SLaMES: Sensitivity-Guided Layerwise and Modality-Aware Expert Skipping for Efficient Inference in MoE MLLMs

Anonymous authors
Paper under double-blind review

Abstract

Mixture-of-Experts (MoE) multimodal large language models (MLLMs) scale capacity efficiently, but fixed Top-K routing activates the same number of experts per token, causing substantial inference overhead for visual-token-heavy inputs. Existing training-free expert-skipping methods use coarse, layer-shared, or skip-rate-invariant criteria, overlooking that skipping tolerance varies across layers, modalities, and ratios. We propose SLaMES, a training-free adaptive framework that exploits this heterogeneity through three components. First, *Layer–Modality Sensitivity Calibration* quantifies sensitivity across layers, modalities, and skipping ratios. Second, *Sensitivity-Guided Threshold Search* uses calibrated sensitivities and offline lookup to derive layer- and modality-specific thresholds with only a few full-model forward passes. Third, *Adaptive Routing-Based Skipping* allocates expert computation based on cumulative routing probability mass, with conditional Top-1 protection for the dominant expert. Across three MoE MLLMs and nine image and video benchmarks, SLaMES retains at least 99.25% of the original performance at a 50% expert-skipping ratio. At the most aggressive settings of 83.4%–87.5%, it retains 96.39%–98.14% of the original performance, consistently outperforming existing expert-skipping methods. Integrated into vLLM, these savings translate into prefill speedups of up to 2.2× on real hardware, demonstrating the efficiency of SLaMES.

1 Introduction

Multimodal large language models (MLLMs) have become a powerful foundation for image, video understanding and multimodal reasoning (Liu et al., 2023a; Bai et al., 2023; Alayrac et al., 2022; Li et al., 2023). Mixture-of-Experts (MoE) architectures further scale their capacity by routing each token to only a subset of experts (Shazeer et al., 2017; Fedus et al., 2022; Lepikhin et al., 2020; Du et al., 2022). However, routing sparsity alone does not guarantee efficient inference: the standard Top-K policy assigns the same number of experts to every token regardless of its need, and because multimodal sequences are dominated by visual tokens, even modest per-token redundancy accumulates into substantial expert computation and communication overhead (Li et al., 2024b).

To mitigate this inefficiency of inference, visual-token pruning (Chen et al., 2024; Yang et al., 2025; Zhang et al., 2025) shortens the sequence by removing redundant visual tokens, but all later computations are discarded irreversibly; static expert pruning applies the same compression strategy to every input, failing to account for variations in input-dependent activation patterns. Dynamic expert skipping (Lu et al., 2024; Huang et al., 2025; 2026) is more flexible, yet still relies on coarse criteria that ignore how sensitivity depends jointly on layer, modality, and operating skipping ratio, as well as on each token's routing distribution. They can therefore meet a target skipping ratio while spending the remaining computation inefficiently.

Our analysis (Figure 1) confirms these gaps. Textual tokens are more sensitive than visual tokens, and the gap widens at higher skipping ratios. Layer priorities shift with both modality and ratio. A fixed per-expert cutoff keeps the dominant expert of a concentrated routing distribution but removes every expert of a diffuse one. We therefore propose **SLaMES**, a training-free framework that makes skipping decisions at two levels. Globally, it uses calibrated, rate-dependent sensitivity to search

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.

layer- and modality-specific thresholds under a target skipping ratio. Per token, it skips the low-probability expert tail bounded by cumulative routing mass, while protecting the dominant expert when routing confidence is high. Integrated into the production-grade vLLM (Kwon et al., 2023) inference system with expert parallelism, SLaMES successfully turns fewer expert assignments into actual computation and communication savings. Our contributions can be summarized as follows:

- We characterize how expert-skipping sensitivity varies across layers, modalities, and skipping ratios in MLLMs, revealing why a single global or skip-rate-invariant criterion cannot allocate expert computation effectively.
- We propose SLaMES, a training-free framework that uses calibrated, rate-dependent sensitivity to search for layer- and modality-specific thresholds under a target skipping budget. At inference time, it adapts the number of retained experts to each token's routing distribution through cumulative-mass skipping and conditional Top-1 protection.
- We validate SLaMES across three MoE MLLMs and nine multimodal benchmarks, outperforming existing expert-skipping methods in average relative accuracy at all settings. Integrated into vLLM with expert parallelism, it achieves measured end-to-end prefill speedups of up to $2.2\times$.

2 Related Work

Token pruning in MLLMs. Existing methods reduce visual-token redundancy at different stages: VisionZip (Yang et al., 2025) prunes tokens before the language model, FastV (Chen et al., 2024) prunes them after shallow layers, and SparseVLM (Zhang et al., 2025) performs text-conditioned, layer-adaptive sparsification. Although these methods reduce subsequent computation by shortening the sequence, removed tokens cannot participate in later layers, limiting fine-grained computation allocation.

Expert pruning and skipping in MoE LLMs. Static methods prune or replace experts offline using demonstration data, router signals, or learned layer-wise ratios (Dong et al., 2026; Liu et al., 2024a; Xie et al., 2024; Bai et al., 2025; Zhang et al., 2026), resulting in expert configurations that remain fixed across inputs. Dynamic methods, such as NAEE (Lu et al., 2024) and MC-MoE (Huang et al., 2025), instead skip experts on a per-token basis, but primarily target text-only MoE LLMs without considering modality-dependent routing.

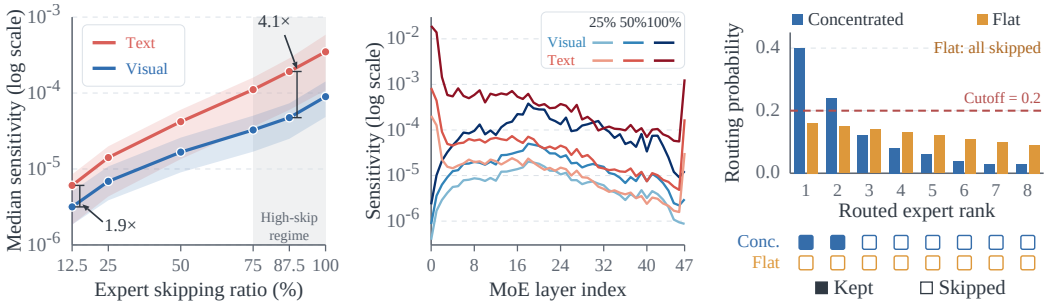
Adaptive expert allocation in MoE MLLMs. MoDES (Huang et al., 2026) performs expert skipping using layer importance and modality-specific thresholds. However, its fixed threshold for each modality and skip-rate-independent layer importance do not capture joint layer–modality–rate variation or token-specific routing patterns. AnyExperts (Gao et al., 2026) and MoE-GRPO (Ko et al., 2026) enable adaptive expert selection through learned importance estimation and reinforcement learning, respectively, but require additional training or model modification.

3 Motivation

Standard MoE routing assigns every token a fixed Top-K expert budget even when fewer experts may suffice, and in MLLMs this redundancy is amplified by the large number of highly redundant visual tokens. Although existing approaches reduce computation at different granularities (Section 2), current training-free strategies neither model how skipping tolerance varies jointly across layers, modalities, and skipping ratios, nor adapt the retained expert count to each token's routing distribution. We examine these two gaps in turn: Section 3.1 analyzes the joint heterogeneity of skipping sensitivity, and Section 3.2 shows why the resulting thresholds still require token-adaptive application. Together, these observations motivate SLaMES, which combines rate-dependent layer–modality sensitivity with routing-mass-aware expert allocation.

3.1 Layer-, Modality-, and Rate-Dependent Skipping Sensitivity

Visual tokens generally tolerate more expert skipping than text tokens (Huang et al., 2026), as our controlled experiment confirms (Appendix A.2). We further quantify how this difference interacts with layer depth and skipping ratio, using the sensitivity defined in Section 4.1, where a larger value indicates lower tolerance to skipping. Figure 1a shows that text is more sensitive than vision at the same skipping ratio, and that its sensitivity grows faster: the text-to-visual sensitivity gap rises



(a) Modality-level sensitivity (b) Layer-wise sensitivity (c) Hard-cutoff failure

Figure 1: Sensitivity heterogeneity on Qwen3-VL-30B-A3B and hard-cutoff failure. (a) Median sensitivity across layers of two modalities with interquartile bands; the gap enlarges as the skipping ratio increases. (b) Distinct layerwise sensitivity patterns of visual and textual modalities under varying skipping ratios. (c) Failure of a fixed routing threshold under different token activation patterns.

from $1.9\times$ at 12.5% skipping ratio to $4.1\times$ at 87.5%. Figure 1b further shows that visual sensitivity concentrates in middle layers, whereas text sensitivity peaks in early layers and rises again at the final layer. Within each modality, the layerwise profile also changes with the skipping ratio. Expert-skipping tolerance is therefore jointly determined by modality, layer, and skipping ratio, rather than by any one of these dimensions alone. A global threshold, a single threshold per modality, or a skip-rate-invariant layer score cannot represent this three-way heterogeneity, which motivates rate-dependent layer–modality calibration and per-layer, per-modality threshold search.

3.2 Hard-Threshold Failure on Dynamic Routing Distributions

Layer- and modality-specific thresholds determine where computation should be allocated, but they do not by themselves determine how a threshold should be applied to each token. A fixed threshold cannot fully accommodate the dynamic activation patterns across tokens, which may result in undesired excessive skipping. Figure 1c gives a schematic example: a threshold of 0.2 behaves reasonably for a concentrated routing distribution, retaining its two dominant experts. For a flatter distribution, however, all expert probabilities fall below the same threshold, so the rule skips every routed expert even though their contributions matter. Thus, an identical cutoff can produce qualitatively different and potentially destructive decisions across tokens. This failure motivates a token-adaptive rule that considers the routing distribution jointly: instead of bounding each skipped expert independently, SLaMES bounds their cumulative routing probability mass, allowing the retained expert count to adapt to how concentrated or diffuse each token's routing distribution is.

4 Method

We propose SLaMES, a training-free framework for expert skipping in MLLMs. As illustrated in Figure 2, the framework comprises three components: (1) *Layer–Modality Sensitivity Calibration*, (2) *Sensitivity-Guided Threshold Search*, and (3) *Adaptive Routing-Based Skipping*. Together, these components determine layer- and modality-specific skipping thresholds under a target global expert-skipping ratio and dynamically adapt for each token based on its routing distribution, thereby mitigating performance degradation while reducing both computation and communication.

