MoEMB: Scaling Universal Multimodal Embeddings with Efficient Mixture-of-Experts Models

Anonymous authors
Paper under double-blind review

Abstract

Universal multimodal embedding (UME) increasingly demands encoder's capacity for handling a broad range of tasks and modalities with increased complexity. Prior scaling methods either increase the representation size, retrieval effort, or scales the encoder into a heavy multimodal LLM. Recent works, such as Think-Then-Embed (TTE), explore scaling via reasoning tokens. However, embedding models are hard to scale up: increasing parameters directly trades off against the large training batch size that contrastive learning needs, and retrieval has to be served under tight latency. Moreover, UME tasks are diverse in complexity, where scaling up embedders can bring significantly more redundant computation for simple retrievals. In this work, we propose MoEMB, which instead scales UME along the *expert* axis through mixture-of-experts (MoE), growing encoder capacity while preserving single-vector, non-autoregressive encoding. Through a systematic study of the design space and training recipes for MoE-based UME, MoEMB sets a new state of the art on both MMEB-V2 and MRMR among models trained on public MMEB-family data: with only $\sim$3B active parameters, MoEMB surpasses TTE-based methods with $> 4\times$ active parameters, using significantly less compute. To further improve the scalability and efficiency, we conduct the first comprehensive study of adaptive computation for MoE-based embedding, spanning diverse strategies across training-based and inference-only methods. Together, these results support expert scaling as an effective and efficient direction for UME, with adaptive computation further improving efficiency for MLLM-based embedding models towards real-world, large-scale retrieval systems.

1 Introduction

Embedding models map heterogeneous inputs into a shared space for retrieval. While earlier embedding tasks focus primarily on simple symmetric retrieval, recent universal multimodal embedding (UME) tasks now require a single encoder to handle diverse modalities, with diverse tasks spanning classification, question answering, retrieval, grounding, and document search (Meng et al., 2025). As the embedding tasks grow more varied and complex, performance depends more on the encoder's capacity for visual reasoning and instruction following. However, most work on scaling embeddings still targets the stored representation instead of that capacity (Fig. 2). Matryoshka Representation Learning adjusts the embedding dimensionality (Kusupati et al., 2022), while multi-vector retrieval methods (Khattab & Zaharia, 2020; Faysse et al., 2025; Xiao et al., 2025) keep multiple vectors per input for finegrained retrieval. These methods scale storage and retrieval budget, but they leave the encoder itself untouched, and so cannot supply the understanding and reasoning that harder UME tasks demand.

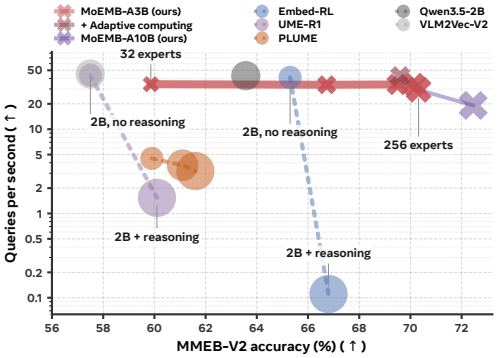

Figure 1: **Accuracy vs. QPS.** Scaling via MoE (MoEMB) raises accuracy by > 10 points at near-constant throughput. Details can be found in D.2.

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.

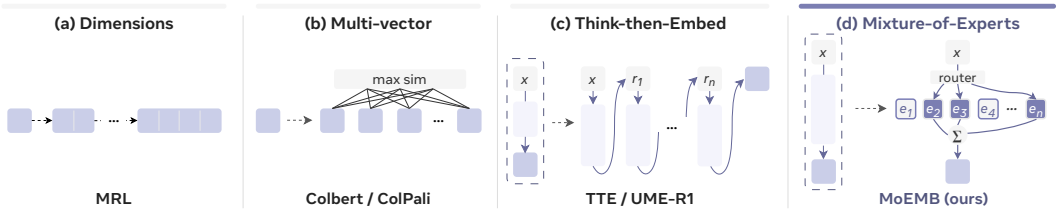

Figure 2: **Four ways to scale multimodal embeddings.** Dimension (a) and multi-vector (b) methods scale the output representation; TTE-based methods (c) scale via intermediate reasoning tokens. MoEMB (d) instead scales the encoder along the expert axis through sparse MoE.

To scale the encoder itself, a line of work turns multimodal large language models (MLLMs) into embedders, and these now lead UME benchmarks (Jiang et al., 2025; Lin et al., 2025; Li et al., 2026). However, even at larger scales than CLIP-based (Radford et al., 2021) lightweight models, these MLLM-based embedders still struggle to cover the diverse tasks of UME. We empirically find (Fig. 3) that one embedder trained jointly over all tasks is outperformed by the same model trained per task; Qwen3-VL-Embedding likewise reports degraded MTEB (Muennighoff et al., 2023) performance against text-only Qwen3-Embedding (Zhang et al., 2025b). Two explanations are possible: destructive gradient interference between tasks, or a model simply not capacious enough to serve them all at once. Both require further scaling up the embedder's capacity.

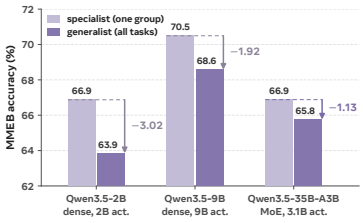

Figure 3: Performance degradation for switching from specialist to generalist shrinks with model's total capacity. Each model trained on the full MMEB mixture (generalist) vs. one task alone (specialist). Details in App. E.2.

A recent line of think-then-embed (TTE) methods scale for reasoning-heavy retrieval by generating intermediate reasoning tokens before producing the final embedding (Cui et al., 2025; Lan et al., 2025; Cheng et al., 2026). However, these methods improve performance with a significant cost of computation. For instance, on the text partition of MMEB-V3 (Huang et al., 2026), it takes 1.7 days for a regular MLLM-based 2B encoder (Meng et al., 2025) to complete evaluation, while UME-R1-2B (Lan et al., 2025), a TTE-based model, needs roughly **7.8 H200-days** to encode queries, and **431 H200-days** for candidates, which is prohibitively expensive for real-world deployment.

In general, scaling a UME encoder is difficult: (1) search and recommendation systems are required to embed astronomically many queries and items, under low latency; (2) contrastive training encodes and backpropagates through both the query and the target, so the activation memory is doubled; (3) added parameters trade against batch size, a key component of contrastive quality; and (4) UME tasks vary widely in complexity: for simpler tasks the extra LLM backbone may be wasteful, since the MLLM's own contrastively pretrained vision encoder (Radford et al., 2021) could already suffice.

To address challenges (1–3), we propose MoEMB, the first Mixture-of-Experts (MoE) UME model. Instead of scaling the embedding dimension, dense parameters, or reasoning tokens, we scale the encoder along the *expert* axis. Sparse MoE is well studied for language modeling (Jiang et al., 2024; Team et al., 2026) but remains underexplored for embedding models, for which we fill the gap. We study three approaches to obtain an MoE-based embedder: (1) adapting a pretrained native sparse MoE, (2) dense-to-MoE upcycling, and (3) modality-level MoE; to our knowledge, MoEMB is the first to adapt and scale pretrained sparse multimodal MoE backbones for UME. Expert scaling gains over 10 points at negligible additional compute (Fig. 1): with only $\sim$3.1B active parameters, MoEMB-A3B reaches a new SOTA of 70.4 on MMEB-V2, surpassing all UME models trained on the MMEB V2 dataset, including TTE-based encoders that require significantly more computes.

Next, to tackle issue (4), heterogeneous complexity across UME tasks, we conduct the first comprehensive study of adaptive computation for MoE-based UME embedders. Prior adaptive computation primarily focus on text generation (Huang et al., 2025a; Chen et al., 2024; Bai et al., 2025) and traditional vision tasks like classification (Yin et al., 2022), with little work on embeddings. Moreover, these works typically study one mechanism at a time. In this work, we study five actions — token pruning, layer skipping, expert skipping, adaptive top-k, and expert pruning — under both training-based and inference-only settings. Token pruning and adaptive top-k give the strongest accuracy–compute trade-off, and inference-only variants are competitive with their trained counterparts. Applying them saves notable compute at under one point of accuracy.

Our contributions are twofold.

1. Expert scaling for UME. (a) We introduce expert capacity as a scaling axis for UME and conduct the first comprehensive study of its model designs and training recipes. **(b)** Without any special data mining or sophisticated multi-stage training, MoEMB sets a new SOTA on two large-scale benchmarks, MMEB-V2 and MRMR (Zhang et al., 2025a), among models trained only on public MMEB-family data, surpassing TTE-based encoders that consume significantly more compute.

2. Adaptive computation for MoE-based UME. (a) We conduct the first systematic exploration for adaptive computation for MoE-based UME embedders, across five actions and both training-based and inference-only methods. **(b)** By leveraging adaptive computation methods, we substantially reduce forward compute for $< 1\%$ performance degradation. **(c)** We are the first to explore token pruning based on Gated Delta Networks' β as a strong alternative to attention scores. **(d)** We provide corresponding inference implementation for realizing saved computes into wall-clock saving.

2 Preliminaries

A UME embedder maps a multimodal, instruction-conditioned query or target $\{q, t\} = (I_x, I_v, I_t)$ composed of a task instruction I_x, optionally visual inputs I_v and a text input I_t, to a single vector (Meng et al., 2025). We build it from a pretrained multimodal LLM: one encoder F_θ processes both queries t and targets t:

$$e_\theta(\{q,t\}) = \mathrm{norm}\left(\mathrm{Pool}(F_\theta(\{q,t\}))\right),$$

where norm is the normalization operation, and Pool is the pooling operation that produces the final embedding representation (Cui et al., 2025). For a batch of N pairs, we follow the standard one-directional InfoNCE objective (Jiang et al., 2025):

$$\mathcal{L}_{\text{emb}} = -\frac{1}{N}\sum_i \log \frac{e^{S_{ii}}}{\sum_j e^{S_{ij}}}, \quad S_{ij} = \cos(q_i, t_j)/\tau \tag{1}$$

where τ denotes the temperature, and cos is the cosine similarity function.

3 Building Sparse MoE UME Models

3.1 MoE Backbone Designs

We explore three ways to build a sparse MoE-based embedder: **(1) Native MoE:** we reuse a backbone whose experts and router are learned in standard language model pretraining; **(2) Dense-to-sparse Upcycling:** we convert a dense MLLM into sparse MoE during contrastive training; **(3) modality-level routing:** we perform routing on the *modality* level.

(1) Native MoE. We directly reuse an MLLM whose feed-forward blocks are already sparse MoE, composed of a shared expert $\mathrm{FFN}_{\text{shared}}$ and E sparse experts. For a token's hidden state $h \in \mathbb{R}^d$, the pretrained router $W_r \in \mathbb{R}^{E\times d}$ scores all E experts and keeps the k highest, $\mathcal{S} = \mathrm{TopK}(W_r h,\, k)$. Their scores are renormalized over $\mathcal{S}$, while a learned vector $w_{\text{shared}} \in \mathbb{R}^d$ scales the shared expert:

$$\mathrm{MoE}(h) = \gamma\,\mathrm{FFN}_{\text{shared}}(h) + \sum_{j\in\mathcal{S}} \alpha_j\,\mathrm{FFN}_j(h), \qquad \begin{aligned}\alpha_j &= \frac{e^{(W_r h)_j}}{\sum_{u\in\mathcal{S}} e^{(W_r h)_u}},\\ \gamma &= \mathrm{sigmoid}\left(w_{\text{shared}}\, h\right).\end{aligned} \tag{2}$$

This variant retains MoE pretrained from language modeling. Empirically we find this setting leads to the best performance. We therefore use it as the default setting for all subsequent experiments.

(2) Dense-to-MoE Upcycling. We explore upcycling a dense MLLM into a sparse MoE embedder by splitting its feed-forward block and replicating the pieces into E experts, then learn a router over them from scratch. We empirically find its performance to lag behind the native MoE setting. Therefore, we report its best variant in Tab. 3 and provide further details in App. E.1.

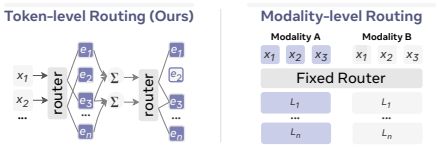

Figure 4: Token- vs. modality-level routing.

(3) Modality-level Experts. As shown in Fig. 4, we explore an alternative routing setting, where we replicate the dense feed-forward blocks into modality-level experts, where a fixed router forwards all tokens from one modality to its corresponding expert. This is in contrast to the regular *token-level routing*, where each token is dynamically routed to its top-k experts. We find this variant slightly improves over the dense baseline, but lags behind the native MoE setting (Tab. 3).

3.2 Pooling, router adaptation, and training dynamics

Beyond determining an effective, MoE backbone design, and adapting a pretrained MoE model to a contrastive embedding objective also requires other design and training choices. In this work we explore three aspects: *pooling*, *router adaptation*, and *training dynamics* for MoE-UME.

Pooling Strategy. Previous UME works (Cui et al., 2025) have discovered that pooling via last token often outperforms other pooling methods like mean (Lee et al., 2025) or latent attention pooling (Lee et al., 2024). Building on this observation, we only compare two designs: a simple last token pooling, versus a designated `<emb>` token. Empirically we find the latter to perform better (Tab. 4).

Router Adaptation and Auxiliary Objectives. Because pre-trained routers establish delicate token-to-expert assignments, downstream contrastive fine-tuning can destabilize learned expert allocation. To systematically evaluate router stability versus adaptability, we compare frozen and trainable router configurations alongside three auxiliary objectives: (1) *load-balancing and z-losses* to prevent expert collapse; (2) a *conflict-aware loss* to mitigate cross-task gradient conflict by penalizing routing distribution overlap across conflicting tasks; and (3) *task conditioning*, which incorporates discrete task labels or continuous instruction embeddings alongside token hidden states into router logit computation. Full details for each formulation are provided in App. C.2. Empirically, we find that the simple frozen router instead yields the best performance, which indicates that the routers learned during language pre-training are also near-optimal for the contrastive objective (Tab. 4).

Adaptation Dynamics. We systematically ablate the hyperparameter settings governing contrastive adaptation for MoE-based UME: global batch size, contrastive temperature, LoRA rank, learning rate, and training epochs (Fig. 6).

4 Adaptive Computation for MoE-UME

While sparse MoE improves the quality–compute tradeoff, it still runs a fixed per-token compute budget with potential computation redundancy. To further improve efficiency and scalability for MoE-based UME models, we explore various adaptive computation actions. Fig. 5 illustrates the six types of adaptive computing actions we study, covering both training-based (§4.1) and inference-only (§4.2) settings. Tab. 1 summarizes different methods we studied. Due to the space limit, we discuss the most effective methods in the following sections, and leave the rest to App. G and F.1.

