from retrieval-guided search.

Evolutionary search. We solve the constrained policy-selection problem with the elitist procedure in Algorithm 2. At each round, the highest-scoring budget-feasible policies seed the next population. Crossover independently inherits each parameter from one of two elite policies. Mutation changes a numerical parameter to an adjacent candidate value or changes a categorical parameter to a different value in its candidate set. Remaining population slots are filled with distinct unevaluated policies from Ψ_{feas}. After $T = 8$ rounds, the search returns the budget-feasible policy with the largest $\mathcal{M}(\psi; \mathcal{V})$, breaking ties by lower mean visual-token cost. All stochastic operations use random seed 42.

A.3 Canonical Rendering Policy

Figure 7(a) shows the performance and visual-token cost of the evaluated candidate policies on $\mathcal{V}$. Their macro-averaged Precision@1 ranges from approximately 0.485 to 0.547, while their mean

[1] https://huggingface.co/datasets/moca-embed/MoCa-CL-Pairs

Algorithm 2 Retrieval-Guided Policy Search

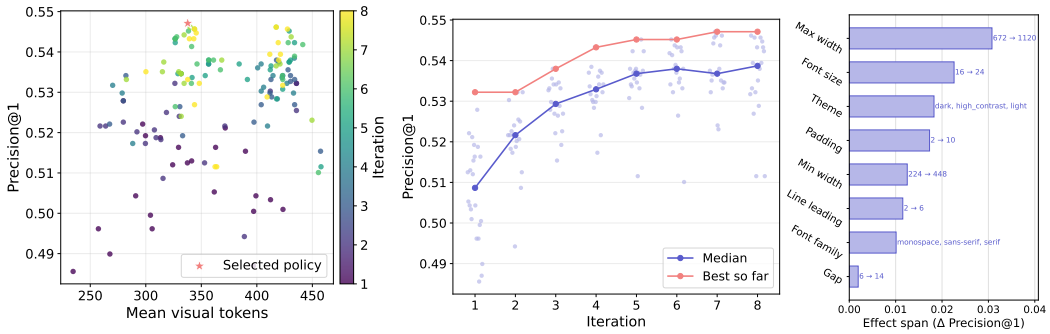
Require: Policy space Ψ, policy-search set $\mathcal{V}$, frozen proxy retriever, budget $\bar{\ell} = 480$, population size $n_{\text{pop}} = 24$, search rounds $T = 8$, elite count $n_{\text{elite}} = 6$, crossover rate $\rho = 0.5$, random seed $s = 42$
Ensure: Canonical rendering policy ψ^*

Construct $\Psi_{\text{feas}} = \{\psi \in \Psi : 2p < w_{\min} \le w_{\max}\}$
Initialize all stochastic operations with seed s
Population initialization
Draw n_{pop} distinct policies from Ψ_{feas} to form $\mathcal{P}$
$\mathcal{H} \leftarrow \varnothing$
Retrieval-guided search
for $g = 1, \ldots, T$ **do**
  for all $\psi \in \mathcal{P} \setminus \mathcal{H}$ **do**
    Render the policy-search set $\mathcal{V}$ with ψ
    Compute $\mathbb{E}_{x\sim\mathcal{V}}[\ell(\psi(x))]$
    if $\mathbb{E}_{x\sim\mathcal{V}}[\ell(\psi(x))] \le \bar{\ell}$ **then**
      Evaluate ψ with the frozen proxy retriever and set $\mathcal{M}(\psi; \mathcal{V})$ to its macro-averaged Precision@1
    else
      $\mathcal{M}(\psi; \mathcal{V}) \leftarrow -\infty$
    end if
    Add ψ to the evaluated set $\mathcal{H}$
  end for
  if $g < T$ **then**
    Let $\mathcal{E}$ be the n_{elite} budget-feasible policies in $\mathcal{P}$ ranked by $\mathcal{M}(\cdot; \mathcal{V})$
    Initialize the next population $\mathcal{P}' \leftarrow \mathcal{E}$
    while $|\mathcal{P}'| < n_{\text{pop}}$ and an unseen offspring is available **do**
      if two policies are available and a probability-ρ crossover is selected **then**
        Choose two policies from $\mathcal{E}$ and independently copy each parameter from either policy to form ψ_{new}
      else
        Choose one policy from $\mathcal{E}$ and one parameter
        if the selected parameter is numerical **then**
          Change it to an adjacent candidate value
        else
          Change it to a different categorical value
        end if
        Let the resulting policy be ψ_{new}
      end if
      if $\psi_{\text{new}} \in \Psi_{\text{feas}}$ and has not been evaluated or added **then**
        Add ψ_{new} to $\mathcal{P}'$
      end if
    end while
    Fill any remaining slots in $\mathcal{P}'$ with distinct unevaluated policies from Ψ_{feas}
    $\mathcal{P} \leftarrow \mathcal{P}'$
  end if
end for
return the budget-feasible policy in $\mathcal{H}$ with the largest $\mathcal{M}(\psi; \mathcal{V})$, breaking ties by lower $\mathbb{E}_{x\sim\mathcal{V}}[\ell(\psi(x))]$