4.1 Layer–Modality Sensitivity Calibration

We quantify the sensitivity of every layer–modality–ratio configuration through a lightweight calibration. Let $\mathcal{L} = \{1, \ldots, L\}$ denote the MoE layers, $\mathcal{M} = \{\text{text}, \text{visual}\}$ the modalities, and $\mathcal{R} = \{q/K : q = 1, \ldots, K\}$ a model-specific discrete set of skipping ratios, where $r \in \mathcal{R}$ is the fraction of the K routed experts to be skipped. We first measure the global importance of each layer–modality pair (l, m): we set $r = 1$, skipping all K routed experts at layer l for tokens of modality m while leaving the other modality, residual paths, and any shared experts unchanged, and complete the forward pass. Let T denote the calibration dataset, and let p_t^{orig} and $p_t^{(l,m,1)}$ be the

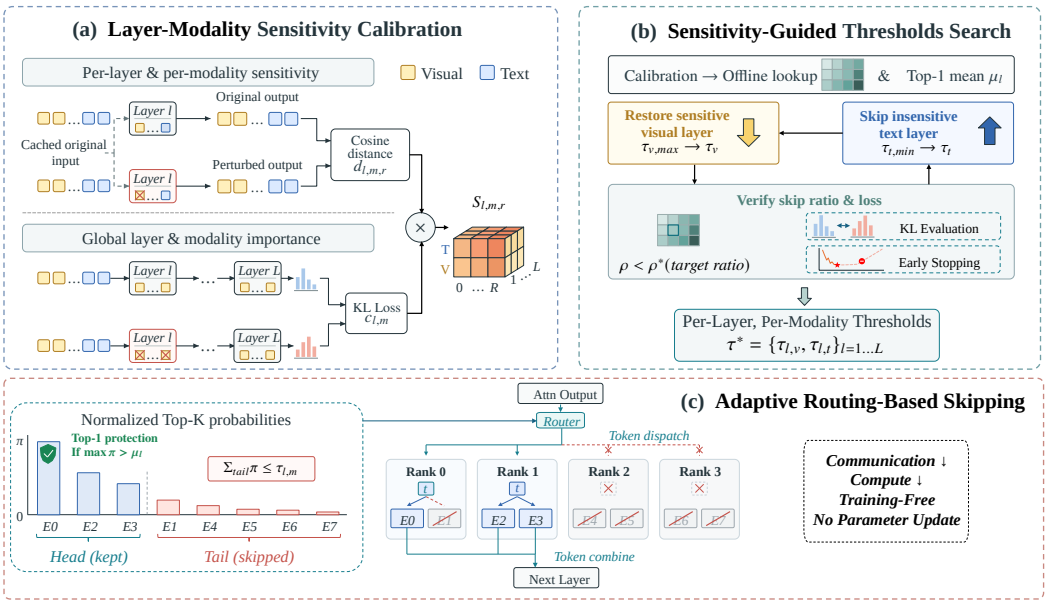


Figure 2: Overview of SLaMES. (a) Isolated perturbations estimate skip-rate-dependent layer–modality sensitivity. (b) Sensitivity-guided frontier search finds per-layer, per-modality thresholds under the target budget. (c) Inference skips a low-probability expert tail by cumulative routing mass while conditionally protecting confident Top-1 experts.

original and perturbed output distributions for input t. We define

$$\widehat{c}_{l,m} = \frac{1}{|T|} \sum_{t \in T} D_{\mathrm{KL}}\left(p_t^{\mathrm{orig}} \,\|\, p_t^{(l,m,1)}\right), \qquad c_{l,m} = \frac{\widehat{c}_{l,m}}{\sum_{j=1}^{L} \widehat{c}_{j,m}}, \tag{1}$$

where $c_{l,m}$ is normalized across layers within the same modality, and a larger value indicates a greater impact of skipping on the final prediction.

As described above, $c_{l,m}$ is calibrated only under complete routed-expert skipping, it is independent of the operating ratio. Since measuring the output divergence for every (l, m, r) would require one end-to-end forward pass per intervention, we therefore run a single forward pass without skipping and cache the input x_l^{orig} and output y_l^{orig} of every MoE layer. To probe (l, m, r), we reset layer l to x_l^{orig}, skip the $\lfloor Kr \rfloor$ lowest-weight experts among its Top-K routes only for modality-m tokens, and recompute only that layer, isolating the local effect of skipping from the remaining layers. With T_m the valid modality-m tokens in the calibration set and $\widetilde{y}_l^{(l,m,r)}$ the resulting isolated layer output, we define the rate-dependent local sensitivity as

$$d_{l,m,r} = \frac{1}{|T_m|} \sum_{t \in T_m} \left[1 - \cos\left(y_{l,t}^{\mathrm{orig}}, \widetilde{y}_{l,t}^{(l,m,r)}\right)\right]. \tag{2}$$

A larger $d_{l,m,r}$ indicates lower tolerance to skipping at ratio r. Coupling this local response with the rate-invariant global prior $c_{l,m}$ yields the calibrated sensitivity (Figure 2(a))

$$S_{l,m,r} = c_{l,m} \cdot d_{l,m,r}, \qquad l \in \mathcal{L},\ m \in \mathcal{M},\ r \in \mathcal{R}. \tag{3}$$

A larger $S_{l,m,r}$ indicates that further skipping at the current ratio is more likely to affect the final prediction. Because this priority changes with the allocated ratio, assigning layer- and modality-specific thresholds under a global compute budget requires adaptive optimization.

4.2 Sensitivity-Guided Threshold Search

To search for layer–modality thresholds $\tau_{l,m}$ that meet a target global skipping ratio ρ^*, we discretize the threshold range as $\mathcal{T} = \{\tau^{(1)} < \cdots < \tau^{(G)}\}$ and let $g_{l,m}$ index the current threshold for (l, m). Since exhaustive search would require G^{2L} full-model evaluations, we instead build a lookup table

$\mathcal{U}$ from routing distributions recorded in one baseline pass. For each (l, m, g), $\mathcal{U}$ stores the estimated skipped expert count $\widehat{N}^{\text{skip}}_{l,m}(\tau^{(g)})$, the local ratio $\widehat{r}^{(g)}_{l,m} = \widehat{N}^{\text{skip}}_{l,m}(\tau^{(g)})/(KN_m)$, and the calibrated sensitivity $S_{l,m,b(\widehat{r}^{(g)}_{l,m})}$, where N_m is the number of modality-m tokens and b maps a ratio to a bin in $\mathcal{R}$. The estimated global skipping ratio of a configuration $\boldsymbol{\tau} = \{\tau_{l,m}\}_{l\in\mathcal{L},m\in\mathcal{M}}$ is

$$\widehat{\rho}_{\mathcal{U}}(\boldsymbol{\tau}) = \frac{\sum_{l\in\mathcal{L}}\sum_{m\in\mathcal{M}} \widehat{N}^{\text{skip}}_{l,m}(\tau_{l,m})}{K\,L\sum_{m\in\mathcal{M}} N_m}. \tag{4}$$

Once the lookup table is constructed, we proceed with the search. Since visual tokens tolerate more skipping (Section 3.1), we initialize all visual thresholds to the maximum $\tau^{(G)}$ and all text thresholds to the minimum $\tau^{(1)}$. Let $\mathcal{F}_{\text{vis}}$ contain visual layers whose thresholds can be decreased by one step, and $\mathcal{F}_{\text{text}}$ text layers whose thresholds can still be increased. We select the most sensitive visual layer to restore and the least sensitive text layer to skip further:

$$l^{\star}_{\text{vis}} = \arg\max_{l\in\mathcal{F}_{\text{vis}}} S_{l,\text{visual},\bar{r}_{l,\text{visual}}}, \qquad l^{\dagger}_{\text{text}} = \arg\min_{l\in\mathcal{F}_{\text{text}}} S_{l,\text{text},\bar{r}_{l,\text{text}}}, \tag{5}$$

where $\bar{r}_{l,m} = b(\widehat{r}^{(g_{l,m})}_{l,m})$ is the current sensitivity bin. We decrease the threshold of $l^{\star}_{\text{vis}}$ until either $\widehat{\rho}_{\mathcal{U}}(\boldsymbol{\tau})$ would fall below ρ^{*} after the next decrease or its ratio enters a new sensitivity bin; we then refresh $\bar{r}$ and reselect the layer. When no visual step remains feasible, we validate the boundary configuration with one full-model evaluation, increase the threshold of $l^{\dagger}_{\text{text}}$ by one step, and repeat. For the evaluated boundary configurations $\{\boldsymbol{\tau}^{(q)}\}_{q=1}^{Q}$, we compute the KL loss on calibration set T:

$$\mathcal{L}_{\text{KL}}(\boldsymbol{\tau}^{(q)}) = \frac{1}{|T|}\sum_{t\in T} D_{\text{KL}}\left(p^{\text{orig}}_t \,\|\, p^{\boldsymbol{\tau}^{(q)}}_t\right), \tag{6}$$

and select $\boldsymbol{\tau}^{\star} = \arg\min_{1\le q\le Q} \mathcal{L}_{\text{KL}}(\boldsymbol{\tau}^{(q)})$ as the final threshold configuration. During the whole procedure, intermediate updates rely only on $\mathcal{U}$, so full forward passes are needed only for the $Q \le 1 + L(G-1)$ boundary configurations, and early stopping after P consecutive evaluations without improvement reduces this number further. Figure 2(b) illustrates the search, and Algorithm 1 (Appendix A.10) details the procedure.

4.3 Adaptive Routing-Based Skipping

Routing-mass-based expert skipping. Motivated by the token-level failure of hard cutoffs in Section 3.2, we apply each selected threshold to the cumulative routing mass rather than to each expert probability independently. At inference time, for token t at layer l with Top-K routed experts $\mathcal{E}^{K}_{l,t}$ and router weights $w_{l,t,k}$, we normalize the weights within this set as $\pi_{l,t,k} = w_{l,t,k}/\sum_{q\in\mathcal{E}^{K}_{l,t}} w_{l,t,q}$. Then we sort them in ascending order, $\pi_{l,t,(1)} \le \cdots \le \pi_{l,t,(K)}$, let $e_{l,t,(j)}$ denote the expert associated with $\pi_{l,t,(j)}$, and let $m(t)$ denote the token modality. The cumulative criterion finds the largest low-weight prefix whose total mass does not exceed the threshold $\tau_{l,m(t)}$:

$$C_{l,t,j} = \sum_{i=1}^{j} \pi_{l,t,(i)}, \qquad j^{\star}_{l,t} = \max\left\{j \in \{0,\dots,K\} : C_{l,t,j} \le \tau_{l,m(t)}\right\}, \tag{7}$$

where $C_{l,t,0} = 0$. Accordingly, the set of skipped experts is given by $\{e_{l,t,(j)} : 1 \le j \le j^{\star}_{l,t}\}$.