4.1 Training-Based Adaptive Computation Methods

Token pruning. We study two types of token pruning: *one-shot* and *gradual* pruning. A *one-shot* pruning makes a single pruning decision at an early layer (Huang et al., 2025a) K; a *gradual* pruning applies a per-token gate at every layer, so a token's alive mass $\alpha_{t,\ell} = \prod_{j<\ell}(1 - p^{\text{drop}}_{t,j})$ decays with depth and its downstream compute is removed progressively. We train one-shot pruning with a straight-through estimator (STE). For gradual pruning, we explore both STE and an AViT-style (Yin et al., 2022) halting mechanism, where both methods failed due to unstable training (App. F.1). Therefore, we realize gradual pruning with the GRPO policy of App. F.1, which samples the discrete action instead of differentiating through it. Both settings distill from a separate teacher embedding obtained without adaptive computing: e^{full}. Given the target budget $\rho^\star$ and the model's current keep rate ρ_{keep}, the overall loss becomes:

$$\mathcal{L}_{\text{drop}} \;=\; \mathcal{L}_{\text{emb}} \;+\; \lambda_{\text{d}}\big(1 - \cos(e,\, e^{\text{full}})\big) \;+\; \lambda_{\text{b}}\big(\rho_{\text{keep}} - \rho^\star\big)^2.$$

Learned expert skipping. Inspired by Mixture-of-Depths (Raposo et al., 2024), we add a skip gate $p^{\text{skip}}_{t,\ell} \in [0, 1]$ over existing routers, which determines whether to skip the entire experts computation:

$$\mathcal{L}_{\text{skip}} \;=\; \mathcal{L}_{\text{emb}} \;+\; \lambda_{\text{c}}\big(1 - \rho_{\text{skip}}\big), \qquad \rho_{\text{skip}} \;=\; \frac{1}{L\,|\mathcal{T}|}\sum_{\ell=1}^{L}\sum_{t\in\mathcal{T}} p^{\text{skip}}_{t,\ell},$$

with L being the number of MoE layers, $|\mathcal{T}|$ denoting the number of tokens, and $p^{\text{skip}}_{t,\ell}$ the normalized probability produced by each skip gate. The skip budget ρ_{skip} applies to the entire model, which allows the skip rate to differ across layers (Fig. 19). As a hard skip gate is non-differentiable, we train it through a soft fade. Let $u_{t,\ell}$ be the output of the routed experts for token t at layer ℓ, and $v_{t,\ell}$ the rest of that layer's output; the routed output $h_{t,\ell}$ is scaled by the non-skip probability $1 - p^{\text{skip}}_{t,\ell}$:

$$h_{t,\ell} \;=\; v_{t,\ell} \;+\; \big(1 - p^{\text{skip}}_{t,\ell}\big)\, u_{t,\ell}.$$

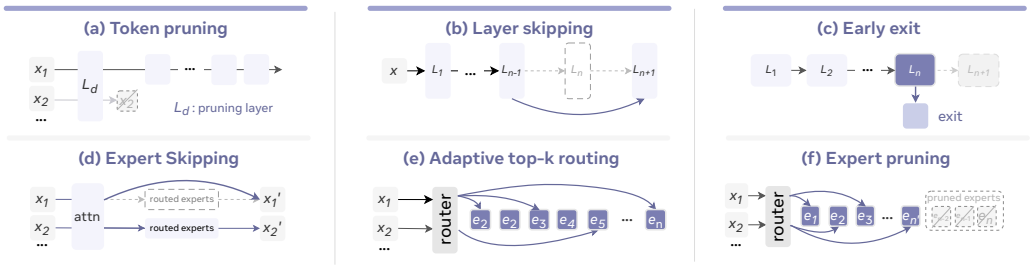

Figure 5: We consider six adaptive computing actions in MoEMB. The top-three (a-c) are general adaptive computing actions, while the bottom-three (d-f) are specific to MoEs.

Table 1: Summarization for different designs. *Train* is training-based, *Inf.* means inference-only. *Basis* names the work a method is adapted from, or *ours* where we know of no prior counterpart. †marks those detailed in Apps. G and F.1 instead of in the main text.

Action	Method	Regime	Basis
Token pruning	One-shot pruner after decoder block n (Gumbel/STE)	Train	Huang et al. (2025a)
	Layer-wise gradual pruning with GRPO†	Train	ours
	GDN β-ranked one-shot drop	Inf.	ours
Expert skipping	Per-token three-way skip gate (soft fade)	Train	ours
	GDN β-ranked expert skip	Inf.	ours
	Per-layer Otsu split on β or router logit†	Inf.	Otsu (1979)
Adaptive top-k	Null experts, load-balance–scheduled†	Train	Zeng et al. (2024)
	Dynamic modality thresholding	Inf.	Huang et al. (2025b)
	Fixed router-score threshold†	Inf.	ours
Layer skip	Trained layer-skip gate (STE, distillation)†	Train	ours
Early exit	Multi-depth contrastive loss	Train	ours
Expert pruning	Routing frequency, router probability mass	Inf.	Lu et al. (2024)
	MAN, MSAN, REAP saliency	Inf.	Liu et al. (2026); Lasby et al. (2025)

Early exit. We supervise a set of candidate exit depths $\mathcal{D}$ within the LLM backbone. Let q_d, t_d denote the query and target embeddings taken at depth d, and $\bar{q}_L, \bar{t}_L$ the full-depth embeddings, precomputed from same pre-trained embedder. The objective has two parts, $\mathcal{L}_{\text{exit}} = \mathcal{L}_{\text{con}} + \lambda_{\text{a}}\mathcal{L}_{\text{dis}}$: the mean contrastive loss at every depth, and a distillation loss pulling every embedding into a shared space onto its cached teacher:

$$\mathcal{L}_{\text{con}} = \frac{1}{|\mathcal{D}^+|}\sum_{d\in\mathcal{D}^+}\mathcal{L}_{\text{emb}}(q_d, t_d), \qquad \mathcal{L}_{\text{dis}} = \frac{1}{2|\mathcal{D}|}\sum_{d\in\mathcal{D}}\Big[\big(1-\cos(q_d,\bar{q}_L)\big)+\big(1-\cos(t_d,\bar{t}_L)\big)\Big],$$

where $\mathcal{L}_{\text{emb}}(\cdot,\cdot)$ is Eq. 1 evaluated on the given pair of embedding sets.

4.2 Inference-Based Efficiency

Token importance without an attention map. Inference-only token pruning conventionally ranks visual tokens by the attention mass they receive (Chen et al., 2024), which presumes the attention matrix is available. However, as the efficient FlashAttention kernels (Dao et al., 2022) do not materialize the attention map, we need to find an alternative training-free signal for token importance. In this work, we propose to leverage the writing strength β in Gated DeltaNet (GDN) (Yang et al., 2024), which is used in recent frontier MLLMs (Qwen Team, 2026; Team et al., 2026). At each GDN layer the gated delta rule updates a recurrent memory S using a token-dependent write strength. Within one head of one GDN layer, the rule keeps a memory $S_t \in \mathbb{R}^{d_v\times d_k}$ and updates it as

$$S_t = S_{t-1}\big(\alpha_t(I-\beta_t\,k_t k_t^\top)\big) + \beta_t\,v_t k_t^\top, \qquad \beta_t = \text{sigmoid}\left(\mathbf{W}_\beta^\top x_t\right) \in (0,1), \tag{3}$$

where x_t is the token's hidden state, $k_t \in \mathbb{R}^{d_k}$ and $v_t \in \mathbb{R}^{d_v}$ its ℓ_2-normalized key and its value, $\alpha_t \in (0,1)$ the forget gate, and I the identity. Thus, β_t interpolates between leaving the memory untouched ($\beta_t \to 0$) and fully overwriting its k_t direction with v_t ($\beta_t \to 1$), which provides a natural

surrogate for token importance. Averaging β_t over those layers and over heads gives each token an importance score, which drives every inference-only token and expert decision below.

Token pruning. We utilize the β score to prune visual tokens. After decoder block K we rank the visual tokens by β_t, keep the highest-scoring fraction, and discard the rest for every layer that follows.

Adaptive top-k. Let $\alpha_{\ell t(1)} \geq \cdots \geq \alpha_{\ell t(k)}$ denote the sorted routing weights for token t at layer ℓ (Eq. 2). Adaptive top-k routing selects a dynamic number of experts $k_{\ell t} \leq k$ per token via a predefined condition. For instance, MoDES (Huang et al., 2025b) keeps an expert m if its scaled routing weight satisfies $s_\ell \alpha_{\ell t(m)} \geq \tau$, for which the global threshold τ and layer-scaling factors s_ℓ are estimated during a calibration pass (App. F.2). Here, τ acts as a uniform decision threshold, while s_ℓ upweights critical layers to retain more experts where capacity matters the most.

Total expert pruning. We consider pruning experts from the model permanently, which reduces memory instead of compute. Each layer keeps the E' experts with the largest saliency ϕ_j, measured on a calibration pass. We compare two families of saliency. *Router-based* scores read the router alone (Lu et al., 2024), either as the fraction of calibration tokens routed to expert j (routing frequency) or as its mean router probability. *Output-based* scores instead read the output produced by the expert, averaged over the tokens routed to it,

$$\phi_j \;=\; \underset{t \to j}{\text{mean}}\; w_{tj}, \qquad w_{tj} \in \Big\{ \underbrace{\|o_{tj}\|}_{\text{MAN}},\; \underbrace{\|o_{tj}\|^2}_{\text{MSAN}},\; \underbrace{\alpha_{\ell tj}\|o_{tj}\|}_{\text{REAP}} \Big\}, \qquad o_{tj} = \text{FFN}_j(h_{\ell t}), \tag{4}$$

where o_{tj} is the expert's output and $\alpha_{\ell tj}$ its routing weight from Eq. 2. The three output-based scores differ only in how they weight that output. MAN and MSAN (Liu et al., 2026) take its norm and squared norm, ignoring the router entirely, so an expert is judged purely by how large an output it produces. REAP (Lasby et al., 2025) additionally scales the norm by the routing weight. Empirically, we find this variant to generally perform better (Fig. 7).

5 Experiments

5.1 Experimental Setup

Models and training. We adapt the Qwen3.5-35B-A3B and Qwen3.5-122B-A10B multimodal MoEs (Qwen Team, 2026). All training is done with two epochs at length 8192 in BF16, global batch 512, AdamW at learning rate 10^{-4} with ten warmup steps and cosine decay, and LoRA of rank 64 and scale 128. We use Nvidia Megatron Core MoE (Liu et al., 2025) with Expert Parallelism for accelerated MoE-UME training. App. C gives full implementation details.

Benchmarks. We evaluate our models on two comprehensive and large-scale multimodal retrieval benchmarks: MMEB V2 (Meng et al., 2025) and MRMR (Zhang et al., 2025a). **MMEB-V2** (Meng et al., 2025) comprises 78 tasks spanning image, video, and visual document (VisDoc) retrieval tasks. **MRMR** (Zhang et al., 2025a) is a reasoning-intensive multimodal retrieval benchmark, which requires multi-step logical inference and complex domain-specific interpretation across multimodal contexts (e.g., cross-modal knowledge retrieval, theorem identification, and contradiction detection). We follow the official evaluation setup for both benchmarks.

5.2 Results and Ablations on MoE Design and Training Dynamics for MoE-UME

Main results. Tab. 2 shows that MoEMB achieves superior performance on MMEB V2, outperforming all public-data baselines across model scales despite using fewer active parameters and avoiding costly autoregressive reasoning passes. As shown in Fig. 1, expert scaling yields a significantly better accuracy–throughput trade-off than TTE-based methods: expanding total expert capacity delivers substantial accuracy gains with negligible throughput loss. Furthermore, on MRMR (Tab. 5), MoEMB-A3B achieves the top overall score despite its smaller active footprint, outperforming TTE methods designed for reasoning-intensive retrieval. This confirms that scaling model capacity via MoE is more effective than test-time reasoning scaling. Moreover, as MMEB V2 is largely OOD, many frontier models rely on external corpora mirroring evaluation formats, inflating scores through data leakage. On evaluation splits where MMEB V2 provides in-domain supervision (e.g., ViDoRe and VisRAG), MoEMB achieves SOTA performance without external adaptation data, matching or exceeding significantly larger models trained on massive external data and complex pipelines.

Table 2: MMEB-V2 results. **bold**/underline mark best/second within the public-data group. †Embed-RL pairs a shared 8.77B autoregressive reasoner with a 4.44B/2.13B embedder, giving 13.2B/10.9B active parameters. ‡These models do not disclose an active parameter count. CLS: classification, QA: question answering, RET: retrieval, GD: grounding, MRT: moment retrieval; V1/V2: ViDoRe-v1/v2, VR: VisRAG, OOD: out-of-domain.

Model	Act. (B)	Image					Video					VisDoc					All
		CLS	QA	RET	GD	**Ov**	CLS	QA	RET	MRT	**Ov**	V1	V2	VR	OOD	**Ov**	
# Datasets→		10	10	12	4	36	5	5	5	3	18	10	4	6	4	24	78
Models adapted with external data																	
∼2–4B active params																	
DME-2B	2.2	73.0	73.8	76.2	90.7	76.3	82.2	57.6	53.5	45.6	61.3	86.0	59.3	92.5	72.0	80.9	74.2
Qwen3-VL-Embedding-2B	2.1	70.3	74.3	74.8	88.5	75.0	71.9	64.9	53.9	53.3	61.9	84.4	65.3	86.4	69.4	79.2	73.2
RzenEmbed-v1-2B	2.2	65.3	61.7	73.8	77.8	68.5	45.6	47.5	38.3	36.7	42.6	87.0	57.6	85.4	67.1	78.4	65.6
Ops-MM-Embedding-v1-2B	2.2	68.1	65.1	69.2	80.8	69.0	53.6	55.6	41.7	33.7	47.6	76.4	53.2	77.6	65.1	71.0	64.7
≥8B active params																	
DME-Large‡	–	75.4	82.9	79.6	95.2	81.1	90.2	75.7	64.5	62.2	74.4	89.1	61.5	93.8	75.1	83.4	80.2
Octen-VL-Embedding-Large‡	–	75.7	84.5	80.6	94.4	81.9	87.1	82.1	68.0	60.4	76.0	85.9	66.3	86.6	72.2	80.5	80.1
DME-Medium	9.4	74.5	80.9	78.2	94.5	79.8	87.7	71.0	61.0	58.5	70.8	87.6	57.8	94.5	73.5	82.0	78.4
Qwen3-VL-Embedding-8B	8.1	74.2	81.1	80.2	92.3	80.1	78.4	71.0	58.7	56.1	67.1	87.2	69.9	88.7	73.3	82.4	77.8
IFM-TTE-7B	8.3	76.7	78.5	74.6	89.3	77.9	60.5	67.9	51.7	54.9	59.2	85.2	71.5	92.8	67.3	81.8	74.8
WeMM-Embedding-8B	8.8	73.5	76.1	78.6	92.9	78.1	66.5	71.7	56.4	55.2	63.2	89.5	59.3	90.4	35.1	75.6	73.9
RzenEmbed-v2-7B	8.3	70.6	71.7	78.5	92.1	75.9	58.8	63.5	51.0	45.5	55.7	89.7	60.7	88.7	69.4	81.2	72.9
Ops-MM-Embedding-v1-7B	8.3	69.6	69.6	73.1	87.2	72.7	59.7	62.2	45.7	43.2	53.8	80.0	59.6	79.3	67.5	74.4	68.9
E5-Omni-7B	8.0	66.7	68.5	73.0	83.9	71.2	46.6	52.9	36.7	34.2	43.5	87.6	62.4	87.5	34.6	74.5	65.8
GME-Qwen2-VL-7B	8.3	57.6	34.6	71.2	59.5	55.9	37.3	50.3	28.3	37.5	38.4	89.6	55.5	85.0	69.4	79.4	59.1
Models adapted only with public MMEB-family data																	
∼2–4B active params																	
PLUME	2.1	66.5	59.2	67.6	79.7	66.3	45.0	52.3	33.5	46.7	44.1	72.1	49.8	78.1	57.4	67.5	61.6
UME-R1-2B	2.2	64.8	62.8	67.6	77.2	66.6	44.3	50.9	32.9	39.7	42.2	72.4	46.2	79.2	57.8	67.3	61.2
VLM2Vec V2-2B	2.2	62.9	56.4	69.6	77.1	64.9	39.2	34.7	28.4	37.5	34.7	74.4	44.6	79.3	62.5	68.7	59.1
BToks	2.2	64.3	59.8	68.8	77.4	66.0	43.7	47.0	33.0	33.6	39.9	71.1	38.6	81.3	38.1	62.7	59.0
VLM2Vec V1-2B	2.2	58.6	49.2	65.0	73.0	59.7	33.3	30.7	20.4	30.7	28.5	20.6	13.2	52.2	46.9	31.7	43.9
MoEMB-A3B	3.1	60.6	73.8	72.4	92.4	71.8	53.8	65.9	45.3	45.8	53.5	85.0	57.8	88.5	82.8	81.0	70.4
≥8B active params																	
Embed-RL-4B†	13.2	63.7	70.5	71.3	91.3	71.2	57.6	58.4	45.1	**49.5**	53.0	80.2	53.4	84.9	67.1	74.7	68.1
Embed-RL-2B†	10.9	62.8	67.9	68.6	90.4	69.2	57.0	55.9	45.1	49.4	52.1	79.9	52.0	84.6	65.7	74.1	66.8
UME-R1-7B	8.3	**67.1**	69.2	71.9	84.9	71.2	48.6	60.7	38.2	39.3	47.5	75.7	50.5	83.7	58.3	70.6	65.6
UniME-V2	8.0	65.6	68.7	73.1	90.8	71.8	37.2	50.6	28.9	39.6	39.0	61.8	42.0	70.5	58.3	60.1	60.6
VLM2Vec V1-7B	8.3	62.8	56.5	69.4	81.9	65.4	39.0	30.1	29.1	39.2	33.8	20.0	9.2	58.9	57.1	34.1	48.5
MoEMB-A10B	10.2	64.4	**75.4**	**74.7**	**92.7**	**74.0**	**58.1**	**67.6**	**47.3**	49.4	**56.3**	**85.5**	**62.7**	**89.1**	**82.9**	**82.2**	**72.4**

Ablations on MoE Design. We compare different MoE designs at equal active compute (Tab. 3). We can observe upcycling (App. E.1) does not outperform the dense backbone. We hypothesize this is due to that the limited scale for contrastive training over the MMEB V2 dataset is not able to train a strong router. Meanwhile, modality-wise MoE shows only marginal gains over the dense baseline. In contrast, fine-tuning a native pretrained sparse MoE backbone delivers substantial gains.