visual-token counts range from approximately 230 to 460. As the search proceeds, the candidates progressively concentrate in the high-performance region. Among the budget-feasible evaluated candidates, the canonical rendering policy ψ^* attains the highest macro-averaged Precision@1, with ties resolved by lower visual-token cost. Defined by the values in Table 6, ψ^* uses 338 mean visual tokens and achieves a macro-averaged Precision@1 of 0.547. This selection shows that strong retrieval performance does not require the largest rendering budget.

Table 6: Rendering-policy search space and the selected canonical rendering policy.

Parameter	**Candidate values**	**Selected**
Minimum width (px)	224, 336, 400, 448	448
Maximum width (px)	672, 784, 896, 1008, 1120	896
Body font size (px)	16, 18, 20, 22, 24	22
Header font size (px)	10, 12, 14, 16	12
Line leading (px)	2, 4, 6	4
Card padding (px)	2, 4, 6, 8, 10	4
Card gap (px)	6, 8, 10, 14	6
Font family	Sans serif, serif, monospace	serif
Color theme	Light, high contrast, dark	high contrast
Single-text aspect-ratio threshold η_{text}	0.02, 0.05, 0.1, 0.15, 0.18	0.15
Mixed-card height-ratio threshold η_{mix}	1.5, 2.0, 3.0, 4.0, 6.0	2.0



(a) Overall distribution (b) Generation evolution (c) One-factor sensitivity

Figure 7: Rendering-policy search results on $\mathcal{V}$. (a) Candidate policies by macro-averaged Precision@1 and mean visual-token count, with color indicating the search iteration and the star marking the selected policy. (b) Evolution of the score distribution, median, and best-so-far policy across search iterations. (c) Sensitivity of the eight most influential factors around the selected policy, measured by the macro-averaged Precision@1 range obtained when varying each factor while holding the others fixed.

Figure 7(b) traces performance across search generations. The median macro-averaged Precision@1 rises from approximately 0.509 to 0.537, while the best-so-far score rises from approximately 0.532 to 0.546 during the early generations. Both stabilize after generation 5, with a final best score of approximately 0.547, indicating that the search has largely converged.

Figure 7(c) presents a one-factor sensitivity analysis around ψ^*. We vary one rendering parameter across its search range while holding all remaining parameters at their selected values, and measure the resulting range of macro-averaged Precision@1. The figure reports the eight most influential factors, while those with smaller observed ranges are omitted for clarity. Maximum width has the largest effect, with a score range of approximately 0.031, followed by body font size, color theme, and card padding. Minimum width, line leading, and font family have intermediate effects, while card gap has the smallest effect among the eight factors shown. These ranges measure sensitivity within the search space and do not imply that increasing a parameter monotonically improves performance.

B Additional Experimental Details

B.1 Training and Model Configuration

Training data. We train RenderEmbed on MoCa-CL-Pairs (Chen et al., 2026), which combines MMEB-train (Jiang et al., 2024) and ViDoRe-train (Faysse et al., 2025) in a single training corpus. All retrieval units are rendered before encoding, following the input construction described in Section 3.

Table 7: Model and training configuration for RenderEmbed.