Conditional Top-1 protection. To protect confident routing decisions without imposing an unconditional one-expert minimum, we precompute the mean Top-1 probability of each layer from the recorded calibration routes:

$$\mu^{(1)}_{l} = \frac{1}{N_l}\sum_{t=1}^{N_l} \pi_{l,t,(K)}, \qquad P_{l,t} = \mathbb{I}\left[\pi_{l,t,(K)} > \mu^{(1)}_{l}\right], \tag{8}$$

where N_l is the number of calibration tokens at layer l. The final retained-expert set is the union of the experts retained by the cumulative criterion and by Top-1 protection:

$$\mathcal{E}^{\text{keep}}_{l,t} = \left\{e_{l,t,(j)} : j > j^{\star}_{l,t}\right\} \cup \left\{e_{l,t,(K)} : P_{l,t} = 1\right\}. \tag{9}$$

Table 1: Main results of SLaMES across models and benchmarks (as vLLM does not support video inputs for Kimi-VL, we leave its video results blank). ρ^* denotes the target skipping ratio. The best and second-best results within each setting are marked in bold and underlined, respectively.

ρ^*	Method	Image Understanding							Video Understanding		Avg. Rel. (%)
		ChartQA	DocVQA	InfoVQA	OCRBench	MMB_{EN}	MMB_{CN}	RWQA	MVBench	V-MME	
Qwen3-VL-30B-A3B-Instruct (Top-8)											
0%	Baseline	85.28	93.30	81.03	907	87.11	85.74	73.99	70.67	67.67	100.00
50%	NAEE	80.84	91.49	77.50	882	85.31	82.30	73.59	69.45	66.52	97.30
	MC-MoE	82.64	91.93	77.76	879	86.00	81.10	74.41	**70.42**	65.63	97.65
	MoDES	82.60	92.54	78.41	884	**87.37**	**85.05**	**75.56**	69.88	67.15	98.89
	SLaMES	**84.88**	**93.16**	**80.29**	**900**	87.29	84.79	72.55	69.80	**67.44**	**99.25**
62.5%	NAEE	74.52	87.73	68.58	719	72.51	79.38	70.07	67.63	64.67	89.68
	MC-MoE	78.00	89.55	71.00	827	84.02	79.64	70.07	68.80	63.89	93.56
	MoDES	82.24	92.24	77.66	883	**87.46**	**85.40**	73.46	69.30	**66.96**	98.31
	SLaMES	**84.48**	**92.90**	**79.24**	**891**	87.11	84.62	**73.85**	**70.03**	66.70	**98.98**
75%	NAEE	48.32	57.88	44.20	640	76.89	68.04	63.14	61.48	56.67	74.17
	MC-MoE	59.64	69.68	49.30	681	75.26	71.39	64.18	63.67	59.59	79.46
	MoDES	81.28	91.46	76.68	882	**86.17**	84.79	**75.03**	68.78	66.37	97.76
	SLaMES	**83.68**	**91.93**	**77.47**	**888**	**86.17**	**84.88**	73.59	**69.58**	**66.78**	**98.30**
87.5%	NAEE	4.20	0.73	5.86	137	31.79	20.62	34.77	40.50	37.33	27.56
	MC-MoE	9.20	7.83	9.96	198	41.58	35.22	42.88	48.35	43.85	37.03
	MoDES	76.60	88.81	70.98	844	85.05	**83.42**	73.20	68.40	64.93	94.70
	SLaMES	**80.56**	**90.48**	**73.66**	**876**	**85.57**	82.73	**73.73**	**68.97**	**65.37**	**96.39**
Kimi-VL-16B-A3B-Instruct (Top-6)											
0%	Baseline	87.20	98.73	83.73	883	79.73	80.67	69.28	–	–	100.00
50%	NAEE	84.84	98.34	80.03	855	78.09	79.81	68.89	–	–	97.95
	MC-MoE	85.68	98.39	80.11	859	78.09	79.47	68.10	–	–	97.95
	MoDES	86.92	**98.68**	82.86	874	79.73	**81.01**	69.28	–	–	99.71
	SLaMES	**87.28**	98.57	**83.44**	**882**	**79.90**	80.84	**69.41**	–	–	**100.01**
66.7%	NAEE	80.12	97.41	70.09	772	74.14	76.46	66.80	–	–	92.27
	MC-MoE	80.68	97.12	71.43	775	74.48	76.80	66.93	–	–	92.74
	MoDES	86.40	98.24	79.97	863	79.55	80.33	**70.33**	–	–	98.96
	SLaMES	**86.92**	**98.53**	**82.36**	**880**	**79.73**	**80.76**	69.15	–	–	**99.63**
83.4%	NAEE	50.84	74.56	35.52	458	57.99	61.77	59.09	–	–	66.10
	MC-MoE	56.36	76.85	38.27	499	59.02	63.66	61.44	–	–	69.47
	MoDES	78.92	96.31	71.24	804	**78.18**	78.35	**71.24**	–	–	94.60
	SLaMES	**84.72**	**97.64**	**77.37**	**862**	78.09	**79.98**	69.67	–	–	**97.68**
Qwen3.5-122B-A10B (Top-8)											
0%	Baseline	88.00	91.31	88.14	910	88.14	87.29	81.96	72.90	70.52	100.00
50%	NAEE	85.88	82.55	86.58	882	86.25	85.40	80.01	70.92	68.33	96.73
	MC-MoE	85.32	86.63	86.26	894	85.74	85.89	79.48	71.10	69.56	97.42
	MoDES	83.36	**91.92**	83.73	880	87.03	85.65	78.17	70.05	70.00	97.19
	SLaMES	**87.40**	91.81	**87.82**	**898**	**88.92**	**86.94**	**81.44**	**71.95**	**70.93**	**99.70**
62.5%	NAEE	82.84	76.01	83.04	860	84.97	82.30	78.43	70.63	64.07	93.35
	MC-MoE	82.80	81.40	84.46	872	85.82	84.54	**79.61**	70.60	66.52	95.26
	MoDES	82.52	91.83	82.82	877	86.68	85.05	78.56	69.50	68.96	96.60
	SLaMES	**87.44**	**92.11**	**87.88**	**889**	**88.40**	**86.77**	79.22	**72.55**	**70.63**	**99.25**
75%	NAEE	71.76	67.08	73.14	795	81.44	76.20	76.08	65.72	55.30	85.16
	MC-MoE	74.48	79.22	76.80	788	82.90	80.67	76.99	67.83	61.22	89.49
	MoDES	81.44	91.88	82.07	865	86.77	84.97	78.17	67.72	69.59	96.00
	SLaMES	**86.76**	**92.41**	**87.22**	**891**	**87.80**	**86.68**	**80.00**	**71.92**	**70.04**	**99.02**
87.5%	NAEE	20.16	25.16	28.45	436	64.95	55.07	65.62	46.63	40.11	52.03
	MC-MoE	39.68	54.73	44.84	507	69.33	65.03	67.58	55.55	52.33	66.40
	MoDES	76.44	91.28	79.46	841	86.08	83.33	78.43	67.35	67.67	94.06
	SLaMES	**84.72**	**92.53**	**85.64**	**878**	**86.94**	**85.82**	**79.87**	**71.40**	**70.26**	**98.14**

All routed experts outside $\mathcal{E}_{l,t}^{\text{keep}}$ are skipped. Since $C_{l,t,K} = 1$, the cumulative criterion can skip all K experts only when $\tau_{l,m(t)} \geq 1$; protection then retains the dominant expert of tokens whose routing is more confident than the layer average.

5 Experiments

5.1 Experimental Setup

Models and benchmarks. We evaluate SLaMES on three multimodal MoE models with different architectures, scales, and routing widths: Qwen3-VL-30B-A3B-Instruct ($K = 8$) (Qwen Team,

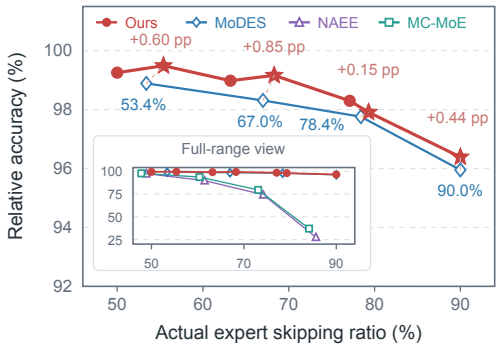


Figure 3: Accuracy versus actual skipping on Qwen3-VL-30B-A3B. SLaMES achieves better accuracy retention even at higher skipping ratio.

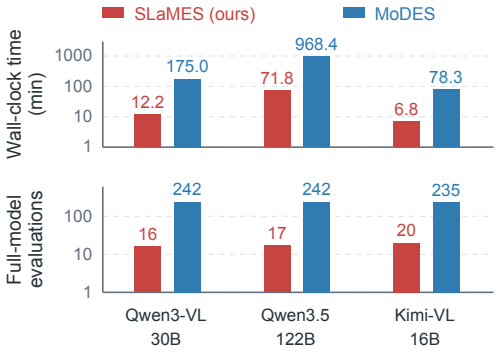


Figure 4: Threshold-search efficiency of SLaMES. Top: wall-clock time consumed; bottom: full-model evaluations required.

2025), Kimi-VL-A3B-Instruct (16B total, $K = 6$) (Kimi Team, 2025), and Qwen3.5-122B-A10B ($K = 8$) (Qwen Team, 2026). The evaluation benchmarks comprise seven image understanding tracks, namely ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021), InfoVQA (Mathew et al., 2022), OCRBench (Liu et al., 2023b), MMBench (English and Chinese) (Liu et al., 2024b), and RealWorldQA (xAI, 2024), and two video understanding benchmarks, MVBench (Li et al., 2024a) and Video-MME (Fu et al., 2025). We use each benchmark's standard metric and report the average relative score, where each task score is normalized by that of the unmodified model.

Baselines. We use the unmodified fixed Top-K model as the reference and compare SLaMES with NAEE (Lu et al., 2024), MC-MoE (Huang et al., 2025), and MoDES (Huang et al., 2026). NAEE and MC-MoE were developed for text-only MoE LLMs and evaluated primarily under Top-2 routing. Following MoDES, we adapt their expert-skipping rules to Top-K routing and evaluate only their expert-skipping components.