Table 3: Ablation on different expert scaling design.

Method	Act.	Image	Video	VisDoc	Overall
Qwen3.5-2B (dense)	2.2B	64.4	43.0	77.7	63.6
Dense-to-MoE upcycling	3.0B	62.6	45.3	77.3	63.1
Modality-wise MoE	2.2B	65.1	43.6	78.3	64.2
MoEMB-A3B (native)	3.1B	**71.8**	**53.5**	**81.0**	**70.4**

Ablations on Pooling and Routing Designs. Evaluating adaptation choices on the native sparse backbone (Tab. 4) reveals three insights: **(1) Pooling Strategy**: Replacing last-token pooling with a dedicated `<emb>` token provides notable performance gain. **(2) Router Updating**: Freezing the pretrained router performs identically to fine-tuning. **(3) Routing Objectives**: auxiliary load-balancing and z-losses degrade accuracy, while task priors and conflict-aware routing offer no meaningful gains. Together, (2) and (3) indicate that routing assignments learned during language pre-training remain highly effective for contrastive adaptation.

Table 4: Ablations on pooling designs and router adaptations for MoE-UME. Details for each router setting are provided in App. C.2.

Pooling	Routing objective	Router trained	Image	Video	VisDoc	Overall
last	✗	✓	70.7	51.6	81.1	69.50
last	LB + z	✓	65.6	53.7	77.8	66.62
last	conflict-aware	✓	70.4	53.2	81.3	69.80
<emb>	✗	✓	71.1	54.5	81.1	70.34
<emb>	task-ID prior	✓	69.9	53.5	81.0	69.55
<emb>	instruction prior	✓	70.1	54.1	80.3	69.55
<emb>	✗	✗	**71.8**	53.5	81.0	**70.38**

Ablations on Training Hyperparameters. Fig. 6 shows ablations on major training hyperparameters, where we can draw three observations: **(1)** the effective range is narrow for learning rate, where small decreases improve accuracy, but larger drops degrade it sharply due to underfitting; **(2)** performance peaks at two epochs, after which contrastive gains saturate. **(3)** logit scale is the most sensitive parameter, while batch size has the least impact.

Table 5: Reasoning-intensive retrieval on MRMR (Zhang et al., 2025a). †reported by the MRMR benchmark; other rows are our evaluations. **Bold**/underline mark best/second per column.

		Knowledge				Theorem				Contradiction			
Model	**Act. (B)**	Art	Med.	Sci.	Hum.	Math	Phy.	Eng.	Bus.	Neg.	Design	Traffic	**Avg**
≤4B active parameters													
BToks	2.2	72.0	48.2	62.2	62.3	22.5	36.1	29.1	49.0	14.5	46.4	36.0	43.5
UME-R1-2B	2.2	70.8	48.9	60.6	57.8	21.3	31.5	27.2	47.0	11.5	33.7	30.1	40.0
PLUME	2.1	69.9	47.7	63.6	56.7	18.6	30.3	27.7	44.5	9.5	43.2	27.5	39.9
VLM2Vec-V2	2.2	53.1	28.9	43.4	39.9	11.7	22.3	22.5	43.0	11.0	8.3	25.0	28.1
ColPali†	3.0	36.1	29.9	42.7	29.2	7.3	17.5	13.5	34.6	**28.5**	19.4	18.2	25.2
VISTA†	0.2	21.3	27.8	32.6	17.0	18.8	17.1	17.3	28.6	20.0	20.2	9.4	20.9
VLM2Vec†	4.0	53.5	22.4	36.7	24.0	2.1	2.8	2.8	2.9	11.5	5.6	18.3	18.1
MoEMB-A3B	3.1	**80.3**	**62.6**	**73.3**	**76.3**	28.2	35.3	34.3	**54.0**	11.0	63.6	**49.2**	**51.6**
≥7B active parameters													
LaME-7B	8.3	72.7	57.1	69.8	63.3	29.9	44.4	**38.0**	49.4	6.5	**66.9**	39.8	48.9
UME-R1-7B	8.3	79.2	58.2	71.8	66.1	28.2	40.2	32.7	50.3	6.0	57.6	39.5	48.2
Ops-MM-Embedding†	7.0	79.3	52.5	70.0	67.8	27.7	39.5	30.1	52.3	8.0	55.9	45.8	48.1
Embed-RL-4B	13.2	73.5	53.1	60.5	69.8	**32.9**	**45.6**	35.1	52.6	6.0	54.3	31.7	46.8
UniME-V2-7B	8.0	71.0	49.7	58.4	59.6	25.1	39.8	28.4	51.0	7.0	37.0	34.6	42.0
MM-Embed†	8.0	65.6	53.0	63.5	62.8	23.6	30.8	27.4	44.9	7.0	23.8	34.9	39.8
GME-Qwen2-VL†	7.0	54.3	40.1	46.8	45.6	28.8	36.0	30.2	45.1	15.0	26.3	29.6	36.2
E5-V†	8.0	25.1	11.7	16.6	10.8	2.1	3.4	2.5	5.2	11.5	3.7	2.1	8.6

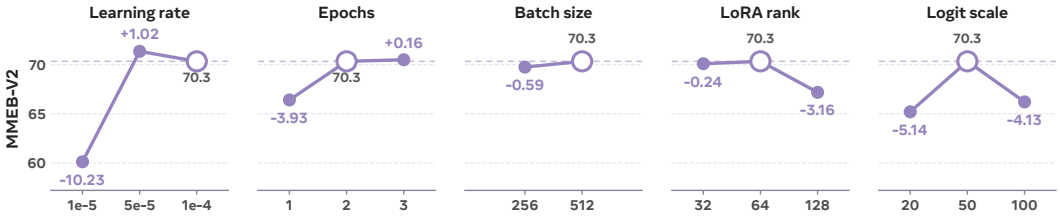

Figure 6: Hyperparameter ablations on the 35B backbone. Each panel varies one setting.

5.3 Results and Analysis for Adaptive Computation on MoE-UME

Adaptive computation yields strong accuracy–compute trade-offs for MoE-based UME. Overall, visual token pruning reduces execution cost notably and static expert pruning halves the expert pool, both at minimal accuracy loss (Tab. 6), confirming substantial token-level and parameter-level redundancy in MoE-UME.

Token pruning has the strongest end-to-end efficiency; expert-level actions give the weakest. We observe token pruning to be the most effective adaptive computation approach, converting significant

Table 6: Summarized results on adaptive-computation. We show a representative budget for each method.

Action	**Method**	**Budget**	**Image**	**Video**	**VisDoc**	**Overall**	**FLOPs↓%**	**QPS↑×**
—	No reduction	top-8 / keep all	71.0	54.5	81.1	70.3	—	1.00
Training-based methods								
Adaptive top-k	AdaMoE (App. F.1)	2.94 experts	68.2 (−2.8)	49.1 (−5.4)	79.3 (−1.8)	67.2 (−3.1)	15	1.07
Expert pruning	DiEP (Bai et al., 2025)	E=128	66.1 (−4.9)	47.9 (−6.6)	79.9 (−1.2)	66.2 (−4.1)	0.3	1.00
Expert skip	Learned skip gate	skip 50%	69.7 (−1.3)	53.4 (−1.1)	81.1 (+0.0)	69.4 (−0.9)	12	1.06
Token pruning	One-shot drop	drop 70%	69.4 (−1.6)	54.2 (−0.3)	80.5 (−0.6)	69.3 (−1.0)	**32**	**1.27**
Layer skip	Trained gate (App. F.1)	skip 14%	70.0 (−1.0)	52.6 (−1.9)	81.3 (+0.2)	69.5 (−0.8)	7.7	1.17
Early exit	Multi-exit	exit at 36/40	69.3 (−1.7)	52.8 (−1.7)	79.6 (−1.5)	68.7 (−1.6)	5.4	1.12
Inference-only methods								
Adaptive top-k	Uniform top-k	50% skip	70.5 (−0.5)	53.6 (−0.9)	80.9 (−0.2)	69.8 (−0.5)	11.6	1.06
	MoDES (DMT) (Huang et al., 2025b)	59% skip	70.4 (−0.6)	53.6 (−0.9)	80.9 (−0.2)	69.8 (−0.5)	13.8	1.04
	Random	50% skip	63.2 (−7.8)	46.4 (−8.1)	77.5 (−3.6)	63.7 (−6.6)	11.6	1.05
Expert pruning	REAP (Lasby et al., 2025)	E=128	70.3 (−0.7)	53.8 (−0.7)	80.1 (−1.0)	69.5 (−0.8)	0.3	1.00
	MSAN (Liu et al., 2026)		69.9 (−1.1)	54.0 (−0.5)	80.2 (−0.9)	69.4 (−0.9)		
	MAN (Liu et al., 2026)		70.1 (−0.9)	53.6 (−0.9)	80.3 (−0.8)	69.4 (−0.9)		
	Router-prob		68.8 (−2.2)	52.7 (−1.8)	80.4 (−0.7)	68.7 (−1.6)		
Token pruning	GDN-β	keep 50%	69.7 (−1.3)	53.5 (−1.0)	80.2 (−0.9)	69.2 (−1.1)	22.8	1.21
	Random		69.0 (−2.0)	52.8 (−1.7)	79.4 (−1.7)	68.4 (−1.9)		1.19

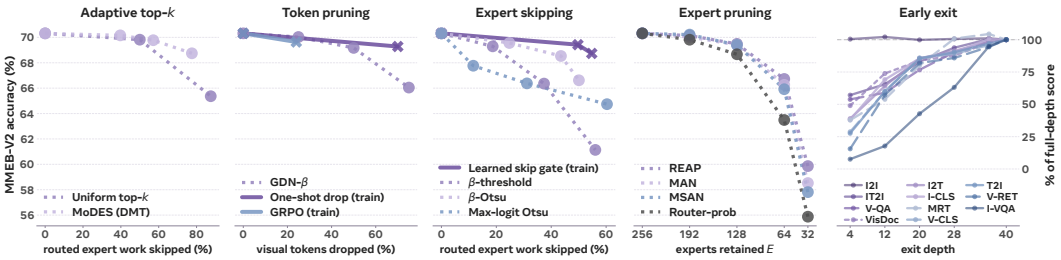

Figure 7: Compute-accuracy curves per method. More details are reported in Tab. 6 and App. G.

FLOP savings directly into throughput gains at minimal accuracy cost. By contrast, adaptive top-k methods yield minor wall-clock speedups.

Inference-only adaptive approaches are competitive with training-based methods. Inference-only MoDES outperforms trained adaptive top-k (Zeng et al., 2024), while our GDN-β-based pruning rivals trained token pruning gates. In general, we observe training-based method to perform slightly better mainly under extreme pruning regimes (Fig. 7).

Exit depths vary widely across tasks. Fig. 7 shows results for early exit. We can observe (1) Supervising intermediate exit depths has negligible impact on full-depth accuracy; (2) Exit depth depends heavily on task complexity: symmetric I2I retrieval plateaus after 4 of 40 layers, which confirms our hypothesis that the LLM layers can bring significant redundancy for simple retrievals. On the other hand, complex, text-guided retrievals require substantially deeper layers.

Our implementations effectively convert FLOP savings to wall-clock speedups. As shown in Fig. 8, packed varlen and SonicMoE convert 56–82% of FLOP reductions into runtime gains, far outperforming unmodified grouped GEMM (41%) and masking (0%). Baseline implementations fail to leverage adaptive top-k efficiently: grouped GEMM is tile-bound and saturates past $k \approx 4$, while the naive HuggingFace loop remains 5.7–8.0× slower overall despite high relative scaling.

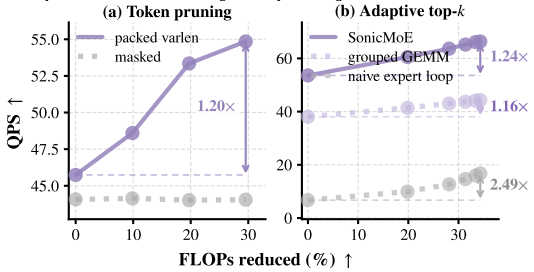

Figure 8: Transferring FLOPs saving into wall-clock speedup under baseline (grey) and our implementations.

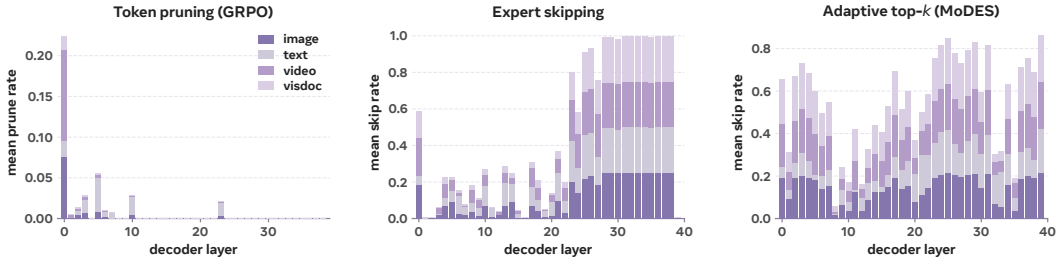

Figure 9: Depth-wise adaptive compute allocation across modalities.

Depth-wise adaptive compute allocation across modalities. Layer-wise allocation patterns (Fig. 9) reveal distinct modality behaviors: **Token pruning** acts early, immediately removing redundant image and video tokens while preserving text queries and document layouts for early contextualization. **Expert skipping** exhibits higher early-layer skip rates for text before unifying across modalities in deep layers and dropping to near-zero at the final layer for output embedding construction. Conversely, **adaptive top-k** allocates more experts to text in earlier layers.

6 Conclusion

We presented MoEMB, the first sparse MoE-based UME model. This approach gains over 10 accuracy points at near-constant active compute, establishing a new SOTA on MMEB-V2 and MRMR among public-data models. Furthermore, we provide the systematic study over adaptive computation methods for MoE-based embedding models, where we show that MoE embedders can reduce compute budget significantly with under a one-point accuracy loss.