Configuration	Value
Backbone checkpoint	https://huggingface.co/Qwen/Qwen2.5-VL-3B-Instruct
Anchors per query/candidate bank M	16
Fusion layers $\mathcal{I}_{\text{fuse}}$	$\{18, 24, 30, 36\}$
Embedding dimension H	2048
LoRA rank	32
LoRA scaling alpha	32
Adam β_1	0.9
Adam β_2	0.999
Learning rate	1×10^{-4}
Warmup steps	100
Global batch size	128
Training epochs	1
Temperature τ	0.02
Rendering-consistency weight λ_{rc}	0.1
Random seed	42

Table 8: Public sources for the baseline results reproduced in our experiments.

Model	Public source
GME (Zhang et al., 2024b)	`https://huggingface.co/Alibaba-NLP/gme-Qwen2-VL-7B-Instruct`
UniME-V2-2B (Gu et al., 2026)	`https://huggingface.co/TianchengGu/UniME-V2-Qwen2VL-2B`
UniME-V2-7B (Gu et al., 2026)	`https://huggingface.co/TianchengGu/UniME-V2-Qwen2VL-7B`
ColQwen2.5 (Faysse et al., 2025)	`https://huggingface.co/vidore/colqwen2.5-v0.2`
Jina-v4 (Günther et al., 2025)	`https://huggingface.co/jinaai/jina-embeddings-v4`
AGC (Qin et al., 2026)	`https://huggingface.co/hltcoe/AGC_qwen2.5-vl_colpali`

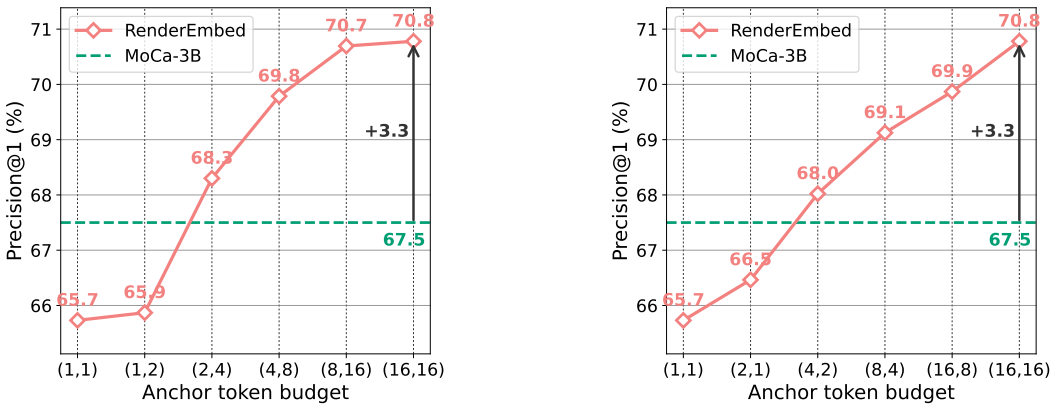
Model and optimization. RenderEmbed uses Qwen2.5-VL-3B as its backbone. We train the role-specific query and candidate anchor banks, the cross-layer fusion module, and LoRA adapters inserted into every LLM layer, while keeping all remaining parameters frozen. LoRA uses rank 32 and scaling factor $\alpha = 32$. Both anchor banks use $M = 16$ anchors, and the fusion module combines anchor states from layers $\mathcal{I}_{\text{fuse}} = \{18, 24, 30, 36\}$ into $H = 2048$-dimensional retrieval vectors. We optimize the model with Adam using $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of 1×10^{-4}, and 100 warmup steps. Training runs for one epoch with a global batch size of 128, temperature $\tau = 0.02$, rendering-consistency weight $\lambda_{\text{rc}} = 0.1$, and random seed 42. Table 7 summarizes these settings. We also provide our anonymized code in `https://anonymous.4open.science/r/RenderEmbed-anonymous-7614`.

Baseline result provenance. Table 8 lists the public checkpoints or implementations used for the baselines that we reproduce. All remaining baseline results reported in Tables 1 and 2 are taken from VLM2Vec (Jiang et al., 2024), MetaEmbed (Xiao et al., 2026), and MAGIC (Yuan et al., 2026b).

B.2 Intervention Construction

Rendering-controlled semantic negatives. For each training triple (q_i, d_i^+, d_i^-), we scan the elements of d_i^+ in their input order and select the first element whose same-type counterpart in d_i^- has different content. Since each candidate contains at most one text element and one image element, the element type uniquely determines the corresponding replacement. We construct the rendering-controlled negative $\hat{d}_i^-$ by replacing only the selected content while preserving all untouched elements and the canvas, card geometry, column structure, and typography determined by the canonical rendering policy ψ^*. Replacement text is limited to the length of the original text and clipped within its content region. Replacement images are fitted proportionally within the same region without cropping or upscaling. Every training triple admits such a replacement. For robustness to inputs without one, we fall back to d_i^- rendered with ψ^*.