Calibration and implementation. For each model and benchmark, all methods use the same checkpoint, inputs, prompts, and generation settings. We consider target skipping ratios of 50%, 62.5%, 75%, and 87.5% for the Top-8 models, and 50%, 66.7%, and 83.4% for the Top-6 Kimi-VL model. For calibration, we use 1,024 sampled GQA examples (Hudson & Manning, 2019) to estimate sensitivity, record routing distributions, and optimize thresholds. We deploy the models with vLLM (Kwon et al., 2023), integrate SLaMES into its fused MoE routing, and evaluate the resulting services through a unified platform (details in Appendix A.1).

5.2 Main Results

SLaMES achieves state-of-the-art accuracy retention. As shown in Table 1, SLaMES achieves the highest average relative score among the compared methods in evaluated settings. At a target skipping ratio of 50%, it retains 99.25%, 100.01%, and 99.70% of baseline performance on Qwen3-VL, Kimi-VL, and Qwen3.5, respectively, indicating near-baseline accuracy at this budget. Even at the highest tested ratios of 87.5%, 83.4%, and 87.5%, it retains 96.39%, 97.68%, and 98.14%, respectively. Figure 3 further compares accuracy against measured rather than preset skipping ratios on Qwen3-VL. SLaMES supplies the observed Pareto-optimal operating points among the compared methods, retaining higher accuracy even at comparable or greater actual skipping. Together, these results show that SLaMES achieves a more favorable accuracy–skipping trade-off.

The advantage widens under aggressive skipping and at larger scale. Relative to the strongest competing method, SLaMES gains 0.30–2.28 percentage points at 50% skipping and 1.69–4.08 points at the most aggressive budget of each model. These gains concentrate on high-resolution, text-dense benchmarks whose sequences are dominated by visual tokens: at 87.5% skipping on Qwen3.5, SLaMES exceeds MoDES by 8.28 points on ChartQA, 6.18 on InfoVQA, and 37 points on OCRBench. Comparing the two Top-8 models over identical budgets, the margin also grows with scale, from 0.36–1.69 points on Qwen3-VL-30B-A3B to 2.28–4.08 points on Qwen3.5-122B-A10B. These widening gaps suggest that fine-grained expert allocation becomes more consequential as the skipping budget tightens and model scales.

Table 2: Component ablation results of SLaMES on Qwen3-VL-30B-A3B-Instruct across six representative benchmarks. CRM, CTP, and LTS abbreviate cumulative routing mass, conditional Top-1 protection, and layer-specific threshold search. ρ^* is the target skipping ratio.

ρ^*	Enabled Components			Image Understanding				Video Understanding		Avg. Rel. (%)
	CRM	CTP	LTS	DocVQA	InfoVQA	OCRBench	MMB_{EN}	MVBench	V-MME	
0%	Baseline			93.30	81.03	907	87.11	70.67	67.67	100.00
50%	✓	✓	✓	93.16	80.29	900	87.29	69.80	67.44	**99.47**
	–	✓	✓	92.76	80.17	902	87.03	69.71	66.96	99.22
	✓	–	✓	93.00	80.44	896	87.03	69.45	67.44	99.26
	✓	✓	–	92.78	79.69	892	86.94	70.03	67.67	99.17
62.5%	✓	✓	✓	92.90	79.24	891	87.11	70.03	66.70	**98.88**
	–	✓	✓	92.72	79.36	888	86.68	69.63	66.22	98.52
	✓	–	✓	92.65	79.22	889	86.94	69.70	67.26	98.82
	✓	✓	–	92.52	77.66	895	86.51	69.78	66.30	98.28
75%	✓	✓	✓	91.93	77.47	888	86.17	69.58	66.78	**98.02**
	–	✓	✓	90.71	77.85	874	85.12	69.38	66.11	97.21
	✓	–	✓	91.86	76.95	878	86.01	69.40	66.96	97.69
	✓	✓	–	91.13	74.29	877	85.31	68.78	65.52	96.36
87.5%	✓	✓	✓	90.48	73.66	876	85.57	68.97	65.37	**96.15**
	–	✓	✓	90.50	74.37	870	84.68	68.80	65.81	96.09
	✓	–	✓	86.57	65.78	829	83.25	67.67	63.74	91.81
	✓	✓	–	88.99	75.10	860	84.45	68.97	64.37	95.42

5.3 Threshold-Search Efficiency

SLaMES moves most budget tracking outside full-model evaluation: recorded routing distributions provide an offline threshold-to-skip lookup, while true KL loss is evaluated only at sensitivity-guided frontier points until early stopping (Appendix A.3). For a controlled cross-model efficiency comparison, we benchmark SLaMES and MoDES at a common intermediate target skipping ratio of $\rho^* = 60\%$, using matched batch sizes and parallel settings for each model; one evaluation denotes a complete calibration-set pass under one candidate threshold configuration. As shown in Figure 4, SLaMES requires only 16–20 evaluations, compared with 235–242 for MoDES. This corresponds to 11.8–15.1$\times$ fewer full-model evaluations and an 11.5–14.3$\times$ reduction in wall-clock search time across models from 16B to 122B.

5.4 Ablation Studies of SLaMES Components

We isolate three components of SLaMES on Qwen3-VL-30B-A3B while keeping all other settings fixed. Table 2 reports the results on six representative image and video benchmarks; further rate-band and Top-1 threshold analyses appear in Appendix A.4.

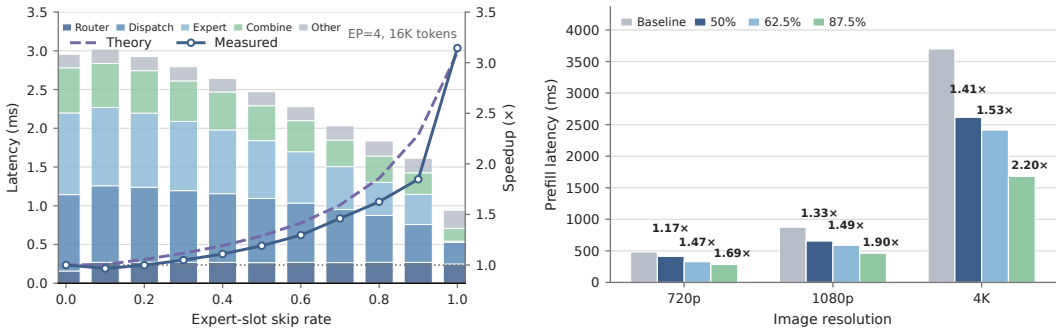
Cumulative routing mass (CRM). Replacing the adaptive cumulative-mass rule with pointwise expert-score comparisons reduces average relative accuracy at every setting, by 0.06–0.81 percentage points. The largest drop occurs at 75% skipping, suggesting that utilizing the cumulative routing mass better preserves accuracy than thresholding individual expert scores independently.

Conditional Top-1 protection (CTP). Protection has a modest effect at moderate skipping ratios but becomes critical under aggressive skipping. At 87.5% skipping, disabling it lowers average relative accuracy from 96.15% to 91.81%, a 4.34-point drop. This confirms that preserving a dominant expert when routing confidence is high helps prevent severe performance degradation.

Layer-specific threshold search (LTS). Searching for a separate threshold for every layer and modality under target skipping ratio consistently outperforms sharing only one threshold per modality. The gain ranges from 0.30 to 1.66 points and peaks at 75% skipping, suggesting that a shared modality-level threshold cannot fully account for differences in skipping tolerance across layers.

5.5 Latency Analysis and Inference Speedup

The preceding experiments establish the accuracy–skipping trade-off of SLaMES. We now examine how fewer token–expert assignments translate into wall-clock savings under expert parallelism, from single-layer MoE latency to end-to-end prefill latency on Qwen3-VL-30B-A3B-Instruct.



(a) Single-layer latency and speedup

(b) End-to-end prefill latency and speedup

Figure 5: Inference speedup on Qwen3-VL-30B-A3B-Instruct with EP=4. (a) Single-layer latency breakdown and predicted versus measured speedup with 16K tokens. (b) End-to-end prefill latency at three image resolutions with batch size 4. Bars compare the unmodified baseline with SLaMES at skipping ratios of 50%, 62.5%, and 87.5%; annotations report speedup relative to the baseline.

Single-layer latency analysis. We model SLaMES under DeepEP (Zhao et al., 2025), a widely adopted high-performance communication backend for large-scale MoE inference and the backend used in our deployment. Its shared token-copy semantics and rank-level load imbalance determine how assignment skipping translates into latency savings. We model layer-l latency as

$$T_l^{\mathrm{SLaMES}} \approx T_{\mathrm{fix},l}(n_l) + T_{\mathrm{sel},l}(n_l, p_l) + \beta_l \Big[(1 - p_l) c_{\mathrm{exp},l} + g_l c_{\mathrm{comm},l} \Big] n_l, \tag{10}$$

where n_l, p_l, g_l, and β_l denote per-rank token count, assignment skipping ratio, retained communication ratio, and load imbalance. The coefficients $c_{\mathrm{exp},l}$ and $c_{\mathrm{comm},l}$ are balanced per-token expert-compute and communication costs; $T_{\mathrm{fix},l}$ and $T_{\mathrm{sel},l}$ are fixed and selection overheads.

We calibrate the latency cost coefficients using only zero-skipping measurements, while g_l and β_l are evaluated from the retained routing pattern. Under EP=4 and 16K tokens, the resulting theoretical curve agrees reasonably well with measured speedup (Figure 5a). Expert-compute latency decreases more strongly than dispatch and combine, while router and other overheads remain relatively stable. The model can thus estimate the skipping ratio required for a target single-layer speedup before SLaMES calibration. Appendix A.6 provides derivation and measurement details.

End-to-end prefill latency. To verify that layer-level savings translate into system-level gains, we integrate SLaMES into vLLM (Kwon et al., 2023) and measure end-to-end prefill latency on real hardware. Figure 5b compares SLaMES with the unmodified model under EP=4 and batch size 4 at matched image resolutions. SLaMES consistently reduces prefill latency, with larger gains at higher skipping ratios and resolutions. These results show that expert-computation savings translate into end-to-end acceleration, especially for larger visual-token workloads. Together with Section 5.2, they characterize the accuracy–latency trade-off across skipping budgets.