Reproducibility Statement

The settings behind each reported number are documented in the paper. Backbone configurations, the training data mixture with its sampling weights, and the optimization settings for both MoEMB-A3B and MoEMB-A10B are given in §5.1 and App. C; App. C.2 formalizes the router adaptation variants compared in Tab. 4. For adaptive computation, the per-method training setups appear in App. F.1, and the inference-only procedures, with their calibration, in App. F.2. The efficiency claims depend on how compute is counted, so App. D states the FLOP accounting, the hardware and batching conditions under which throughput was measured, and the construction of Fig. 1. MMEB-V2 and MRMR are both public and we follow their official evaluation protocols; App. H lists our per-task scores so that any individual entry can be checked against them.

AI use statement

Anthropic Claude Code (Opus 4.8 and Opus 5) was used extensively for drafting code, aggregating results, and drafting manuscript. The authors verified experiment artifacts, numerical claims, and citations and take responsibility for the content.

References

Sikai Bai, Haoxi Li, Jie Zhang, Zicong Hong, and Song Guo. DiEP: Adaptive mixture-of-experts compression through differentiable expert pruning. *arXiv preprint arXiv:2509.16105*, 2025. URL https://arxiv.org/abs/2509.16105.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021. URL https://arxiv.org/abs/2107.05407.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. *arXiv preprint arXiv:2403.06764*, 2024. URL https://arxiv.org/abs/2403.06764.

Jianpeng Cheng, Xian Wu, Jiangfan Zhang, Wentao Bao, Chaitanya Ahuja, Shlok Kumar Mishra, Hanchao Yu, Yang Gao, Fan Xia, Qi Guo, Shaodan Zhai, Xiangjun Fan, and Jun Xiao. TTE-Flash: Accelerating reasoning-based multimodal representations via think-then-embed tokens. *arXiv preprint arXiv:2605.16638*, 2026. URL https://arxiv.org/abs/2605.16638.

Xuanming Cui, Jianpeng Cheng, Hong-you Chen, Satya Narayan Shukla, Abhijeet Awasthi, Xichen Pan, Chaitanya Ahuja, Shlok Kumar Mishra, Yonghuan Yang, Jun Xiao, Qi Guo, Ser-Nam Lim, Aashu Singh, and Xiangjun Fan. Think then embed: Generative context improves multimodal embedding. *arXiv preprint arXiv:2510.05014*, 2025. URL https://arxiv.org/abs/2510.05014.

Xuanming Cui, Hong-You Chen, Hao Yu, Hao Yuan, Zihao Wang, Shlok Kumar Mishra, Hanchao Yu, Yonghuan Yang, Jun Xiao, Ser-Nam Lim, Jianpeng Cheng, Qi Guo, and Xiangjun Fan. Reason to contrast: A cascaded multimodal retrieval framework. *arXiv preprint arXiv:2602.23369*, 2026. URL https://arxiv.org/abs/2602.23369.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2205.14135.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. ColPali: Efficient document retrieval with vision language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ogjBpZ8uSi.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. *arXiv preprint arXiv:2211.15841*, 2022. URL https://arxiv.org/abs/2211.15841.

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. *arXiv preprint arXiv:2510.13515*, 2025. URL https://arxiv.org/abs/2510.13515.

Wentao Guo, Mayank Mishra, Xinle Cheng, Ion Stoica, and Tri Dao. SonicMoE: Accelerating MoE with IO and tile-aware optimizations. *arXiv preprint arXiv:2512.14080*, 2025. URL https://arxiv.org/abs/2512.14080.

Chenwei He, Xiangzhao Hao, Tianyu Yang, Yuxiang Ma, Yuheng Jia, Lingxiang Wu, Chaoyang Zhao, Haiyun Guo, and Jinqiao Wang. PLUME: Latent reasoning based universal multimodal embedding. *arXiv preprint arXiv:2604.02073*, 2026.

Haohang Huang, Xuan Lu, Mingyi Su, Xuan Zhang, Ziyan Jiang, Ping Nie, Kai Zou, Tomas Pfister, Wenhu Chen, Wei Zhang, Xiaoyu Shen, and Rui Meng. MMEB-V3: Measuring the performance gaps of omni-modality embedding models. *arXiv preprint arXiv:2604.23321*, 2026. URL https://arxiv.org/abs/2604.23321.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder tasks need more experts: Dynamic routing in MoE models. *arXiv preprint arXiv:2403.07652*, 2024. URL https://arxiv.org/abs/2403.07652.

Wenxuan Huang, Zijie Zhai, Yunhang Shen, Shaosheng Cao, Fei Zhao, Xiangfeng Xu, Zheyu Ye, Yao Hu, and Shaohui Lin. Dynamic-LLaVA: Efficient multimodal large language models via dynamic vision-language context sparsification. *arXiv preprint arXiv:2412.00876*, 2025a. URL https://arxiv.org/abs/2412.00876.

Yushi Huang, Zining Wang, Zhihang Yuan, Yifu Ding, Ruihao Gong, Jinyang Guo, Xianglong Liu, and Jun Zhang. MoDES: Accelerating mixture-of-experts multimodal large language models via dynamic expert skipping. *arXiv preprint arXiv:2511.15690*, 2025b. URL https://arxiv.org/abs/2511.15690.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Haonan Jiang, Yuji Wang, Yongjie Zhu, Xin Lu, Wenyu Qin, Meng Wang, Pengfei Wan, and Yansong Tang. Embed-RL: Reinforcement learning for reasoning-driven multimodal embeddings. *arXiv preprint arXiv:2602.13823*, 2026. URL https://arxiv.org/abs/2602.13823.

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*, 2025. URL https://arxiv.org/abs/2410.05160.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. *arXiv preprint arXiv:2004.12832*, 2020. URL https://arxiv.org/abs/2004.12832.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30233–30249, 2022. URL https://arxiv.org/abs/2205.13147.

Zhibin Lan, Liqiang Niu, Fandong Meng, Jie Zhou, and Jinsong Su. UME-R1: Exploring reasoning-driven generative multimodal embeddings. *arXiv preprint arXiv:2511.00405*, 2025. URL https://arxiv.org/abs/2511.00405.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. REAP the experts: Why pruning prevails for one-shot MoE compression. *arXiv preprint arXiv:2510.13999*, 2025.

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. *arXiv preprint arXiv:2405.17428*, 2024. URL https://arxiv.org/abs/2405.17428.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, Xiaoqi Ren, et al. Gemini embedding: Generalizable embeddings from Gemini. *arXiv preprint arXiv:2503.07891*, 2025. URL https://arxiv.org/abs/2503.07891.

Dongxu Li, Yudong Liu, Haoning Wu, Yue Wang, Zhiqi Shen, Bowen Qu, Xinyao Niu, Fan Zhou, Chengen Huang, Yanpeng Li, et al. Aria: An open multimodal native mixture-of-experts model. *arXiv preprint arXiv:2410.05993*, 2025. URL https://arxiv.org/abs/2410.05993.

Jiamin Li, Qiang Su, Yitao Yang, Yimin Jiang, Cong Wang, and Hong Xu. Adaptive gating in mixture-of-experts based language models. *arXiv preprint arXiv:2310.07188*, 2023. URL https://arxiv.org/abs/2310.07188.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 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. URL https://arxiv.org/abs/2601.04720.

Xianming Li, Zongxi Li, Jing Li, Haoran Xie, and Qing Li. 2D matryoshka sentence embeddings. *arXiv preprint arXiv:2402.14776*, 2024. URL https://arxiv.org/abs/2402.14776.

Ziyue Li and Tianyi Zhou. Your mixture-of-experts LLM is secretly an embedding model for free. *arXiv preprint arXiv:2410.10814*, 2024. URL https://arxiv.org/abs/2410.10814.

Sheng-Chieh Lin, Chankyu Lee, Mohammad Shoeybi, Jimmy Lin, Bryan Catanzaro, and Wei Ping. MM-Embed: Universal multimodal retrieval with multimodal llms. *arXiv preprint arXiv:2411.02571*, 2025. URL https://arxiv.org/abs/2411.02571.

Dennis Liu, Zijie Yan, Xin Yao, Tong Liu, Vijay Korthikanti, Evan Wu, Shiqing Fan, Gao Deng, Hongxiao Bai, Jianbin Chang, Ashwath Aithal, Michael Andersch, Mohammad Shoeybi, Jiajie Yao, Chandler Zhou, David Wu, Xipeng Li, and June Yang. Moe parallel folding: Heterogeneous parallelism mappings for efficient large-scale moe model training with megatron core. *arXiv preprint arXiv:2504.14960*, 2025.

Zongfang Liu, Jinghui Zhang, Zijian Ma, Guangyi Chen, and Xin Yuan. How to score experts for one-shot MoE expert pruning: A unified formulation and selection principle. *arXiv preprint arXiv:2606.15716*, 2026.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*, 2024. URL https://arxiv.org/abs/2402.14800.

Yubo Ma, Jinsong Li, Yuhang Zang, Xiaobao Wu, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Jiaqi Wang, Yixin Cao, and Aixin Sun. Towards storage-efficient visual document retrieval: An empirical study on reducing patch-level embeddings. *arXiv preprint arXiv:2506.04997*, 2025. URL https://arxiv.org/abs/2506.04997.

Rui Meng, Ziyan Jiang, Ye Liu, Mingyi Su, Xinyi Yang, Yuepeng Fu, Can Qin, Zeyuan Chen, Ran Xu, Caiming Xiong, Yingbo Zhou, Wenhu Chen, and Semih Yavuz. VLM2Vec-V2: Advancing multimodal embedding for videos, images, and visual documents. *arXiv preprint arXiv:2507.04590*, 2025. URL https://arxiv.org/abs/2507.04590.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2023. URL https://arxiv.org/abs/2210.07316.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024. URL https://arxiv.org/abs/2409.02060.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

Qwen Team. Qwen3.5, 2026. URL https://qwen.ai/blog?id=qwen3.5. Official model cards for Qwen3.5-35B-A3B and Qwen3.5-122B-A10B.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024. URL https://arxiv.org/abs/2404.02258.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024. URL https://arxiv.org/abs/2403.15388.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. URL https://arxiv.org/abs/1701.06538.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo

Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Raghuveer Thirukovalluru, Rui Meng, Ye Liu, Karthikeyan K, Mingyi Su, Ping Nie, Semih Yavuz, Yingbo Zhou, Wenhu Chen, and Bhuwan Dhingra. Breaking the batch barrier (B3) of contrastive learning via smart batch mining. *arXiv preprint arXiv:2505.11293*, 2025. URL https://arxiv.org/abs/2505.11293.

Yebo Wu, Feng Liu, Ziwei Xie, Zhiyuan Liu, Changwang Zhang, Jun Wang, and Li Li. TSEmbed: Unlocking task scaling in universal multimodal embeddings. *arXiv preprint arXiv:2603.04772*, 2026. URL https://arxiv.org/abs/2603.04772.

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. URL https://arxiv.org/abs/2509.18095.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024. URL https://arxiv.org/abs/2412.06464.

Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10809–10818, 2022.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. AdaMoE: Token-adaptive routing with null experts for mixture-of-experts language models. *arXiv preprint arXiv:2406.13233*, 2024. URL https://arxiv.org/abs/2406.13233.

Longxiang Zhang, Weilong Dai, Guanghao Zhang, Hao Jiang, and Pipei Huang. Think when needed: Adaptive reasoning-driven multimodal embeddings with a dual-LoRA architecture. *arXiv preprint arXiv:2605.14448*, 2026. URL https://arxiv.org/abs/2605.14448.

Siyue Zhang, Yuan Gao, Xiao Zhou, Yilun Zhao, Tingyu Song, Arman Cohan, Anh Tuan Luu, and Chen Zhao. MRMR: A realistic and expert-level multidisciplinary benchmark for reasoning-intensive multimodal retrieval. *arXiv preprint arXiv:2510.09510*, 2025a. URL https://arxiv.org/abs/2510.09510.

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*, 2025b. URL https://arxiv.org/abs/2506.05176.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022. URL https://arxiv.org/abs/2202.08906.

APPENDIX

A Related Work **17**

B Limitations and future work **17**

C Implementation Details **17**
C.1 Model Architecture and Experimental Setup 17
C.2 Router Adaptation Objectives and Priors 18

D Measurement Protocol for FLOPs and Throughput **18**
D.1 Counting Canonical FLOPs 18
D.2 Accuracy–Throughput Figure Construction 19
D.3 Converting FLOP Savings into Wall-Clock Speedups 19

E Additional Studies on MoE Design **20**
E.1 Dense-to-MoE Upcycling 20
E.2 Capacity Ablation on Specialist and Generalist Models 21
E.3 Expert Routing Behavior Across Modality and Task 23

F Additional Details for Adaptive Computation **23**
F.1 Training-Based Methods 23
F.2 Inference-Based Methods 25

G Additional Analysis on GDN Write Strength β **26**

H Per-Task MMEB-V2 Results **31**

A Related Work

Universal Multimodal Embeddings. Recent advances in multimodal representation learning adapt vision-language models into general-purpose encoders evaluated on broad benchmark suites such as MMEB and MMEB-V2 (Jiang et al., 2025; Meng et al., 2025). Prior efforts to improve universal embedding performance primarily focus on data supervision, including hard-negative mining (Thirukovalluru et al., 2025; Lin et al., 2025; Gu et al., 2025) and multi-stage reranker distillation (Li et al., 2026; Cui et al., 2026). Another emerging direction leverages test-time reasoning, optimizing autoregressive thought generation prior to embedding emission (Lan et al., 2025; Jiang et al., 2026; Cheng et al., 2026; Zhang et al., 2026). In contrast to data-centric scaling or test-time generation overhead, we examine the complementary axis of encoder parameter capacity, evaluating how native sparse expert scaling improves contrastive representations.

Our work also differs fundamentally from existing MoE-based embedding approaches. TSEmbed (Wu et al., 2026) employs fixed task- or modality-wise LoRA routing, which caps expert capacity by the task taxonomy. We instead adapt native token-level sparse MoE backbones, allowing capacity to scale with expert count via dynamic, learned routing. Furthermore, unlike MoEE (Li & Zhou, 2024), which extracts off-the-shelf text embeddings directly from frozen router probabilities, we fine-tune sparse multimodal architectures to pool high-quality representation states.

Sparse Mixture-of-Experts Backbones. Sparsely gated MoE architectures scale model capacity while holding per-token compute constant (Shazeer et al., 2017; Fedus et al., 2022). Modern foundation architectures increasingly adopt fine-grained sparse routing to expand representation power, as demonstrated in Mixtral (Jiang et al., 2024), DeepSeek-V3 (DeepSeek-AI, 2024), OLMoE (Muennighoff et al., 2024), Kimi K3 (Team et al., 2026), Qwen3.5 (Qwen Team, 2026), and Aria (Li et al., 2025). However, these routing mechanisms are pre-trained exclusively under next-token prediction. We study how these pre-trained sparse routing distributions transfer when adapted contrastively to produce fixed-length vector representations.

Efficient Embeddings and Adaptive Computation. Adjustable-cost embedding methods have been studied via Matryoshka dimension truncation (Kusupati et al., 2022; Li et al., 2024), variable late-interaction vectors (Faysse et al., 2025; Xiao et al., 2025), and visual patch reduction (Ma et al., 2025). However, these works focus on reducing retrieval and storage cost, not the active computation for calculating the embedding. In parallel, adaptive computation frameworks reduce inference latency via dynamic depth skipping (Banino et al., 2021; Raposo et al., 2024), visual token pruning (Chen et al., 2024; Huang et al., 2025a), and adaptive top-k routing (Huang et al., 2025b), but these works mostly focus on visual classification and next token prediction (text generation) tasks, are mostly explored separately, and have not been thoroughly tested on embedding task. We unify these paradigms by systematically comparing token-, expert-, and layer-level execution controls within a single MoE multimodal embedder, identifying which mechanisms lead to best accuracy-efficiency tradeoff, and which best translate theoretical FLOP reductions into deployed wall-clock speedups.