Alternative rendering policies. We construct the alternative rendering-policy pool by randomly sampling ten distinct non-canonical policies from the budget-feasible policies whose retrieval perfor-



(a) Candidate-heavy allocation: $M_q \leq M_d$

(b) Query-heavy allocation: $M_q \geq M_d$

Figure 8: Asymmetric test-time anchor budgets on MMEB. Each pair (M_q, M_d) denotes the number of retained query and candidate anchors. Corresponding asymmetric settings across the two panels keep $M_q M_d$ fixed and therefore require the same computational overhead. Beyond the smallest budget, allocating more anchors to candidates consistently yields higher Precision@1. The dashed line marks the MoCa-3B baseline.

mance on the policy-search queries exceeds the mean performance of all policies evaluated during the search. The selected policies vary in visual style, layout geometry, and routing rules. At each training step, we sample one policy ψ' for each local batch and apply it to every positive candidate d_j^+ in that batch, producing $\psi'(d_j^+)$. The sampled policy preserves the content and order of all elements, while each query continues to use the canonical rendering $\psi^*(q_i)$.

B.3 Retrieval Efficiency

To isolate retrieval-stage efficiency, we construct a shared BF16 workload with 1,000 queries and 100,000 candidates. For each model, M_q and M_d denote the average numbers of query and candidate vectors, respectively, H denotes the embedding dimension, and $\mathcal{C}$ denotes the candidate collection. Since each BF16 value occupies two bytes, the candidate-side index size is $2|\mathcal{C}|M_d H$ bytes. Counting a multiply-add as two FLOPs, scoring one query against the complete collection requires $2|\mathcal{C}|M_q M_d H$ FLOPs. In Table 4, average vector counts are rounded for display, while FLOPs use the unrounded averages. For late-interaction scoring latency, we measure execution on an NVIDIA H20 using query and candidate batch sizes of 1,024. We report latency per query, averaged over the 1,000-query workload and ten timed runs after three warm-up runs. The timing includes pairwise dot products and MaxSim reduction while excluding encoding overhead.

C Additional Experimental Results

C.1 Asymmetric Test-Time Anchor Budgets

Figure 8 extends the symmetric-budget analysis by varying the query and candidate budgets separately. At the smallest asymmetric budget, expanding the query representation is more beneficial. Once the query side retains several anchors, allocating additional anchors to candidates consistently produces stronger retrieval at matched computational cost. In particular, preserving the full candidate budget while reducing the query budget nearly maintains full-budget performance, whereas the converse allocation incurs a larger degradation. This pattern suggests that candidate coverage becomes the primary bottleneck after the query is represented by a sufficient number of anchors, since each query vector benefits from a richer set of candidate evidence under MaxSim. The two allocation directions, therefore, expose a practical trade-off: candidate-heavy settings favor retrieval quality, whereas query-heavy settings reduce candidate-index storage.

C.2 Detailed Results on MMEB

We present the detailed results of RenderEmbed and some representative baseline models on the MMEB benchmark in Table 9, covering 36 tasks across four categories: classification, VQA, retrieval, and visual grounding.

Table 9: Detailed performance on 36 MMEB tasks Jiang et al. (2024). We show the results of our RenderEmbed and baseline models. Rows in yellow indicate metrics of an OOD task.