6 Conclusions

We introduced SLaMES, a training-free expert-skipping framework for efficient inference in pre-trained MoE MLLMs. We showed that skipping tolerance is jointly determined by layer, modality, and operating skipping ratio, and that no single global or skip-rate-invariant criterion can capture this heterogeneity. SLaMES exploits it at two levels: calibrated rate-dependent sensitivity allocates a target expert-compute budget across layer–modality pairs, and cumulative routing mass adapts the retained expert count to each token's routing distribution at inference time. The search needs only about twenty full-model evaluations and updates no parameters. Across MoE MLLMs of different scales and routing widths, SLaMES achieves state-of-the-art accuracy–efficiency trade-offs on both image and video benchmarks, with the margin widening under aggressive skipping, and its reduced expert assignments translate into measured prefill acceleration on real hardware. These results demonstrate the value of fine-grained, distribution-adaptive expert allocation for practical multimodal MoE inference.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. DivPrune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9392–9401, 2025.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Sikai Bai, Haoxi Li, Jie Zhang, Zicong Hong, and Song Guo. DiEP: Adaptive mixture-of-experts compression through differentiable expert pruning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

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. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024.

Zican Dong, Han Peng, Peiyu Liu, Xin Zhao, Dong Wu, Feng Xiao, and Zhifeng Wang. Domain-specific pruning of large mixture-of-experts models with few-shot demonstrations. *Advances in Neural Information Processing Systems*, 38:103552–103577, 2026.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

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.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhui Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24108–24118, 2025.

Yuting Gao, Lan Wang, Hengyuan Zhao, Linjiang Huang, Si Liu, and Qingpei Guo. AnyExperts: On-demand expert allocation for multimodal language models with mixture of experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9205–9214, 2026.

Yefei He, Feng Chen, Jing Liu, Wenqi Shao, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. ZipVL: Accelerating vision-language models through dynamic token sparsity. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20477–20486, 2025.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. Mixture compressor for mixture-of-experts LLMs gains more. In *International Conference on Learning Representations*, 2025.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30205–30215, 2026.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6700–6709, 2019.

Kimi Team. Kimi-VL technical report. *arXiv preprint arXiv:2504.07491*, 2025. URL https://arxiv.org/abs/2504.07491.

Dohwan Ko, Jinyoung Park, Seoung Choi, Sanghyeok Lee, Seohyun Lee, and Hyunwoo J. Kim. MoE-GRPO: Optimizing mixture-of-experts via reinforcement learning in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14957–14967, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Bo Li, Chuan Wu, and Shaolin Zhu. MACS: Modality-aware capacity scaling for efficient multimodal MoE inference. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22134–22148. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1012. URL https://aclanthology.org/2026.acl-long.1012/.

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

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. MVBench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22195–22206, 2024a.

Yanwei Li, Chengyao Wang, and Jiaya Jia. LLaMA-VID: An image is worth 2 tokens in large language models. In *European Conference on Computer Vision*, pp. 323–340. Springer, 2024b.

Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B. Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916, 2023a.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, pp. 216–233. Springer, 2024b.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *arXiv preprint arXiv:2305.07895*, 2023b.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6159–6172. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.334.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.177.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2200–2209, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. InfographicVQA. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1697–1706, 2022.

Qwen Team. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

xAI. Grok-1.5 Vision Preview, April 2024. URL https://x.ai/news/grok-1.5v. Introduces the RealWorldQA benchmark.

Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. MoE-Pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*, 2024.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. VisionZip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19792–19802, 2025.

Geng Zhang, Yuxuan Han, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. MoNE: Replacing redundant experts with lightweight novices for structured pruning of MoE. In *International Conference on Learning Representations*, 2026.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A. Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 74840–74857, 2025.

Chenggang Zhao, Shangyan Zhou, Liyue Zhang, Chengqi Deng, Zhean Xu, Yuxuan Liu, Kuai Yu, Jiashi Li, and Liang Zhao. Deepep: an efficient expert-parallel communication library. https://github.com/deepseek-ai/DeepEP, 2025.

A Appendix

A.1 Implementation Details

Calibration protocol. All calibration stages use the same fixed subset of 1,024 GQA examples, starting from index zero, with a batch size of 32 and no data shuffling. We append the reference answer to each prompt, use left padding and truncation, and disable the key–value cache during calibration. To obtain the global importance $c_{l,m}$ in Equation 1, we completely skip the routed experts of one layer and one modality at a time, measure KL divergence with temperature $T = 1.0$, and normalize the resulting scores across layers separately for each modality. For the local term $d_{l,m,r}$, we use the isolated layer-recomputation setting described in Section 4.1: the local variant is evaluated only on tokens of the perturbed modality using cosine distance. The Top-8 models are probed at $r \in \{0.125, 0.25, 0.5, 0.75, 0.875, 1.0\}$, while Kimi-VL with Top-6 routing is probed at $r \in \{0.17, 0.34, 0.5, 0.67, 0.84, 1.0\}$. We multiply the local scores by $c_{l,m}$ offline to form $S_{l,m,r}$.

Routing statistics and threshold search. We run one unmodified forward pass over the same calibration subset to record the Top-K expert indices and routing probabilities for every MoE layer and token, together with its text/visual label. Routing collection uses batch size 1; probabilities are stored in FP16, while the layer-wise mean Top-1 probability $\mu_l^{(1)}$ is accumulated at higher precision. Threshold search uses 100 linearly spaced candidates over $[0, 1.01]$, a batch size of 32, and KL divergence with temperature $T = 1.0$ as the validation objective. We evaluate every visited frontier configuration, stop after 15 consecutive evaluations without improving the best KL loss, and set the iteration cap to 100,000 so that early stopping or frontier exhaustion is the effective termination condition. The searched target ratios are $\{50, 62.5, 75, 87.5\}\%$ for the Top-8 models and $\{50, 66.7, 83.4\}\%$ for Kimi-VL. All intermediate budget updates are obtained from the offline routing lookup; a full calibration pass is performed only at the frontier configurations selected by the search.

Deployment and evaluation. We deploy all models with vLLM in BF16 and eager-execution mode, with its memory-utilization limit set to 0.85. The maximum model length is 32,768 tokens for Qwen3-VL-30B-A3B-Instruct and Kimi-VL-A3B-Instruct and 262,144 tokens for Qwen3.5-122B-A10B; thinking mode is disabled for Qwen3.5-122B-A10B. Kimi-VL-A3B uses greedy decoding ($T = 0$), with top-p set to 1 and top-k disabled ($k = -1$). For Qwen3-VL-30B-A3B and Qwen3.5-122B-A10B, we use temperature $T = 0.7$, top-$p = 0.8$, top-$k = 20$, and a presence penalty of 1.5. The baseline and all skipping methods use identical prompts, preprocessing, and generation settings. At inference time, the learned layer–modality thresholds and the layer-wise $\mu_l^{(1)}$ values are loaded by the custom vLLM router; skipped routed-expert slots are counted directly to report realized skipping ratios. For the end-to-end latency evaluation, we measure time to first token (TTFT) using the random-image dataset provided by `vllm_bench`. The batch size is set equal to the expert-parallel size. We evaluate three image resolutions: 720P (1280×720), 1080P (1920×1080), and 4K (3840×2160). The multimodal processor runs on the CPU, with `longest_edge` set to 16,777,216 and `shortest_edge` set to 65,536. For each configuration, we run 2,048 inference requests and report the average TTFT and the corresponding speedup over the baseline.

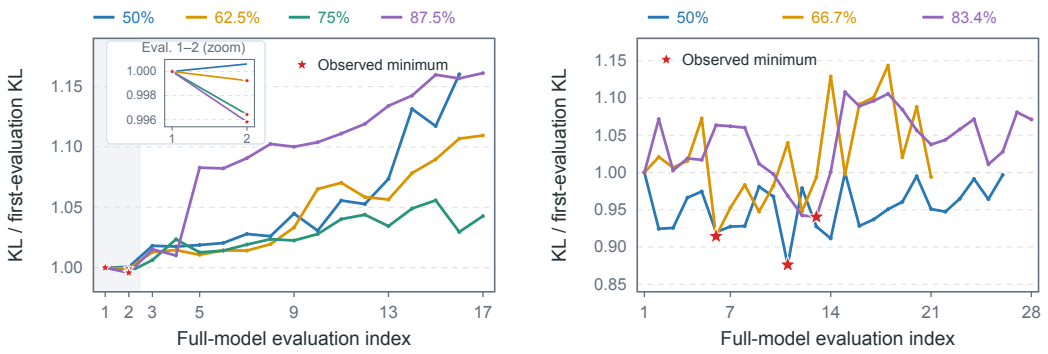
A.2 Modality-Specific Skipping Tolerance

We directly test modality asymmetry on the unmodified Qwen3-VL-30B-A3B-Instruct model with Top-8 routing. At every MoE layer, we skip either four or all eight routed experts only for text or only for visual tokens, leaving the other modality unchanged. We then evaluate the same six benchmarks under each intervention.

At 50% local skipping, the text-only intervention retains 98.68% average relative accuracy despite removing just 2.1% of assignments overall. The visual-only intervention retains 99.32% while removing 47.9% overall, because visual tokens dominate the prefill assignment count. The contrast is stronger when all routed experts of one modality are skipped: text-only skipping nearly collapses performance, whereas visual-only skipping retains 85.93%. These results support modality-aware allocation rather than a uniform expert budget.

Table 3: Modality-isolated expert skipping on Qwen3-VL-30B-A3B-Instruct. Local Skip is the fraction of routed experts removed from tokens of the indicated modality; the other modality is unmodified. Avg. Rel. averages six benchmark scores relative to the baseline. Overall Skip counts skipped routed-expert assignments across prefill and decode; generated lengths can differ between interventions.

Perturbed modality	Local Skip (%)	Image Understanding				Video Understanding		Avg. Rel. (%)	Overall Skip (%)
		DocVQA	InfoVQA	OCRBench	MMB_{EN}	MVBench	V-MME		
None (baseline)	0	93.30	81.03	907	87.11	70.67	67.67	100.00	0.0
Text only	50	91.61	80.83	894	85.14	69.92	66.93	98.68	2.1
Visual only	50	93.00	79.42	905	87.29	69.80	67.33	**99.32**	47.9
Text only	100	0.09	0.00	0	0.00	0.00	0.11	0.04	35.2
Visual only	100	77.37	56.20	753	83.16	66.15	61.70	**85.93**	95.6



(a) Qwen3-VL-30B-A3B

(b) Kimi-VL-16B-A3B

Figure 6: Threshold-search trajectories at different target skipping ratios. KL loss is normalized by the first evaluation of each run; stars mark the lowest observed losses. The Qwen inset enlarges evaluations 1–2, and the panels use different vertical ranges for readability. All traces end at the recorded early-stop point with patience $P = 15$.