B Limitations and Future Work

While MoEMB achieves SOTA performance over models trained with public MMEB-V2 training data, it's currently lagging behind models trained with external data. Also, it currently does not support audio as input modality. Our future work include training MoEMB under larger scale of data, and to include more input modalities (*e.g.* audio) and more diverse tasks.

C Implementation Details

C.1 Model Architecture and Experimental Setup

Backbone Configurations. We evaluate two primary variants built upon native sparse MLLM backbones: MoEMB-A3B (3.1B active parameters) and MoEMB-A10B (10.2B active parameters). Both variants employ $E = 256$ routed experts per layer with top-8 routing ($k = 8$) and a single shared expert. MoEMB-A3B uses a hidden dimension of $d = 2048$, 40 decoder layers (30 Gated-DeltaNet and 10 full-attention blocks), and an expert intermediate dimension of $d_{\text{ffn}} = 512$. MoEMB-A10B expands model capacity with $d = 3072$, 48 decoder layers, and $d_{\text{ffn}} = 1024$.

Training Data. Models are fine-tuned on a multi-task mixture comprising 24 sampling streams across image, video, and visual-document retrieval. This includes 17 image-text tasks from MMEB (Jiang

et al., 2025), 4 visual-document streams from ViDoRe and VisRAG, and 3 video retrieval streams derived from LLaVA-Hound. Following our standard protocol, we exclude generated chain-of-thought sequences from training targets. Complete dataset sampling weights are detailed in Tab. 7.

Training setups. We train for two epochs using BF16 precision and the AdamW optimizer ($\beta_1 = 0.9, \beta_2 = 0.98$, weight decay $= 0.01$). The learning rate warms up to 10^{-4} over 10 steps, followed by a cosine decay to 10^{-5}. Training uses a global batch size of 512 across 64 H200 GPUs with a maximum sequence length of 8,192 tokens. To enable parameter-efficient adaptation, we apply LoRA ($r = 64, \alpha = 128$) to the vision encoder, attention projections, and expert feed-forward layers, yielding 73.6M trainable parameters for MoEMB-A3B and 196.6M for MoEMB-A10B.

Table 7: **Training subsets.** Weights are relative and do not sum to one.

Subset	Weight	Subset	Weight	Subset	Weight
A-OKVQA	0.26	ChartQA	0.35	CIRR	0.43
DocVQA	0.84	ImageNet-1K	2.25	InfographicsVQA	0.31
MSCOCO i2t	2.58	MSCOCO image pairs	3.78	MSCOCO t2i	2.32
NIGHTS	0.23	OK-VQA	0.25	SUN397	0.22
VisDial	3.75	Visual7W	1.68	VisualNews i2t	2.91
VisualNews t2i	3.49	WebQA	0.23	ViDoRe subset 1	5.00
ViDoRe subset 2	5.00	VisRAG subset 1	6.00	VisRAG subset 2	6.00
Video caption	5.27	Video QA	4.38	Video retrieval	5.76

C.2 Router Adaptation Objectives and Priors

We detail the formulations for the router adaptation variants evaluated in Tab. 4.

Load-Balancing and z-Loss. To evaluate whether standard pre-training regularizers aid contrastive adaptation, we include a joint auxiliary loss $\mathcal{L}_{\text{aux}} = \lambda_{\text{lb}}\mathcal{L}_{\text{lb}} + \lambda_z\mathcal{L}_z$. The load-balancing objective $\mathcal{L}_{\text{lb}} = E\sum_{j=1}^{E} f_j P_j$ encourages uniform expert allocation (Fedus et al., 2022), where f_j is the fraction of tokens routed to expert j and P_j is the mean router probability assigned to expert j. The router z-loss $\mathcal{L}_z = \frac{1}{B}\sum_{b=1}^{B}(\log\sum_{j=1}^{E} e^{g_{bj}})^2$ penalizes large pre-softmax router logits g to stabilize routing dynamics (Zoph et al., 2022). We set weighting coefficients to $\lambda_{\text{lb}} = 0.01$ and $\lambda_z = 0.001$.

Conflict-Aware Routing. To mitigate multi-task expert interference, this objective forces task pairs with opposing router gradients onto distinct experts. For task t at layer ℓ, we maintain exponential moving averages of its expert assignment distribution $\bar{\pi}_t^{(\ell)} \in \mathbb{R}^E$ and router gradient $g_t^{(\ell)}$. The gradient conflict between current task s and candidate task t is defined as $C_{st}^{(\ell)} = \text{clamp}(-\cos(g_s^{(\ell)}, g_t^{(\ell)}), 0, 1)$. The objective maximizes the conflict-weighted Jensen–Shannon (JS) divergence between routing distributions across layers:

$$\mathcal{L}_{\text{conf}} = -\frac{1}{|\mathcal{P}|}\sum_{t\in\mathcal{P}}\frac{1}{L}\sum_{\ell=1}^{L} C_{st}^{(\ell)}\ \text{JS}\left(\bar{\pi}_s^{(\ell)} \parallel \bar{\pi}_t^{(\ell)}\right), \tag{5}$$

where $\mathcal{P}$ is the set of tasks with non-zero conflict relative to task s, and $\text{JS}(p \parallel q) = \frac{1}{2}\,\text{KL}(p \parallel m) + \frac{1}{2}\,\text{KL}(q \parallel m)$ for mixture distribution $m = \frac{1}{2}(p+q)$. This pushes conflicting task pairs toward disjoint expert subsets while exerting no penalty on cooperative or orthogonal pairs.

Task-ID and Instruction Priors. We test conditioning router assignments on task information by injecting an additive bias vector $b^{(\ell)} \in \mathbb{R}^E$ into the router logits: $g^{(\ell)} = W_r^{(\ell)}h + b^{(\ell)}$. We test two setups: *(1) task-ID prior*: $b^{(\ell)}$ is obtained from a learned task embedding table with hidden size 256 and dropout 0.3. *(2) instruction prior*, $b^{(\ell)}$ is projected from the mean pooling of instruction tokens at layer ℓ, updating the routing bias dynamically across network depth.

D Measurement Protocol for FLOPs and Throughput

D.1 Counting Canonical FLOPs

Logical FLOP counts are computed using the same query/target (whichever side contains multimodal input) subset per task across all 78 MMEB-V2 tasks. As the regular FlopsCounter does not capture

fused kernels such as FlashAttention (Dao, 2024) and linear attention (Yang & Zhang, 2024), their FLOPS are added analytically. Grouped expert GEMMs are computed based on physically executed row volumes to prevent double-counting pruned expert allocations.

D.2 Accuracy–Throughput Figure Construction

In this section we provide details for constructing Fig. 1.

MoEMB Configurations. The expert-scaling curve for MoEMB-A3B evaluates the performance with different number of total experts $E \in \{32, 64, 128, 192, 256\}$, constructed by pruning the full $E = 256$ model using REAP (Lasby et al., 2025) under the reconstruction-error criterion (Eq. 4). MoEMB-A10B represents the full $E = 256$ backbone evaluated under the identical protocol. The single adaptive compute marker corresponds to the GRPO-trained policy (App. F.1).

Baseline Models. Autoregressive think-then-embed systems, such as Embed-RL (Jiang et al., 2026) and UME-R1 (Lan et al., 2025), are evaluated with reasoning generation both enabled and disabled to highlight their accuracy-throughput tradeoff via reasoning. PLUME (He et al., 2026) scales test-time compute across latent reasoning steps $K \in \{4, 6, 8\}$, under which throughput drops from 4.5 to 3.2 queries/sec. VLM2Vec-V2 (Meng et al., 2025) serve as single-vector dense references.

D.3 Converting FLOP Savings into Wall-Clock Speedups

This section details the empirical latency measurements supporting Fig. 8.

A reduction in logical computation is not a reduction in QPS, and the gap decides which actions are worth having. However, a large body of adaptive computation works focus the former and ignore the latter (Zeng et al., 2024; Huang et al., 2024). In this work, we address this gap by providing implementation for effectively converting the computation saving into actual batched inference acceleration. We then conduct detailed benchmarking for both computation and wall-clock savings.

For token pruning-based actions we adopt a *packed variable-length* approach: at each layer the surviving tokens are gathered into contiguous segments with updated cumulative sequence lengths, and a segment index keeps the attention operations from mixing adjacent samples. This procedure allows for batched adaptive computation where token numbers vary across layers, which is a nontrivial implementation compared to one-shot pruning (Huang et al., 2025a; Shang et al., 2024).

For MoE-specific methods, many report efficiency without an inference wall-clock measurement (Zeng et al., 2024; Huang et al., 2024; Li et al., 2023), do not supply the implementation that would realize the saving (Huang et al., 2025b), or rely on a naive implementation that loops over experts (Lu et al., 2024), which we find to be 5.7–8.0× slower than a fused kernel, so any speedup measured against it is overstated. In this work, we run the wall-clock benchmarks through tile-aware grouped-GEMM kernels (Guo et al., 2025) alongside the batched grouped GEMM (Gale et al., 2022), which exposes where per-token expert sparsity stops gaining wall-clock speedup from FLOPS saving.

Sequential Expert Looping. Sequential expert execution used in some existing MoE-efficiency methods (Lu et al., 2024) incurs heavy kernel launch overhead, running 5.7–8.0× slower than fused grouped GEMMs (1189.5 ms vs. 210.2 ms for grouped GEMM at $B = 8, S = 1024$; Tab. 8). Although sequential loops exhibit high sensitivity to expert sparsity (reducing latency by 32.3–59.9%), this relative gain stems from high baseline launch overheads rather than efficient execution.

Table 8: Different throughput and FLOPS corresponding to different expert skip rate, for different MoE kernels. Throughput is queries per second on one H200.

Policy			Throughput (QPS)		
Mean k	Skip (%)	FLOPs ↓%	Naive loop	Grouped GEMM	SonicMoE
8.00	0.0	0.0	6.7	38.1	53.6
4.02	49.8	19.8	9.9	41.4	60.5
2.35	70.6	28.1	12.7	43.0	63.7
1.70	78.8	31.4	14.7	43.7	65.2
1.30	83.8	33.4	16.0	44.1	66.1
1.10	86.2	34.3	16.8	44.3	66.4

Tile-Bound Constraints in Grouped GEMMs. Grouped GEMM kernels (Gale et al., 2022) process each expert's tokens in tiles of at least $M = 128$ rows, and a partly filled tile costs mostly the same

time as a full one. An expert holding n_j routed rows therefore requires $\lceil n_j/M \rceil$ tiles, and skipping tokens does not produce actual wall-clock saving until it empties a whole tile. Whether it does depends on how many rows an expert holds to begin with. Under top-8 routing a batch of T tokens produces $8T$ routed token–expert pairs, which spread over $E = 256$ experts give $n_j \approx 8T/E = T/32$ rows apiece when the router is balanced, so the count grows with sequence length.

At $T = 4{,}096$ tokens that is roughly 128 rows per expert. A 52.9% skip policy cuts that to about 50, but both amounts occupy a single tile, so the kernel executes the work it would have executed anyway while the model still spends time on the additional skip gates, which results in a $0.91\times$ slowdown. At $T = 8{,}192$ each expert instead holds around 256 rows, which is two tiles. There the same kind of policy at 69.6% skip cuts the count to about 78 rows, one tile ($\lceil 256/128 \rceil = 2 \to \lceil 78/128 \rceil = 1$), and removing that second tile is a genuine saving of $1.11\times$. This suggests that the wall-clock speedup for adaptive top-k or expert skipping also depends on the sequence length or the batch size.

Backend Customization and Speedup Saturation. Hardware speedups are primarily driven by backend kernel design (Fig. 8b). Replacing standard grouped GEMM with SonicMoE (Guo et al., 2025) kernel yields an $1.40\times$ speedup at fixed $k = 8$ (210.2 ms vs. 149.2 ms at $S = 8192$). Within SonicMoE, lowering active expert width from $k = 8$ to $k \approx 4$ provides an additional 1.11–$1.24\times$ speedup. Further reduction to $k \approx 1.1$ yields diminishing returns ($1.17\times$ total for grouped GEMM, $1.74\times$ for SonicMoE; Tab. 8) as execution becomes dominated by non-routed operations, including self-attention, shared experts, and token dispatch.

Physical Sequence Compaction vs. Spatial Masking. Token pruning requires physical sequence compaction to achieve wall-clock latency reductions. Spatial masking preserves activation tensor shapes ($[B, S, H]$), resulting in a flat $1.02\times$ execution speedup across all pruning budgets (Fig. 8a). In contrast, physically packing remaining sequence tokens post-pruning via FlashAttention-2 varlen function (Dao, 2024) shortens activation tensors across downstream layers, yielding speedups of $1.088\times$, $1.194\times$, and $1.228\times$ at 75%, 50%, and 25% visual token keep ratios (Tab. 9).

Table 9: Token pruning under masking and physical compaction.

	Masked		Packed varlen		
Keep	QPS	Speedup	FLOPs ↓%	QPS	Speedup
1.00	45.7	1.061	0.0	45.7	1.024
0.75	44.1	1.025	9.9	48.6	1.088
0.50	44.0	1.023	19.7	53.4	1.194
0.25	44.0	1.023	29.6	54.9	1.228

E Additional Studies on MoE Design

This section expands the design and capacity ablations of §5.2.

E.1 Dense-to-MoE Upcycling

To evaluate whether expert capacity can be synthesized during downstream fine-tuning, we upcycle a dense 2B multimodal encoder into a sparse MoE architecture by varying structural granularities and router initialization strategies.

Structural Granularity. A virtual group count G determines how the dense feed-forward network (FFN) is partitioned. For coarse upcycling ($G = 1$), each expert is a full replica of the original FFN ($d_{\text{expert}} = d_{\text{ffn}}$), where top-1 routing reproduces the dense baseline's FLOP footprint. For fine-grained upcycling ($G > 1$), the FFN is sliced along its intermediate dimension into G narrower experts ($d_{\text{expert}} = d_{\text{ffn}}/G$), requiring $k = G$ active experts per token to match the capacity of the dense layer.

Router Initialization. Because the router gate is newly introduced, its initialization controls initial expert activation patterns. We test three initialization methods: (1) *random* initialization ($\mathcal{N}(0, 0.02)$) breaks symmetry between identical experts; (2) *tiled* initialization replicates the dense gate row across all experts; and (3) *zero+bias* sets initial pre-softmax logits to zero with a positive bias, ensuring every FFN slice is selected from the first step so the layer matches dense output prior to adaptation.

Results. As shown in Tab. 10, upcycling fails to improve representation quality over the dense baseline under same active compute. Coarse top-1 upcycling ($E = 4, G = 1$) performs comparably to the dense reference (62.44 vs. 62.41), while increasing expert count ($E = 32$) causes performance to drop to 60.26. Fine-grained partitioning ($G = 4$) similarly reduces overall accuracy below 60.0. Outperforming the dense baseline requires doubling the active compute per token ($2\times$, top-2 routing),

Table 10: **Dense-to-MoE upcycling on the 2B encoder.** Image, Video and VisDoc are the 36, 18 and 24 task averages; Overall is over all 78.

E	G	k	Init	Act.	Image	Video	VisDoc	Overall
Dense reference					62.5	42.6	77.2	62.41
4	1	2	tiled	2×	62.6	45.3	77.3	**63.12**
4	1	1	random	1×	63.8	40.0	77.2	62.44
32	1	1	random	1×	61.0	38.8	75.2	60.26
32	4	4	zero+bias	1×	60.3	40.0	73.4	59.64
8	4	4	zero+bias	1×	59.5	40.4	73.8	59.50
8	4	4	tiled	1×	59.9	34.9	72.1	57.88

reaching 63.12. These results confirm that synthesizing expert capacity during contrastive fine-tuning is ineffective, highlighting the necessity of native MoE backbones pre-trained at scale.