Task	CLIP	SigLIP	BLIP2	VLM2Vec	GME	MMRet	UniME-V2	MoCa-3B	RenderEmbed
Classification (10 tasks)									
ImageNet-1K	55.8	45.4	10.3	74.5	66.0	58.8	75.7	75.4	81.1
N24News	34.7	13.9	36.0	80.3	52.0	71.3	43.3	80.9	79.6
HatefulMemes	51.1	47.2	49.6	67.9	51.8	53.7	61.9	70.6	64.5
VOC2007	50.7	64.3	52.1	91.5	79.0	85.0	46.1	87.0	85.7
SUN397	43.4	39.6	34.5	75.8	71.3	70.0	63.9	74.8	78.4
Place365	28.5	20.0	21.5	44.0	40.3	43.0	34.7	38.8	44.4
ImageNet-A	25.5	42.6	3.2	43.6	40.8	36.1	48.2	39.7	42.3
ImageNet-R	75.6	75.0	39.7	79.8	83.1	71.6	85.1	75.4	82.8
ObjectNet	43.4	40.3	20.6	39.6	71.2	55.8	54.3	31.3	60.5
Country-211	19.2	14.2	2.5	14.7	30.1	14.7	5.9	24.0	23.2
All Classification	42.8	40.3	27.0	61.2	58.6	56.0	51.9	59.8	64.3
VQA (10 tasks)									
OK-VQA	7.5	2.4	8.7	69.0	33.1	73.3	57.5	40.0	70.7
A-OKVQA	3.8	1.5	3.2	54.4	21.9	56.7	50.5	54.6	59.3
DocVQA	4.0	4.2	2.6	52.0	44.2	78.5	88.1	93.0	95.3
InfographicsVQA	4.6	2.7	2.0	30.7	22.4	39.3	56.5	67.7	73.4
ChartQA	1.4	3.0	0.5	34.8	18.0	41.7	47.3	64.1	68.7
Visual7W	4.0	1.2	1.3	49.8	23.6	49.5	55.2	61.6	58.5
ScienceQA	9.4	7.9	6.8	42.1	29.0	45.2	41.5	45.4	52.7
VizWiz	8.2	2.3	4.0	43.0	40.1	51.7	40.3	52.3	54.1
GQA	41.3	57.5	9.7	61.2	77.0	59.0	55.8	66.9	72.4
TextVQA	7.0	1.0	3.3	62.0	48.6	79.0	71.4	83.1	84.3
Avg. VQA	9.1	8.4	4.2	49.9	35.8	57.4	56.4	62.9	68.9
Retrieval (12 tasks)									
VisDial	30.7	21.5	18.0	80.9	65.9	83.0	33.2	80.5	81.0
CIRR	12.6	15.1	9.8	49.9	60.0	61.4	57.4	55.7	65.2
VisualNews_t2i	78.9	51.0	48.1	75.4	81.6	74.2	69.2	74.4	74.3
VisualNews_i2t	79.6	52.4	13.5	80.0	84.2	78.1	79.9	77.8	78.7
MSCOCO_t2i	59.5	58.3	53.7	75.7	72.7	78.6	76.9	76.4	77.8
MSCOCO_i2t	57.7	55.0	20.3	73.1	59.8	72.4	70.6	72.6	74.9
NIGHTS	60.4	62.9	56.5	65.5	68.4	68.3	65.4	67.4	69.0
WebQA	67.5	58.1	55.4	87.6	91.3	90.2	89.4	90.6	89.9
FashionIQ	11.4	20.1	9.3	16.2	39.3	54.9	14.0	22.2	25.0
Wiki-SS-NQ	55.0	55.1	28.7	60.2	77.1	24.9	47.1	73.3	80.3
OVEN	41.1	56.0	39.5	56.5	79.0	87.5	65.4	75.9	75.3
EDIS	81.0	23.6	54.4	87.8	97.6	65.6	85.3	80.8	91.9
Avg. Retrieval	53.0	31.6	33.9	67.4	73.1	69.9	62.8	70.6	73.6
Visual Grounding (4 tasks)									
MSCOCO	33.8	46.4	28.9	80.6	48.0	76.8	57.9	80.2	67.9
RefCOCO	56.9	70.8	47.4	88.7	65.3	89.8	82.6	92.1	92.5
RefCOCO-matching	61.3	50.8	59.5	84.0	82.3	90.6	88.8	92.8	87.4
Visual7W-pointing	55.1	70.1	52.0	90.9	73.7	77.0	67.8	89.5	85.1
Avg. Grounding	51.8	59.5	47.0	86.1	67.3	83.6	74.3	88.7	83.2
Final Score (36 tasks)									
All IND Avg.	37.1	32.3	25.3	67.5	55.8	59.1	62.3	72.3	74.7
All OOD Avg.	38.7	38.0	25.1	57.1	60.9	68.0	55.5	61.5	65.9
All Tasks Avg.	37.8	34.8	25.2	62.9	58.0	64.1	59.3	67.5	70.8