A.3 Analysis of Early Stopping

We examine the recorded threshold-search trajectories of Qwen3-VL-30B-A3B and Kimi-VL-16B-A3B at four and three target skipping ratios, respectively. Following Section 4.2, each evaluation measures the KL loss of a frontier configuration with a full calibration pass; intermediate lookup-based updates are not counted as evaluations. Figure 6 plots each trajectory relative to its first-evaluation loss and marks its lowest observed value.

Best observed configurations emerge after few evaluations. For Qwen3-VL-30B-A3B, the lowest observed losses occur at evaluations 1, 2, 2, and 2 for target ratios of 50%, 62.5%, 75%, and 87.5%, respectively; the runs terminate after 16–17 evaluations. For Kimi-VL, the corresponding minima occur at evaluations 11, 6, and 13 for ratios of 50%, 66.7%, and 83.4%, within runs of 26, 21, and 28 evaluations. Thus, across all seven settings, the best recorded configuration is identified within the first 13 full-model evaluations. The trajectories fluctuate or trend upward after their observed minima, illustrating that additional evaluations do not consistently improve the calibration objective. Because these runs stop after 15 consecutive non-improving evaluations, the absence of a new minimum in that final window follows from the stopping rule itself; it is not independent evidence that a longer search could never improve the loss.

Practical rationale for patience-based stopping. Unlike lookup-based threshold updates, each loss evaluation requires a full calibration pass. These observations motivate a patience-based rule that limits continued expensive evaluations when improvements stall while allowing temporary loss fluctuations. In particular, the non-monotonic Kimi trajectories show why stopping at the first loss increase, or adopting the very short budget suggested by Qwen alone, could miss a later improve-

ment. We therefore retain the best configuration encountered and reset the patience counter whenever a lower loss is found, rather than returning the last candidate or stopping at a single increase. Across the seven recorded runs, the early emergence of low-loss configurations and the subsequent fluctuations support the practical utility and design rationale of this mechanism. They do not establish global optimality or imply that $P = 15$ is the optimal patience value.

A.4 Additional Experiments

A.4.1 Rate-Conditioned Sensitivity Granularity

During frontier search, SLaMES selects the sensitivity associated with the current layer–modality skipping-rate band. We compare the full six-band calibration with an endpoint-only variant that retains sensitivities measured at only 12.5% and 100%, while keeping the search procedure and all other components unchanged.

Table 4: Effect of sensitivity-bin granularity on Qwen3-VL-30B-A3B-Instruct. Rate-conditioned search uses six skipping-rate bands, whereas endpoint-only search uses only the 12.5% and 100% sensitivity measurements. Avg. Rel. averages the six benchmark scores relative to the unmodified baseline. Blue rows denote the SLaMES setting.

ρ	Sensitivity strategy	Image Understanding				Video Understanding		Avg. Rel. (%)
		DocVQA	InfoVQA	OCRBench	MMB_{EN}	MVBench	V-MME	
0%	Baseline	93.30	81.03	907	87.11	70.67	67.67	100.00
50%	Rate-conditioned (6 bins)	93.16	80.29	900	87.29	69.80	67.44	**99.47**
	Endpoint-only (2 bins)	92.95	80.37	893	87.11	70.05	67.07	99.25
62.5%	Rate-conditioned (6 bins)	92.90	79.24	891	87.11	70.03	66.70	**98.88**
	Endpoint-only (2 bins)	92.61	78.97	896	86.86	69.47	67.41	98.86
75%	Rate-conditioned (6 bins)	91.93	77.47	888	86.17	69.58	66.78	**98.02**
	Endpoint-only (2 bins)	91.50	76.55	886	86.00	68.92	67.37	97.67
87.5%	Rate-conditioned (6 bins)	90.48	73.66	876	85.57	68.97	65.37	**96.15**
	Endpoint-only (2 bins)	90.26	73.49	871	85.57	68.17	65.63	95.86

The six-band calibration improves average relative accuracy at every budget by 0.02–0.35 percentage points, with the largest gain at 75% skipping. This consistent advantage shows that the ordering of candidate layers benefits from sensitivities matched to the current operating rate rather than from only the two endpoint measurements.

A.4.2 Conditional Top-1 Protection Thresholds

We compare the layer-wise Rank-1 mean used by SLaMES with fixed protection thresholds and a variant without Top-1 protection. The layer-wise means are computed from routing scores already recorded during offline calibration, so protection requires no additional model forward pass. We report realized expert skipping for both models to distinguish accuracy gains from additional computation retained by more permissive thresholds.

Table 5: Conditional Top-1 protection on Qwen3-VL-30B-A3B-Instruct at $\rho = 87.5\%$. The layer-wise strategy uses $\mu_l^{(1)}$ from Section 4.3; all other thresholds are shared across 48 layers. Avg. Rel. averages six relative benchmark scores; Actual Skip is the realized expert-assignment skipping ratio. The blue row denotes SLaMES.

Protection threshold	Image Understanding				Video Understanding		Avg. Rel. (%)	Actual Skip (%)
	DocVQA	InfoVQA	OCRBench	MMB_{EN}	MVBench	V-MME		
Layer-wise mean $\mu_l^{(1)}$ (ours)	90.48	73.66	876	85.57	68.97	65.37	96.15	89.8
Fixed threshold 0.1	90.86	74.60	876	85.22	69.20	65.37	96.40	83.2
Fixed threshold 0.3	88.54	70.57	850	84.79	68.05	64.48	94.10	92.4
No protection	86.57	65.78	829	83.25	67.67	63.74	91.81	92.6

Table 6: Conditional Top-1 protection on Kimi-VL-16B-A3B. Fixed thresholds are shared across all layers. Avg. Rel. averages seven relative benchmark scores. Actual Skip is the realized expert-assignment skipping ratio. The blue row denotes SLaMES.

Protection threshold	Image Understanding							Avg. Rel. (%)	Actual Skip (%)
	ChartQA	DocVQA	InfoVQA	OCRBench	MMB_{EN}	MMB_{CN}	RealWorldQA		
Layer-wise mean $\mu_l^{(1)}$ (ours)	84.72	97.64	77.37	862	78.09	79.98	69.67	97.68	71.4
Fixed threshold 0.1	84.56	97.61	78.18	861	78.87	80.07	69.28	97.84	66.0
Fixed threshold 0.4	83.68	97.26	76.16	834	78.09	79.47	69.93	96.75	72.9
No protection	80.68	96.75	73.92	820	78.61	78.87	69.67	95.51	74.3

Without protection, Qwen3-VL drops from 96.15% to 91.81% Avg. Rel. while realized skipping rises only from 89.8% to 92.6%. A fixed 0.1 threshold slightly improves accuracy on Qwen3-VL and Kimi-VL, but reduces skipping by 6.6 and 5.4 percentage points, respectively, relative to the layer-wise mean. Stricter fixed thresholds recover skipping at an accuracy cost. A shared numerical cutoff cannot account for model- and layer-dependent Top-1 score distributions; the route-derived means adapt to each model and layer, yielding a favorable trade-off without manually tuning a protection cutoff per model.

A.5 Realized Layer-Wise Skipping at Inference

We measure realized layer-wise skipping in vLLM using the searched thresholds for Qwen3-VL-30B-A3B and Kimi-VL-16B-A3B on the same five image datasets. The target ratios are $\rho^* \in \{50, 62.5, 75, 87.5\}\%$ for Qwen3-VL-30B-A3B and $\rho^* \in \{50, 66.7, 83.4\}\%$ for Kimi-VL-16B-A3B. Each ratio is the number of skipped routed-expert assignments divided by total assignments for that layer and modality.

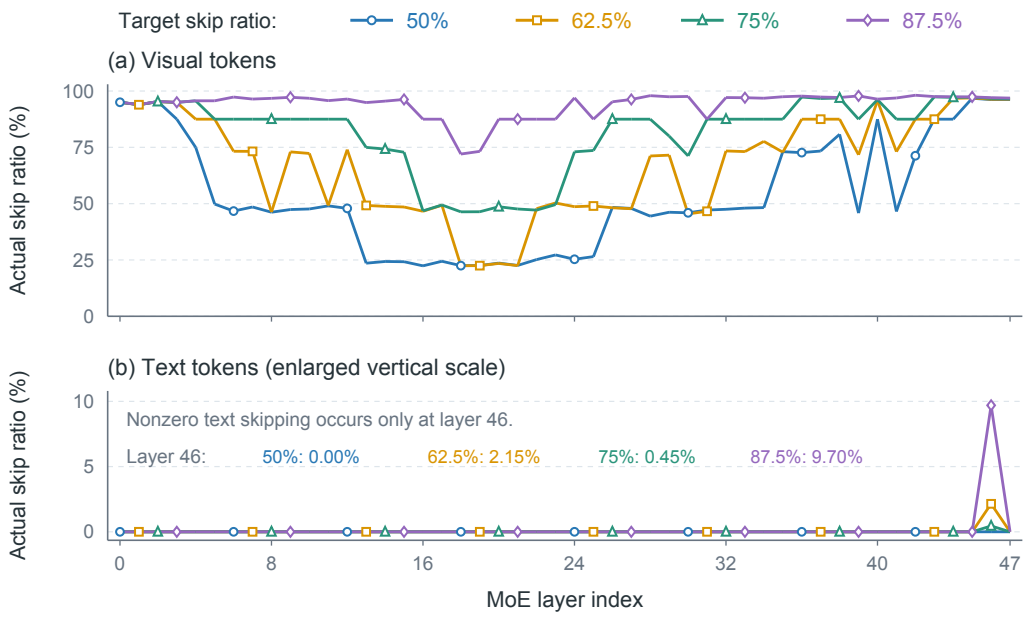


Figure 7: Realized layer-wise skipping on Qwen3-VL-30B-A3B across five image datasets in vLLM. Colors and markers denote target ratios. The two panels use different vertical scales; layer indices are zero-based.