E.2 Capacity Ablation on Specialist and Generalist Models

Experimental Protocol. To evaluate how model capacity impacts multi-task adaptation, we evaluate multi-task generalists against task-group specialists across three backbones: Qwen3.5-2B (dense, 2B active), Qwen3.5-9B (dense, 9B active), and Qwen3.5-35B-A3B (MoE, 3.1B active). Specialist models are fine-tuned independently on individual task groups (Image Classification and Image Retrieval), whereas generalists are fine-tuned on the full dataset mixture. We measure multi-task capacity degradation using the generalist-specialist performance gap. All configurations share identical training hyperparameters (global batch size 256, learning rate 5×10^{-5}, LoRA rank 64, $\alpha = 128$, <emb> pooling, 1 epoch per setting).

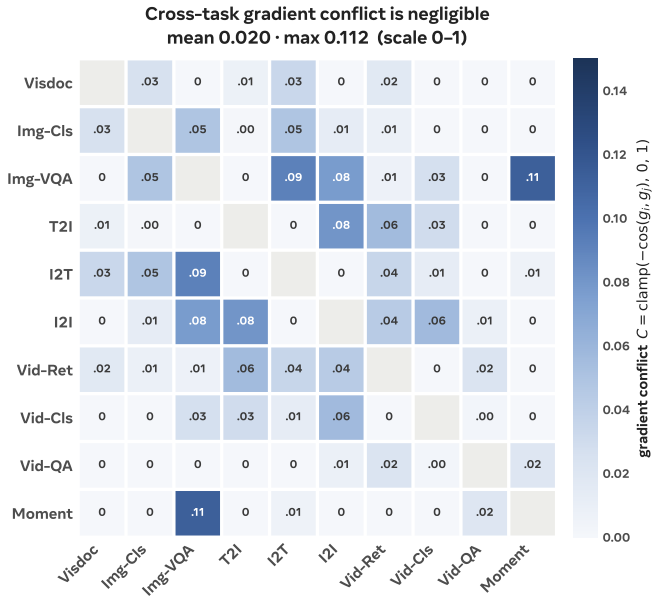

Figure 10: Pairwise conflict $C = \mathrm{clamp}(-\cos(g_i, g_j), 0, 1)$ on the <emb> gradient.

Minimal Cross-Task Gradient Conflict. A primary hypothesis for multi-task performance degradation is destructive gradient interference across task streams. To test this, we compute the pairwise cosine conflict $C = \mathrm{clamp}(-\cos(g_i, g_j), 0, 1)$ across the ten MMEB-V2 categories, where g is the gradient of $\mathcal{L}_{\mathrm{emb}}$ (Eq. 1) with respect to the pooled <emb> embeddings q and t, averaged within a category. As shown in Fig. 10, the conflict is negligible (mean 0.020, max 0.112). The router-parameter gradients that $\mathcal{L}_{\mathrm{conf}}$ acts on (App. C.2) conflict even less at the same checkpoint: mean 0.003 and max 0.035 over the nine training task types. Because categories do not pull the <emb> embedding

in opposing directions, generalist degradation stems primarily from total capacity constraints rather than inter-task gradient competition. Expanding expert capacity directly addresses this limitation 3.

Table 11: **Specialist vs. generalist by backbone.** Group scores are unweighted averages over the group's tasks; **Mean** is the flat mean of the two group scores and is what Fig. 3 plots.

Backbone	**Arm**	**Image-CLS** (10 tasks)	**Image-Ret.** (16 tasks)	**Mean**
Qwen3.5-2B (dense, 2B act.)	specialist	60.69	73.06	66.88
	generalist	56.53	71.17	63.85
	Δ	-4.16	-1.89	-3.02
Qwen3.5-9B (dense, 9B act.)	specialist	63.50	77.49	70.50
	generalist	61.34	75.81	68.58
	Δ	-2.16	-1.68	-1.92
Qwen3.5-35B-A3B (MoE, 3.1B act.)	specialist	59.73	74.04	66.89
	generalist	58.97	72.54	65.75
	Δ	$\mathbf{-0.76}$	$\mathbf{-1.50}$	$\mathbf{-1.13}$

E.3 Expert Routing Behavior Across Modality and Task

To analyze expert routing patterns across modalities and task domains, we profile top-8 routing assignments across all 40 MoE layers using 10 MMEB-V2 categories covering all modalities and tasks, with 16 samples per category. The routing divergences are measured using Jensen–Shannon (JS) distance over the 256-expert distribution.

Modality Preference Over Task. Grouping subsets by input type (photo, document, video, text) shows that expert routing depends far more on input modality than task taxonomy (Fig. 11a). Photos and videos draw on heavily overlapping expert populations, whereas document pages and text branch into distinct pathways. Notably, this modality specialization emerges naturally during contrastive adaptation without needing auxiliary load-balancing or z-loss constraints.**Depth-Wise Modality Convergence.** Expert specialization varies strongly across model depth (Fig. 11). Early decoder layers exhibit high modality divergence, with most active experts dedicated primarily to either vision or text. In deeper layers, this separation steadily fades as representations across all modalities converge toward a compact set of shared experts, before briefly re-diverging at the final layer to form the output embedding.

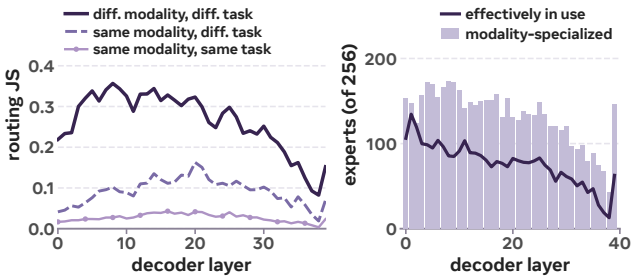

Figure 11: **Expert routing separates input types, and the separation fades with depth.** (a) Per-layer JS distance between two of the 20 subsets, each summarized by its 256-expert histogram, averaged over three disjoint sets of pairs. Crossing a task boundary at fixed input type stays near the same-task curve at every layer, while crossing an input type runs about three times higher. (b) How many of the 256 experts have specialized toward a single modality ($|\pi_j| > 0.5$), and how many the layer keeps occupied ($\exp(H)$ of the routing entropy).

F Additional Details for Adaptive Computation

This section gives implementation details for every adaptive-computation method of §4.1 and Tab. 6, grouped by whether the decision is learned during training or taken at inference.

F.1 Training-Based Methods

All training-based variants extend the contrastive framework described in Appendix C. Models are trained on 64 NVIDIA H200 GPUs (expert parallel size 64) in BF16 precision using AdamW (weight decay $= 0.01$, gradient clip norm 1.0, logit scale 50, maximum sequence length 8,192). LoRA adapters ($r = 64, \alpha = 128$, dropout 0.05) are applied to visual and language projection layers (`linear_qkv`, `linear_proj`, `linear_fc1`, `linear_fc2`). Global batch size is set to 512, except for GRPO, which uses a global batch size of 256 with 8 rollouts per input.

Initialization and Distillation Regimes. Token pruning, layer skipping, AdaMoE, and GRPO are initialized from the pretrained MoEMB-A3B checkpoint and fine-tuned for one epoch. Token pruning and layer skipping train with a learning rate of 10^{-5} and an L_2 distillation loss (weight $= 1.0$) against cached full-compute teacher hidden states to stabilize representations during gate optimization. AdaMoE is fine-tuned at a learning rate of 10^{-4} without distillation. GRPO uses a learning rate of 5×10^{-5} and integrates distillation directly into its reward function. Expert skipping is initialized directly from the base backbone and trained jointly with contrastive adaptation over two epochs at a learning rate of 10^{-4} without distillation.

Token Pruning. Pruning decisions are evaluated after decoder block 2, targeting a 30% visual token retention rate via an L_1 budget loss (weight $= 0.5$, 15% warmup). The Gumbel-Softmax temperature is linearly annealed from 4.0 to 0.5 over 2,000 steps, with predictor weights updated at $40\times$ the backbone learning rate. Pruning is restricted exclusively to visual tokens.

Expert Skipping. Expert skipping penalizes active compute via an L_1 penalty weight of 0.02 on $(1 - \rho_{\text{skip}})$. This penalty is held at zero for the first 15% of training steps and linearly ramped to full weight over the subsequent 5% of steps.

Layer Skipping. A gate module at each block evaluates the hidden state at the <emb> position to produce a binary keep-or-skip decision for the entire block. When a block is skipped, its residual passes through unmodified, bypassing both attention and MoE sub-layers. The gate is trained via a straight-through estimator targeting a 20% skip rate under unit loss weight, applied from decoder block 3 onward. In practice, the policy converges to a 20.5% empirical skip rate concentrated in 9 decoder blocks, with the remaining eligible blocks left unskipped.

Adaptive Top-k. Following AdaMoE (Zeng et al., 2024), we augment the router with 256 null experts alongside the 256 physical experts and perform a joint top-8 selection across all 512 candidates. A token's effective width corresponds to the number of top-8 slots allocated to physical experts, eliminating manual threshold tuning. A load-balancing penalty (weight $= 0.02$) maintains the mean activation near four physical experts during the first half of training, after which it drops to 10^{-4} to allow unconstrained routing convergence. The policy settles at an average of 2.94 physical experts per token at evaluation (63.3% reduction in routed expert execution); we compare this setting against MoDES at an equivalent target budget in §5.3.

Group Relative Policy Optimization (GRPO). Differentiable training (*e.g.* STE) is unstable (App. F.1). We instead treat each gate's decision as a policy and adapt GRPO (Shao et al., 2024), which samples the discrete action in place of differentiating through it. Each efficiency gate (*e.g.* an expert skipping gate) predicts a distribution over its actions. For each input, we sample a group of outputs $\{a_i\}_{i=1}^{G}$, where each rollout samples one action from every gate, producing an embedding e_{a_i}. As adaptive computation has no pretrained reference policy to regularize toward, we replace the KL term with a distillation loss toward a teacher embedding e^{full}, computed by a full-compute greedy forward pass. To avoid reward hacking, we gate the efficiency reward $S(a_i)$ on the success of the in-batch retrieval $\text{recall@}G(a_i) \in [0, 1]$:

$$\mathcal{L} = -\,\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\min\Big(r_i\hat{A}_i,\ \text{clip}\big(r_i, 1{-}\epsilon, 1{+}\epsilon\big)\hat{A}_i\Big)\right] + \lambda_{\text{d}}\,\mathbb{E}\big[1 - \cos(e_a, e^{\text{full}})\big],$$

$$r_i = \frac{\pi_\theta(a_i)}{\pi_{\theta_{\text{old}}}(a_i)},\quad \hat{A}_i = \frac{R(a_i) - R_\mu}{R_\sigma},\quad R(a_i) = \underbrace{\text{recall@}G(a_i)}_{\text{recall gate}}\big(\underbrace{\bar{s}^{+}(a_i) - \bar{s}^{-}(a_i)}_{\text{ranking margin}} + \lambda_{\text{s}} S(a_i)\big),$$

where R_μ, R_σ are the mean and standard deviation of the reward over the group. $\bar{s}^{+}(a_i)$ and $\bar{s}^{-}(a_i)$ denote the similarity between positive and negative pairs.

GRPO optimizes a lightweight three-way classification head per MoE layer that emits categorical decisions over $\{\text{ROUTE}, \text{SKIP}, \text{DROP}\}$ for each token. Native expert routing remains greedy, meaning the policy dictates only execution state without altering expert selection logic. Auxiliary routing losses are disabled.

The policy is trained on 8 rollouts per sequence sampled at temperature 1.0 using a learning rate of 5×10^{-5} without KL penalties or value baselines, utilizing group-standardized advantages across rollouts. Log-probabilities sum categorical policy decisions across active tokens and MoE layers, normalized by the total decision count to equalize variable sequence lengths. To manage memory overhead, gradients are replayed one rollout at a time. Updates are computed on-policy ($r^{(i)} \equiv 1$, rendering ratio clipping inactive). To avoid distorting core feature spaces, policy gradients flow exclusively to the efficiency gate heads; base backbone and LoRA weights remain fixed at their contrastive values (Lan et al., 2025; Jiang et al., 2026).

The overall reward combines an efficiency score $S(a)$ and a cosine distillation reward against teacher embeddings (weight $= 1.0$, softmax temperature 0.2 for negative pairs). The efficiency reward credits each action by the compute it removes at each layer:

$$S(a) = 1 - \frac{\mathbb{E}[\text{FLOPs}(a)]}{\text{FLOPs}_{\text{dense}}}, \tag{6}$$

$$\mathbb{E}[\text{FLOPs}(a)] = \sum_{\ell}\sum_{t}\alpha_{t,\ell}\left[f_\ell^{\text{attn}} + f_{\text{sh}} + (1 - p_{t,\ell}^{\text{skip}} - p_{t,\ell}^{\text{drop}})f_{\text{rt}}\right], \tag{7}$$

where $\alpha_{t,\ell}$ is the probability that token t reaches layer ℓ, and f_ℓ^{attn}, f_{sh}, f_{rt} represent layer-specific attention, shared expert, and routed expert FLOP costs. Expert skipping removes f_{rt} at that layer and token dropping removes all downstream per-token compute.

Explored Variants and Negative Results. We additionally evaluated A-ViT and PonderNet-style dynamic halting mechanisms (Yin et al., 2022; Banino et al., 2021) based on learned hidden states and GDN-β token scores. Training combined a ponder penalty and a geometric distribution prior over steps under a joint Gumbel-Softmax / STE loss formulation (Jang et al., 2017):

$$\mathcal{L}_{\text{halt}} = \mathcal{L}_{\text{emb}} + \lambda_{\text{p}}\mathbb{E}[N_{\text{halt}}] + \lambda_{\text{kl}}\text{KL}\left(q_{\text{halt}} \,\|\, \text{Geom}(\gamma)\right).$$

Table 12 summarizes the observed failure modes. Aggressive regularization degraded target representation quality, whereas conservative settings caused gate gradients to saturate near zero reduction. Joint pruning techniques such as DiEP (Bai et al., 2025) similarly failed to achieve competitive accuracy-efficiency trade-offs. We hypothesize that credit assignment through non-differentiable or delayed attention paths introduces high gradient variance during early training, causing efficiency penalties to dominate before representation alignment can stabilize.

Table 12: Summary of non-convergent or suboptimal baseline implementations.

Variant	Observed behavior
Projected A-ViT gate	Training is initially healthy, then drop exceeds roughly 85%; the embedding token loses visual context and both representation and gradients collapse.
Cumulative drop logit	The compute reward drives nearly all halting to the first layer; the distribution prior does not offset the incentive to drop earliest.
GDN-β halting	With a stable negative sign (low β drops first), the gate frequently becomes inert and returns toward zero drop; more aggressive settings destabilize training.
Learned exit-halting head	Exit probabilities converge to the final layer, producing negligible savings under stable quality.

F.2 Inference-Based Methods

This subsection details the training-free dynamic efficiency heuristics evaluated in §4.2.

Token Pruning via GDN-β. Visual tokens are ranked using the Gated-DeltaNet write strength β evaluated at decoder block 4. The lowest-scoring tokens are pruned, and remaining sequences are gathered to reduce compute across all subsequent blocks. Text tokens are fully preserved.

Expert Skipping via GDN-β. Using the same β signal, tokens ranking in the lower fraction per MoE block skip routed experts.

Static Expert Pruning. Methods including REAP (Lasby et al., 2025), MAN (Liu et al., 2026), MSAN (Liu et al., 2026), and router logit thresholding evaluate expert importance scores over a static calibration set of 16 balanced samples. The lowest-ranked experts are permanently removed across all layers. Because routing maintains top-8 selection over reduced expert pools, this strategy reduces total parameters without reducing dynamic FLOPs per token.

Adaptive Top-k Routing. Uniform top-k keeps the k highest-scoring of the native eight experts at every layer, with no calibration and no learned gate. The pretrained router's ranking is what makes the reduced budget cheap: at $k = 4$ accuracy falls by 0.5 points, while drawing four of the eight at random costs 6.6 at the identical 11.6% FLOPs reduction (Tab. 6). The ordering the router induces over its top-8 therefore accounts for most of what a halved budget preserves.

MoDES Calibration. MoDES derives layer-wise scaling parameters s_ℓ and a global activation threshold τ using a calibration subset from the ImageNet-1K evaluation split.

Adaptive Otsu Thresholding for Expert Skipping. Rather than enforcing uniform skip ratios across layers, we explore dynamically computing per-layer thresholds using Otsu's method (Otsu, 1979). For each layer, the score distribution is partitioned into two classes by maximizing inter-class variance; tokens in the lower partition skip routed experts. We evaluate two scoring signals: (1) GDN write strength β across 30 GDN layers, and (2) maximum pre-softmax router logits across all 40 MoE layers. A dispersion threshold (coefficient of variation > 0.05–0.40) restricts partitioning to layers with sufficiently bimodal score distributions, subject to a maximum 70% per-layer skip cap. Neither variant requires pre-calibration.

G Additional Analysis on GDN Write Strength β

β functions as an independent salience signal. GDN β provides an informative token pruning signal that operates independently of standard attention mechanisms. While attention distributions are spiky and heavily concentrated, β tends to be more spatially and temporally smooth. Moreover, we find β to have a more even focus across different modalities in MoEMB, compared to attention scores, which are mostly concentrated on text tokens. Despite β's distinct properties from attention scores, pruning tokens by β performs comparably to the trained token pruning gate (Fig. 7), confirming its utility as an effective salience metric. The remainder of this section reports further analysis of β: its per-layer profile, its agreement with the <emb> attention row and with received attention mass, and its spread by token type.

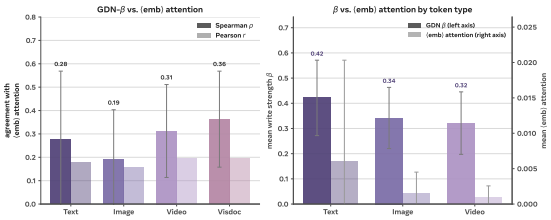

Figure 12: β *vs.* attention score. Left: per-modality agreement (Spearman ρ and Pearson r); Right: mean β (left axis) and mean <emb> attention (right axis) by token type.

Depth Dynamics and Attention Comparison. We evaluate the layer-wise behavior of GDN write strength β using three representative samples per modality. Across all evaluated tokens, the layer-wise mean of β spans $[0.196, 0.596]$, with coefficients of variation ranging from 0.065 to 0.496. In an eight-example diagnostic pairing each of the 10 GDN layers immediately preceding a full-attention layer with its corresponding attention block, the Spearman correlation between β and incoming attention mass reaches 0.546, 0.619, 0.491, and 0.524 for image, text, video, and VisDoc modalities, respectively. However, attention allocation exhibits considerably higher spatial variance, yielding a mean token-wise coefficient of variation of 2.316, compared to 0.226 for β.

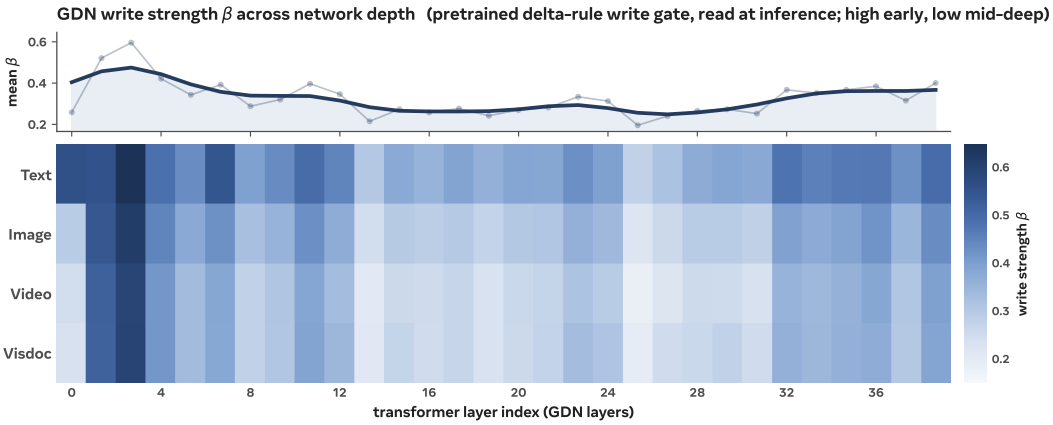

Figure 13: **The GDN write strength is depth-structured.** Per-token Gated-DeltaNet delta-rule write strength β (a gate of the pretrained backbone, read at inference), averaged per layer (top: network-wide marginal; bottom: per modality). Tokens write strongly in the early blocks and weakly through the middle-to-deep blocks. The middle panel of Fig. 9 plots the same quantity beside the two learned actions.

Spatial and Temporal Consistency. Qualitative overlays (Figures 15–17) and layer-wise comparisons across blocks 2, 22, and 38 (Figure 14) demonstrate three key properties of β:

1. Semantic Alignment: β highlights semantically informative regions—such as foreground objects in natural images or plot/text elements in documents—forming contiguous spatial clusters rather than isolated activation noise.
2. Depth Stability: β distributions remain stable across network layers, whereas attention maps shift dynamically. Computing cross-layer cosine similarity yields an average of 0.989 (minimum 0.973) for β, compared to 0.752 (range $[0.68, 0.86]$) for <emb> attention. Tokens identified as high-value in early blocks preserve high scores deeper in the network, even as attention redirects focus.

3. Temporal Persistence: Across video frames within a given layer, β maps exhibit high cross-frame cosine similarity (0.992 vs. 0.744 for attention).

This spatial and temporal stability makes β a reliable candidate for early single-shot pruning, ensuring that token removal decisions remain valid throughout subsequent layers.

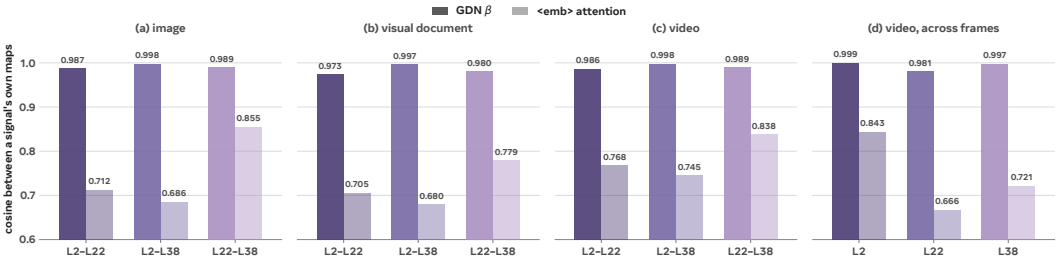

Figure 14: Figures (a)–(c): cosine similarity across layers; Figure (d) compares cosine similarity across video frames within one layer.

Pruning Decision Depth. Table 13 evaluates sensitivity to the layer at which the single-shot pruning mask is generated. Executing the decision at layer 2 rather than layer 4 alters accuracy by $+0.04$ at a 75% keep rate and $+0.21$ at a 25% keep rate, yielding modest incremental FLOP savings of 0.35 and 1.03 percentage points, respectively. These minor variations indicate that pruning effectiveness is broadly insensitive to the exact decision layer.

Table 13: Inference-only GDN-β decision-depth diagnostic. Layer indices are zero-based.

Keep	layer	Accuracy	Δ	FLOPs ↓
75%	2	70.07	−0.25	6.47%
75%	4	70.03	−0.29	6.12%
25%	2	66.26	−4.06	19.36%
25%	4	66.05	−4.27	18.33%

Visual Comparison of Trained and Inference-Only Token Pruning and Expert skipping Methods. Figures 18 and 19 compare trained adaptive computing methods (token pruning and adaptive top-k / expert skipping) with their inference-only counterparts evaluated on identical inputs. As shown in Fig. 18, the single-shot token dropper and the GDN-β policy select nearly identical token subsets. Conversely, Fig. 19 reveals that the trained expert-skipping gate and adaptive top-k routing diverge notably in their allocation decisions.

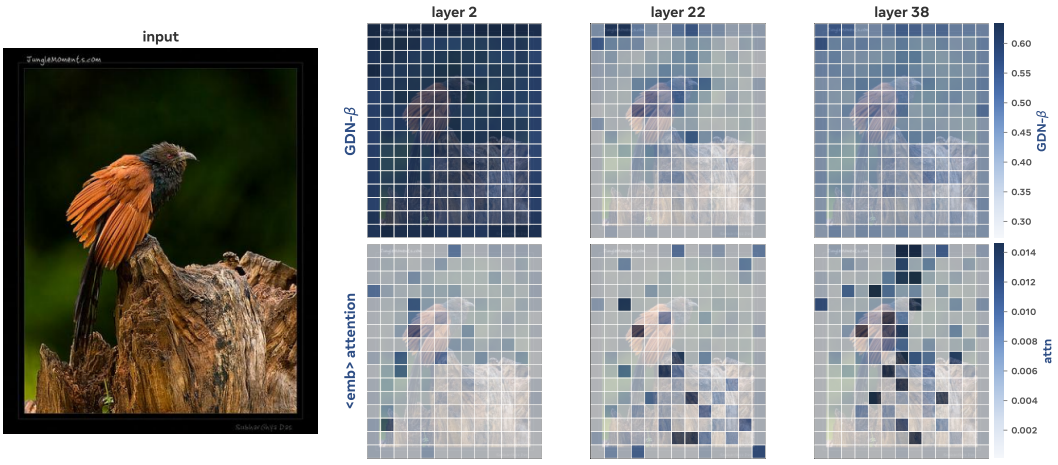

Figure 15: **Image: GDN-β vs. the <emb> attention.** The clean input is at left; the panels overlay GDN-β (top) and the <emb> token's attention onto the visual tokens (bottom) at layers 2, 22, and 38. Cell colour encodes magnitude against the colourbars, with the image kept faintly underneath for spatial reference. β is strong and near-uniform early and localizes smoothly with depth, whereas the <emb> attention is sparse and scattered and only strengthens in the deeper layers.

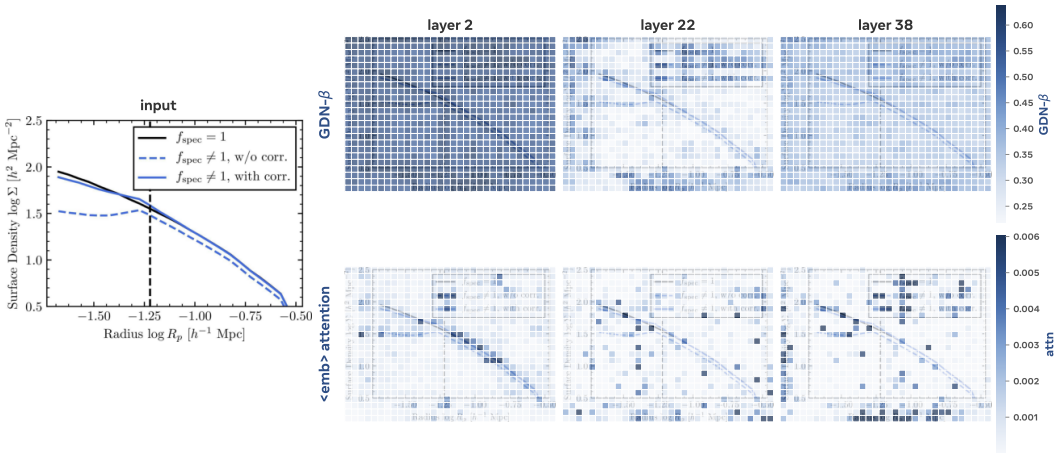

Figure 16: **Visual document: GDN-β vs. the <emb> attention.** Same layout as Fig. 15, on a document page. As on natural images, β is the smoother, more distributed signal across depth.

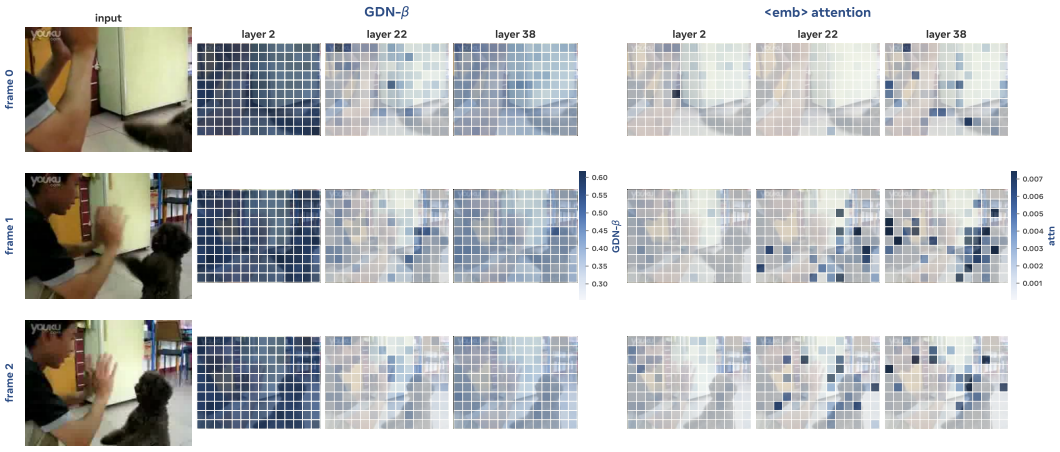

Figure 17: **Video: GDN-β vs. the <emb> attention.** Rows are frames: the clean frame at left, then GDN-β and the <emb> attention side by side, each at layers 2, 22, and 38. The <emb> attention is nearly empty at layer 2 and sharpens into scattered per-frame peaks with depth, whereas β is strong early and spatially smooth, so the two signals carry complementary information.

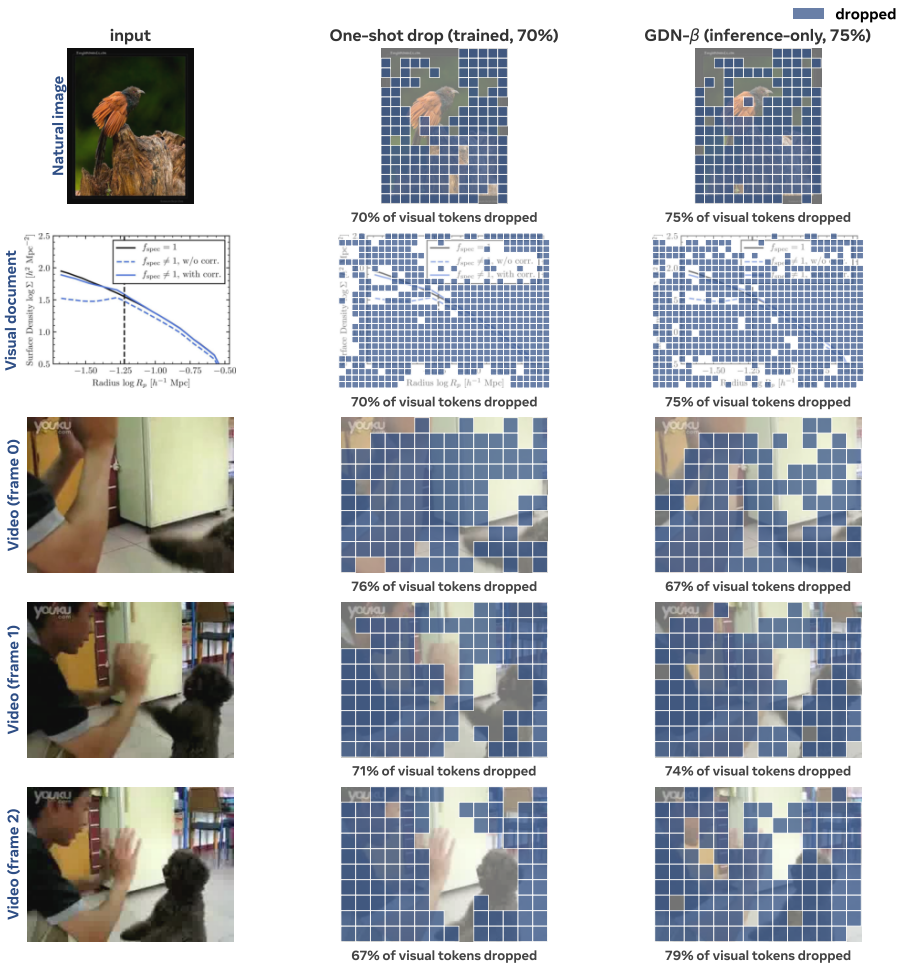

Figure 18: Trained vs. inference-only token dropping, at an equal budget.