For Qwen3-VL-30B-A3B (Figure 7), low visual skipping concentrates around layers 16–24, consistent with their higher sensitivity in Figure 1(b). This matches our search rule: visual thresholds are preferentially lowered in highly sensitive layers to restore computation, whereas text thresholds are raised in less sensitive layers to increase skipping. Even at the 87.5% target, a visual skipping trough remains near layers 18–19. When nonzero, text skipping is confined to layer 46. The deployed thresholds therefore preserve more computation in sensitive visual layers and nearly all text

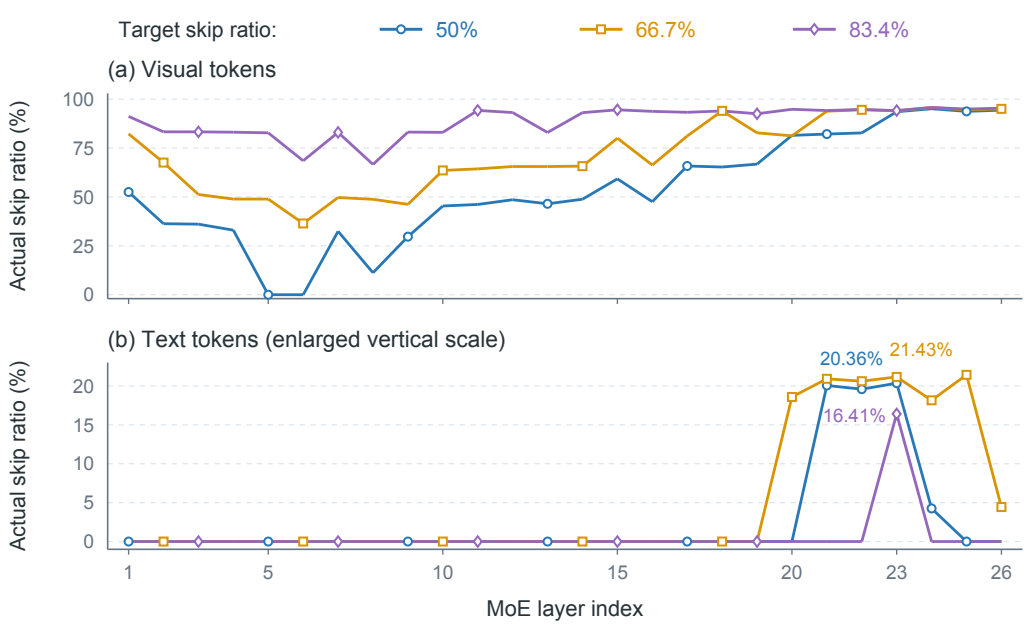


Figure 8: Realized layer-wise skipping on Kimi-VL-16B-A3B across five image datasets in vLLM. Colors and markers denote target ratios. The two panels use different vertical scales.

expert assignments. Sparse text skipping does not, by itself, imply that text-side threshold search is redundant: at high target ratios, sensitivity guides the search toward thresholds in text layers that are more tolerant of skipping, aiming to limit the impact on accuracy.

Kimi-VL-16B-A3B (Figure 8) exhibits a different allocation across layers and modalities. Text experts are also skipped in selected later layers, with peak per-layer rates of 20.36%, 21.43%, and 16.41% at the 50%, 66.7%, and 83.4% targets, respectively. Text skipping is lower at the 83.4% target than at 66.7%, illustrating that a higher global target need not increase skipping in every modality or layer. Together, the two models show that the deployed allocation is not a fixed visual-only skipping policy. Their inference behavior is consistent with the sensitivity-guided allocation underlying our design, while the precise distribution depends on the model, target ratio, and evaluated workload.

A.6 DeepEP Latency Model: Derivation and Measurement Details

This appendix expands the single-layer model in Section 5.5. We distinguish expert assignments from DeepEP token copies, account for rank-level load imbalance, and describe baseline-only cost calibration and speedup prediction. All predictions refer to one FusedMoE layer; end-to-end prefill latency is measured separately in Section 5.5.

A.6.1 Configuration, Notation, and Timing Scope

The profiling configuration uses Qwen3-VL-30B-A3B under EP=4 with vLLM 0.23.0 and the DeepEP high-throughput backend; DropRouter is disabled. The layer has $E = 128$ routed experts, Top-K routing with $K = 8$, hidden size $H = 2048$, and expert intermediate size $D = 768$. Experts are evenly sharded over $R = 4$ ranks. Following the main text, n_l denotes the input token count per rank under balanced input partitioning. We write $N_l = Rn_l$ for the global token count in the EP group and let $\mathcal{I}_l$ index these tokens. Thus, the 16K-token setting in Figure 5a has $N_l = 16{,}384$ and $n_l = 4{,}096$.

The timed layer consists of

$$T_l^{\mathrm{SLaMES}} = T_{\mathrm{router},l} + T_{\mathrm{sel},l} + T_{\mathrm{dispatch},l} + T_{\mathrm{expert},l} + T_{\mathrm{combine},l} + T_{\mathrm{res},l}. \tag{11}$$

Router includes gating, Top-K selection, and expert-ID mapping; $T_{\mathrm{sel},l}$ denotes the additional skipping decision. Permutation and restoration are incorporated into dispatch, expert, and combine

kernels, while communication startup is included in dispatch and combine. Residual time covers operator dispatch, workspace management, stream handoff, and other orchestration. These operations are not counted again as separate permutation or all-to-all terms. We collect router time, stage intercepts, and residual overhead in $T_{\mathrm{fix},l}(n_l)$, which is approximately invariant to skipping at a fixed token count.

A.6.2 Expert Assignments and Token Copies

For token t, $\mathcal{E}_{l,t}^{\mathrm{keep}}$ is the retained expert set defined in the main text. The baseline assignment count, retained count, and realized skipping ratio are

$$A_l^0 = N_l K, \qquad A_l = \sum_{t \in \mathcal{I}_l} |\mathcal{E}_{l,t}^{\mathrm{keep}}|, \qquad p_l = 1 - \frac{A_l}{A_l^0}. \tag{12}$$

The realized layerwise ratio p_l aggregates visual and text assignments; it is distinct from the global target ρ^* used in threshold search. If $A_{l,q}$ is the assignment count executed on rank q, then

$$\sum_{q=1}^{R} A_{l,q} = N_l K(1-p_l), \qquad \overline{A}_l = n_l K(1-p_l). \tag{13}$$

A SwiGLU expert requires approximately $6HD$ FLOPs per assignment. With a_{exp} denoting the effective time per assignment on one rank, the balanced variable compute cost is

$$T_{\mathrm{exp,bal},l} = a_{\mathrm{exp}} n_l K(1-p_l) = c_{\mathrm{exp},l} n_l (1-p_l), \qquad c_{\mathrm{exp},l} = K a_{\mathrm{exp}}. \tag{14}$$

The effective cost is calibrated from timings rather than peak hardware FLOP/s and excludes the expert-stage intercept. We suppress the layer index on the auxiliary unit costs a_{exp} and a_{comm}.

DeepEP (Zhao et al., 2025) shares one activation copy among a token's retained experts on the same destination rank. Define

$$D_{l,t} = \{\mathrm{rank}(e) : e \in \mathcal{E}_{l,t}^{\mathrm{keep}}\}, \qquad d_{l,t} = |D_{l,t}|. \tag{15}$$

The mean destination count and global token-copy count are

$$\bar{d}_l(p_l) = \frac{1}{N_l} \sum_{t \in \mathcal{I}_l} d_{l,t}, \qquad C_l = N_l \bar{d}_l(p_l). \tag{16}$$

A copy disappears only when the token's last retained expert on that rank is removed. Consequently, the retained communication ratio is

$$g_l(p_l) = \frac{\bar{d}_l(p_l)}{\bar{d}_l(0)} = \frac{\sum_t |D_{l,t}|}{\sum_t |D_{l,t}^0|}, \tag{17}$$

which generally differs from $1 - p_l$. Let a_{comm} denote the effective dispatch-plus-combine cost per copy on one rank. Then

$$\begin{aligned} T_{\mathrm{comm,bal},l} &= a_{\mathrm{comm}} n_l \bar{d}_l(p_l) = c_{\mathrm{comm},l} n_l g_l(p_l), \\ c_{\mathrm{comm},l} &= a_{\mathrm{comm}} \bar{d}_l(0). \end{aligned} \tag{18}$$

This is an effective token-copy workload model, rather than an exact count of inter-GPU bytes; metadata and local-copy costs enter the measured coefficients and intercepts.

Analytical reference for random skipping. For uniformly sampled Top-K experts, the number J on a specified rank has distribution

$$\Pr(J=j) = \frac{\binom{E/R}{j}\binom{E-E/R}{K-j}}{\binom{E}{K}}. \tag{19}$$

If each selected assignment is independently skipped with probability p, all j assignments on that rank disappear with probability p^j. Hence

$$\begin{aligned} \bar{d}_{\mathrm{rand}}(p) &= R\left[1 - \sum_{j=0}^{K} \Pr(J=j) p^j\right], \\ g_{\mathrm{rand}}(p) &= \bar{d}_{\mathrm{rand}}(p) / \bar{d}_{\mathrm{rand}}(0). \end{aligned} \tag{20}$$

Table 7: Zero-skipping FusedMoE profiling under EP=4. N_l is the global token count and $n_l = N_l/4$ is the per-rank count.

N_l	n_l	Total	Router	Dispatch	Expert	Combine
1,024	256	1.2822	0.1734	0.3246	0.4336	0.1277
2,048	512	1.2681	0.1365	0.3485	0.4498	0.2199
4,096	1,024	1.4898	0.1319	0.4652	0.5056	0.3043
8,192	2,048	1.8271	0.1523	0.6354	0.5870	0.3719
16,384	4,096	2.9037	0.1433	1.0183	1.0573	0.5847
32,768	8,192	5.1622	0.1515	1.8050	1.9762	1.1488

At $p = 0$, the sum is interpreted as $\Pr(J = 0)$. For the profiled configuration, $\bar{d}_{\mathrm{rand}}(0) \approx 3.629$; at $p = 0.7$, $\bar{d}_{\mathrm{rand}}(p) \approx 1.866$ and $g_{\mathrm{rand}}(p) \approx 0.514$. Thus, retaining 30% of assignments still retains about 51.4% of copies under this reference model. SLaMES uses structured, score-dependent skipping, so its g_l is computed from retained routing patterns rather than assumed to equal g_{rand}.