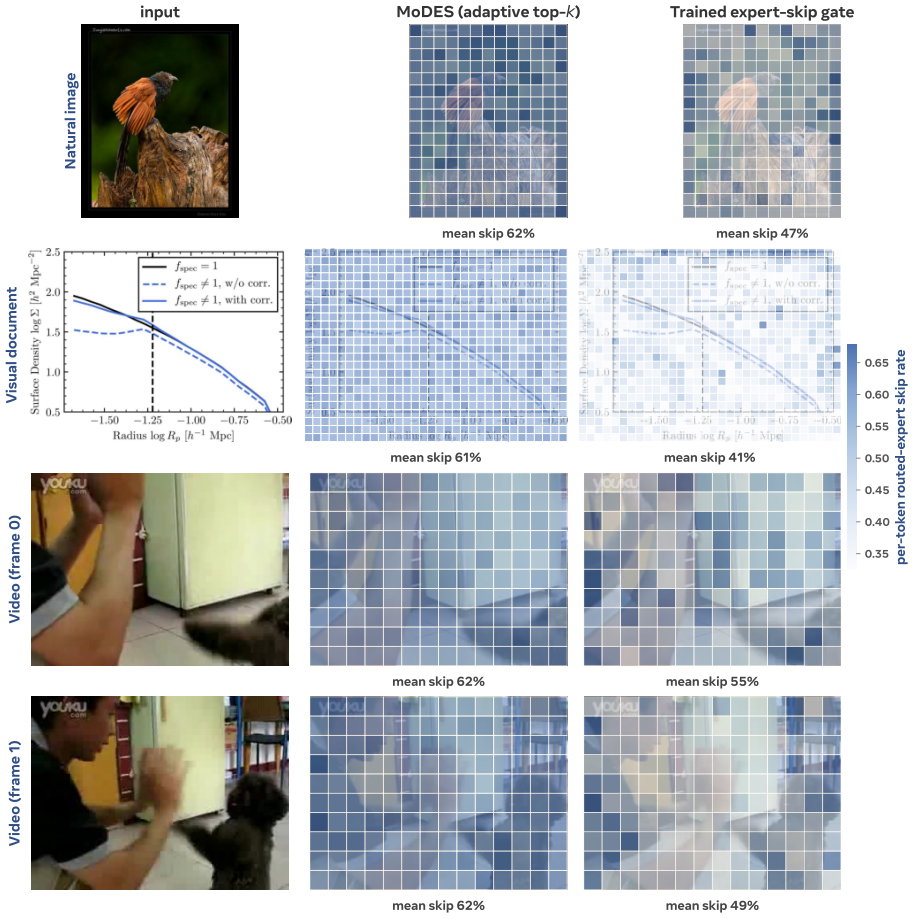

Figure 19: **How the two expert-skipping policies allocate their budget.** Per-token fraction of the 40 MoE layers whose routed-expert branch is skipped, for MoDES (adaptive top-k, inference-only) and the trained expert skipping, at the same overall skip rate.

H PER-TASK MMEB-V2 RESULTS

Table 14: Per-task MMEB-V2 scores.

	Single-pass			TTE-based					MoEMB	
Task	**VLM2Vec V2-2B**	**UniME V2-7B**	**BToks**	**PLUME**	**UME-R1 2B**	**Embed RL-2B**	**UME-R1 7B**	**Embed RL-4B**	**A3B**	**A10B**
Image classification (10 tasks)	62.9	65.6	64.3	66.5	64.8	62.8	**67.1**	63.7	60.6	64.4
Country211	25.8	19.0	25.2	25.0	23.4	20.0	25.0	19.4	21.4	**26.3**
HatefulMemes	55.8	66.1	62.5	75.5	75.2	65.0	**79.0**	66.2	59.8	69.8
ImageNet-1K	**80.8**	78.9	80.5	74.1	75.3	78.0	80.4	79.5	78.1	79.5
ImageNet-A	47.6	49.0	46.9	50.8	50.4	**59.2**	53.9	58.1	52.9	55.2
ImageNet-R	89.3	89.1	85.6	87.5	88.7	88.5	**90.1**	88.2	87.9	88.4
N24News	73.0	66.0	74.0	81.1	81.1	44.9	**82.3**	48.3	39.5	50.5
ObjectNet	65.1	73.2	68.5	61.5	52.0	74.8	42.3	**75.4**	69.1	73.7
Place365	36.1	43.8	38.7	42.4	42.6	43.9	**46.8**	43.1	44.1	45.3
SUN397	70.9	78.8	74.9	76.9	79.4	75.4	**80.3**	79.2	72.9	74.3
VOC2007	84.9	**92.5**	85.7	86.1	80.0	78.7	90.8	79.5	80.8	81.3
Image VQA (10 tasks)	56.4	68.7	59.8	59.2	62.8	67.9	69.2	70.5	73.8	**75.4**
A-OKVQA	44.0	**71.5**	48.6	49.9	51.1	54.7	58.7	59.3	63.1	64.6
ChartQA	48.1	59.2	51.2	49.8	64.9	80.7	75.1	80.9	79.0	**82.5**
DocVQA	90.1	92.4	91.9	89.9	92.2	92.4	93.8	94.3	**96.0**	**96.0**
GQA	65.4	69.0	63.9	69.1	67.3	64.9	69.3	68.5	76.3	**77.6**
InfographicsVQA	59.1	67.1	61.6	59.6	67.7	76.7	79.2	77.5	80.0	**82.6**
OK-VQA	51.7	71.7	61.8	60.5	62.4	61.4	71.7	67.3	73.9	**75.1**
ScienceQA	38.1	55.2	40.1	42.9	42.7	57.3	53.7	61.6	63.6	**67.4**
TextVQA	71.6	84.4	79.5	78.9	78.6	83.8	83.5	84.3	88.5	**89.6**
Visual7W	52.8	**62.7**	49.2	47.6	54.1	52.7	55.2	55.3	59.7	58.8
VizWiz	43.3	53.4	49.8	46.5	46.8	54.5	51.6	56.2	58.0	**60.1**
Text-to-image retrieval (6 tasks)	79.3	82.0	78.6	76.2	77.2	79.2	81.9	80.8	83.2	**85.4**
EDIS	84.2	88.5	85.4	81.8	88.0	84.5	92.0	87.4	91.9	**92.9**
MSCOCO_t2i	75.9	**80.0**	73.9	74.1	75.1	79.4	78.3	78.9	78.7	79.7
VisDial	82.7	84.8	78.4	72.6	76.6	81.5	80.7	84.9	85.1	**88.4**
VisualNews_t2i	74.7	77.3	72.7	71.3	71.7	71.9	76.8	73.7	76.4	**81.9**
WebQA	90.6	90.2	90.5	89.1	90.0	89.3	**90.9**	90.5	89.1	**90.9**
Wiki-SS-NQ	67.6	70.9	70.5	68.6	62.0	68.9	72.5	69.6	77.8	**78.8**
Image-to-text retrieval (2 tasks)	74.7	77.3	72.4	71.2	71.5	74.4	76.7	75.1	80.3	**80.9**
MSCOCO_i2t	71.1	74.6	69.1	69.8	68.9	75.3	71.4	76.3	76.9	**77.7**
VisualNews_i2t	78.3	80.1	75.8	72.7	74.2	73.6	82.0	73.9	83.8	**84.1**
Image-to-image retrieval (8 tasks)	64.8	**74.3**	64.8	66.0	64.2	70.1	69.7	73.2	72.4	74.1
CIRR	57.3	**67.0**	54.0	54.6	53.7	47.6	55.3	61.2	52.5	54.9
FashionIQ	19.6	27.0	19.1	20.3	17.1	24.0	23.4	**31.9**	21.2	26.7
MSCOCO	66.2	81.3	66.4	66.9	69.5	92.9	72.7	**93.6**	92.3	91.2
NIGHTS	68.4	68.3	68.1	68.0	67.2	66.3	68.1	66.4	67.9	**69.2**
OVEN	64.8	68.5	67.8	68.4	66.9	61.4	**71.4**	60.7	68.0	71.0
RefCOCO	87.0	95.3	87.2	86.5	83.3	94.9	91.4	95.9	96.1	**96.6**
RefCOCO-Matching	86.3	**92.8**	86.4	88.4	84.4	85.8	91.1	88.0	91.3	92.7
Visual7W-Pointing	69.0	**94.0**	69.4	74.9	71.5	88.0	84.2	87.9	89.8	90.4
Video classification (5 tasks)	39.2	37.2	43.7	45.0	44.3	57.0	48.6	57.6	53.8	**58.1**
Breakfast	14.8	18.0	18.0	20.1	20.1	**36.7**	21.5	33.0	30.0	36.5
HMDB51	40.2	42.8	47.1	51.2	54.4	56.7	58.3	**60.1**	47.5	54.4
Kinetics-700	38.2	38.0	43.1	42.2	35.8	55.8	42.8	**56.8**	51.6	55.6
SmthSmthV2	43.0	25.2	41.0	44.8	44.1	56.7	50.4	59.5	59.4	**61.8**
UCF101	60.0	61.8	69.3	66.5	67.2	79.3	70.0	78.5	80.7	**82.1**
Video QA (5 tasks)	34.7	50.6	47.0	52.3	50.9	55.9	60.7	58.4	65.9	**67.5**
ActivityNetQA	53.0	64.6	64.8	69.2	57.8	74.8	76.0	74.4	77.4	**80.8**
EgoSchema	35.0	51.6	37.0	47.8	45.4	53.0	52.4	52.8	60.4	**60.6**
MVBench	33.6	42.2	45.4	47.4	49.9	50.8	58.2	55.9	63.0	**64.8**
NExTQA	20.9	58.8	47.9	57.3	60.0	53.9	69.6	58.2	73.7	**74.9**
Video-MME	30.8	35.8	39.9	40.0	41.7	47.1	47.3	50.5	55.0	**56.6**
Video retrieval (5 tasks)	28.4	28.9	33.0	33.5	32.9	45.1	38.2	45.1	45.3	**47.3**
DiDeMo	30.0	31.5	33.0	32.7	32.4	45.3	40.0	46.8	48.9	**50.9**
MSR-VTT	27.8	27.6	33.8	36.2	34.3	45.7	38.9	46.2	45.9	**47.4**
MSVD	47.3	57.5	56.0	56.1	55.4	67.2	60.7	65.8	66.9	**67.5**
VATEX	26.2	22.5	27.6	28.2	29.9	**43.6**	32.6	43.4	40.5	42.5
YouCook2	10.6	5.6	14.5	14.5	12.7	23.5	18.5	23.3	24.6	**28.1**
Moment retrieval (3 tasks)	37.5	39.6	33.6	46.7	39.7	49.4	39.3	**49.5**	45.8	49.4
Charades-STA	20.1	30.0	18.0	19.4	20.4	26.4	21.9	25.0	21.3	**30.7**
MomentSeeker	42.9	32.0	40.4	**63.5**	41.2	50.9	41.1	49.9	49.3	48.6
QVHighlight	49.7	56.8	42.2	57.1	57.5	70.7	54.8	**73.6**	66.8	69.1
Visual-document retrieval (24 tasks)	68.7	60.1	62.7	67.5	67.3	74.1	70.6	74.7	81.0	**82.2**

Continued on next page

Table 14 – continued from previous page

	Single-pass			TTE-based					MoEMB	
Task	**VLM2Vec V2-2B**	**UniME V2-7B**	**BToks**	**PLUME**	**UME-R1 2B**	**Embed RL-2B**	**UME-R1 7B**	**Embed RL-4B**	**A3B**	**A10B**
MMLongBench-doc	–	40.8	41.4	32.0	39.7	50.3	41.3	50.7	86.1	**86.4**
MMLongBench-page-fixed	44.7	–	–	39.9	–	47.7	–	51.0	71.4	**72.3**
ViDoRe_arxivqa	78.9	51.9	76.5	72.6	73.9	86.1	73.6	88.7	90.9	**91.2**
ViDoRe_biomedical_lectures_v2_multilingual	44.6	33.8	39.1	48.2	46.1	51.0	50.7	50.1	59.7	**61.3**
ViDoRe_docvqa	37.1	38.2	37.2	36.2	37.9	45.7	41.1	47.5	54.0	**54.1**
ViDoRe_economics_reports_v2_multilingual	42.3	36.0	39.7	49.6	45.7	53.0	57.8	53.9	61.9	**65.4**
ViDoRe_esg_reports_human_labeled_v2	45.8	54.7	40.0	52.1	50.2	56.9	50.4	59.8	58.0	**68.7**
ViDoRe_esg_reports_v2_multilingual	45.7	43.6	35.7	49.0	42.6	46.9	43.2	49.7	51.5	**55.4**
ViDoRe_infovqa	82.7	73.2	80.9	79.0	76.2	86.8	80.8	86.9	91.5	**91.5**
ViDoRe_shiftproject	61.0	45.4	62.3	64.8	66.8	70.7	65.0	69.0	78.7	**79.0**
ViDoRe_syntheticDocQA_artificial_intelligence	89.1	76.8	79.9	83.8	85.9	94.0	89.5	91.6	96.3	**96.5**
ViDoRe_syntheticDocQA_energy	86.3	77.3	85.3	82.6	83.3	86.7	85.7	88.1	91.5	**93.2**
ViDoRe_syntheticDocQA_government_reports	85.6	79.9	81.0	83.2	82.6	89.0	89.8	90.7	94.4	**94.8**
ViDoRe_syntheticDocQA_healthcare_industry	91.1	81.7	85.3	91.1	90.8	91.1	94.3	90.4	**95.5**	**95.6**
ViDoRe_tabfquad	87.8	57.7	80.4	88.8	86.1	94.5	90.2	94.7	94.4	**95.5**
ViDoRe_tatdqa	44.3	35.5	42.0	36.6	40.6	54.6	46.7	54.8	62.8	**63.9**
ViDoSeek-doc	–	75.8	78.3	76.9	75.9	82.6	75.3	82.4	89.0	**89.1**
ViDoSeek-page-fixed	80.3	–	–	80.6	–	82.0	–	84.4	**84.5**	83.8
VisRAG_ArxivQA	76.7	53.1	76.9	71.6	74.3	84.9	80.5	86.9	88.9	**89.7**
VisRAG_ChartQA	84.2	83.7	86.2	80.8	86.0	88.3	84.9	88.5	87.4	**90.9**
VisRAG_InfoVQA	85.9	82.6	88.7	85.7	84.4	90.0	89.2	89.6	94.2	**94.4**
VisRAG_MP-DocVQA	71.8	66.4	78.6	74.9	75.6	79.1	83.4	79.3	**89.2**	89.1
VisRAG_PlotQA	65.9	51.0	65.6	66.2	68.0	73.0	72.7	72.4	**75.3**	74.2
VisRAG_SlideVQA	91.4	86.4	91.8	88.9	87.1	92.3	91.5	92.6	95.8	**96.3**