A.6.3 Rank-Level Imbalance and Single-Layer Speedup

Let $C_{l,q} = \sum_t \mathbb{I}[q \in D_{l,t}]$ be the copies processed on rank q. Assignment and copy counts must be weighted by their respective costs before being combined:

$$W_{l,q}(p_l) = a_{\mathrm{exp}} A_{l,q}(p_l) + a_{\mathrm{comm}} C_{l,q}(p_l). \tag{21}$$

We define the unified load-imbalance coefficient as

$$\beta_l(p_l) = \frac{\max_q W_{l,q}(p_l)}{R^{-1} \sum_q W_{l,q}(p_l)} \geq 1. \tag{22}$$

It equals one under perfect balance. Baseline and retained routing can have different critical ranks, so $\beta_l(p_l)$ need not equal $\beta_l(0)$. When no assignments remain, the variable workload is zero and its imbalance factor can be set to one by convention. Combining the balanced costs gives

$$\begin{aligned} T_l^{\mathrm{SLaMES}}(n_l, p_l) \approx\ & T_{\mathrm{fix},l}(n_l) + T_{\mathrm{sel},l}(n_l, p_l) \\ & + \beta_l(p_l)\big[(1 - p_l) c_{\mathrm{exp},l} + g_l(p_l) c_{\mathrm{comm},l}\big] n_l. \end{aligned} \tag{23}$$

Here, $g_l(p_l)$ and $\beta_l(p_l)$ are the explicit-argument forms of g_l and β_l in the main text, evaluated for a given retained routing pattern; neither is determined by p_l alone. This recovers Equation 10. The unified coefficient is a first-order approximation: if compute and communication have different critical ranks, a stage-wise alternative is

$$T_{\mathrm{var},l} \approx a_{\mathrm{exp}} \max_q A_{l,q} + a_{\mathrm{comm}} \max_q C_{l,q}. \tag{24}$$

An assignment-only max-to-mean ratio can diagnose compute imbalance, but cannot replace Equation 22 when communication matters.

The unmodified layer has no skipping-selection overhead:

$$T_l^0(n_l) \approx T_{\mathrm{fix},l}(n_l) + \beta_l(0)(c_{\mathrm{exp},l} + c_{\mathrm{comm},l}) n_l. \tag{25}$$

Its predicted speedup is

$$\Gamma_l(n_l, p_l) = \frac{T_l^0(n_l)}{T_{\mathrm{fix},l}(n_l) + T_{\mathrm{sel},l}(n_l, p_l) + \beta_l(p_l)[(1 - p_l) c_{\mathrm{exp},l} + g_l(p_l) c_{\mathrm{comm},l}] n_l}. \tag{26}$$

The dependence on both g_l and β_l explains why equal skipping ratios can yield different speedups.

A.6.4 Zero-Skipping Calibration and Validation

We calibrate latency cost coefficients using only $p_l = 0$ measurements. Table 7 reports the baseline profiling data; all times are in milliseconds. The complete operator is timed separately from its stages, and the residual is not listed as a separate column.

For $N_l \geq 4{,}096$, the baseline enters an approximately linear high-throughput regime. Fitting the stage costs against global assignment and copy counts gives the following reference parameterization:

$$\begin{aligned} T^0_{\text{expert},l} &\approx 0.2169 + 6.6295 \times 10^{-6}(8N_l), \\ T^0_{\text{dispatch},l} &\approx 0.2598 + 1.2938 \times 10^{-5}(3.629N_l), \\ T^0_{\text{combine},l} &\approx 0.1410 + 8.2777 \times 10^{-6}(3.629N_l). \end{aligned} \tag{27}$$

Here, $3.629N_l$ is the uniform-routing reference copy count. Combining the stage intercepts with router and residual overhead yields

$$T^0_l(N_l/R) \approx 0.8490 + \left(5.3036 \times 10^{-5} + 7.6993 \times 10^{-5}\right)N_l. \tag{28}$$

A direct fit to total latency gives $0.8408 + 1.3053 \times 10^{-4}N_l$, close to the stage-based fit. These numerical slopes use *global* token counts; conversion to the main text's per-rank convention multiplies each slope by $R = 4$.

Under the balanced random-skipping approximation, the corresponding reference prediction is

$$\begin{aligned} T_{\text{rand},l}(N_l/R, p) \approx\ & 0.8490 + 5.3036 \times 10^{-5}N_l(1-p) \\ & + 7.6993 \times 10^{-5}N_l g_{\text{rand}}(p) + T_{\text{sel},l}(N_l/R, p). \end{aligned} \tag{29}$$

The balanced reference fit and the imbalance-aware formulation must not count the same baseline imbalance twice. For the latter, per-rank assignment and copy traces supply the workloads used to estimate effective unit costs and $\beta_l(0)$; measured baseline slopes that already include imbalance cannot simply be multiplied by $\beta_l(0)$ again.

For SLaMES, retained assignment and token-copy counts determine $g_l(p_l)$ and $\beta_l(p_l)$ without using skipped-run latency labels. Selection overhead is profiled separately, rather than absorbed by refitting the compute or communication coefficients. The $p_l > 0$ full-layer timings are reserved for validation, as in Figure 5a. Profiling uses the same warm-up, synchronization, and aggregation procedures for baseline and skipped runs. Workload statistics and latency should be paired per iteration before aggregating predictions, so that the comparison uses a consistent timing statistic.

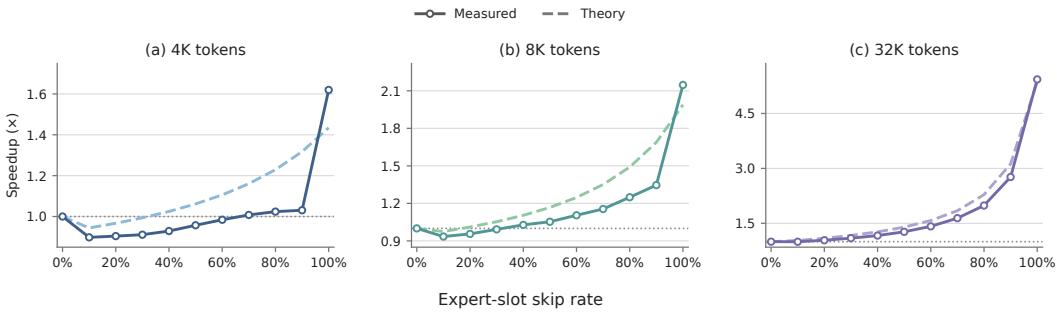


Figure 9: Additional single-layer speedup measurements on Qwen3-VL-30B-A3B under EP=4 with global token counts of (a) 4K, (b) 8K, and (c) 32K. Solid lines with markers show measured speedup relative to the unmodified layer; dashed lines show theoretical predictions using cost coefficients calibrated only at zero skipping.

Figure 9 extends the 16K-token comparison in the main text to 4K, 8K, and 32K tokens. Larger token counts yield greater speedups, consistent with amortizing fixed overheads over more variable work. The theoretical curves capture the overall trend, while the measured curves show more pronounced plateaus and endpoint changes, especially at 4K and 8K. These differences illustrate the limits of the continuous approximation at small retained workloads, discussed below in Section A.6.6.

A.6.5 Choosing a Skipping Budget from a Speedup Target

After hardware cost calibration, a desired single-layer speedup $\Gamma^\star$ can guide the skipping budget before SLaMES threshold calibration. For a specified token count and candidate routing assumptions, evaluate Equation 26 over feasible ratios and select

$$p_l^\star = \min\{p \in \mathcal{P}_l : \widehat{\Gamma}_l(n_l, p) \geq \Gamma^\star\}, \tag{30}$$

where $\mathcal{P}_l$ is the candidate set. A balanced random-skipping curve provides an initial estimate when retained routes are unavailable; candidate masks applied to recorded routing patterns can instead supply policy-dependent g_l and β_l. Thus, this inversion is conditional on the assumed routing and balance, not determined by the skipping ratio alone. The estimate guides the subsequent accuracy-constrained threshold search and should be checked against the learned policy. If the feasible set is empty, the target cannot be attained under the assumed workload and model. It does not imply the same speedup for the entire model.

A.6.6 Implementation and Applicability

SLaMES implements expert skipping through DeepEP's invalid expert-ID convention. After cumulative-mass selection and conditional Top-1 protection, we set skipped entries of `expert_ids` to -1 and pass this tensor as DeepEP's `topk_idx`. The official DeepEP repository (Zhao et al., 2025) documents -1 as indicating no expert selection. Its layout kernel excludes these entries from expert counts and destination-rank membership, while its masking test also sets the corresponding routing weights to zero.[1] The following Python snippet illustrates our masking and dispatch sequence using the high-throughput buffer API; `keep_mask` is the Boolean SLaMES decision with the same shape as the Top-K routing tensors, and `num_experts` denotes the total expert count E.

```
import deep_ep

# x, router_ids, router_weights, and buffer already exist.
# keep_mask includes SLaMES conditional Top-1 protection.
expert_ids = router_ids.to(deep_ep.topk_idx_t).clone()
expert_ids.masked_fill_(~keep_mask, -1)
expert_weights = router_weights.float().clone()
expert_weights.masked_fill_(expert_ids < 0, 0.0)

# Build the layout from the masked routes.
(n_rank, n_rdma, n_expert, in_rank, _) = (
    buffer.get_dispatch_layout(expert_ids, num_experts)
)
dispatched = buffer.dispatch(
    x,
    topk_idx=expert_ids,
    topk_weights=expert_weights,
    num_tokens_per_rank=n_rank,
    num_tokens_per_rdma_rank=n_rdma,
    num_tokens_per_expert=n_expert,
    is_token_in_rank=in_rank,
)
```

The layout must be constructed after masking so that communication reflects the retained routes. A token–rank copy is removed only when all of that token's expert IDs on the destination rank have been invalidated, matching Equation 17. Downstream expert packing and execution exclude invalid assignments, and output reduction uses the masked routing weights. DeepEP provides the dispatch/combine mechanism; the expert-compute path must also honor the invalid-ID convention to avoid executing skipped assignments. Profiling records per-rank assignment and copy counts alongside selection, dispatch, expert, combine, and full-operator timings.

A.7 Expert Load Balance on Qwen3.5-122B-A10B

We examine whether SLaMES causes substantial rank-level load imbalance on Qwen3.5-122B-A10B at the 87.5% target skipping setting. Following Equation 22, we compare the maximum rank workload to the mean rank workload in each layer. The workload combines expert assignments and token copies as defined in Equation 21.

Figure 10 shows that across 48 layers, the mean β_l rises only from 1.163 to 1.200. The increase exceeds 0.1 in just four layers, and the maximum β_l under SLaMES is 1.508, compared with 1.531

[1] Source references at commit `a56d615`: layout API, expert counting, rank membership, and masking example. These links document the API mechanism, not the deployment version.

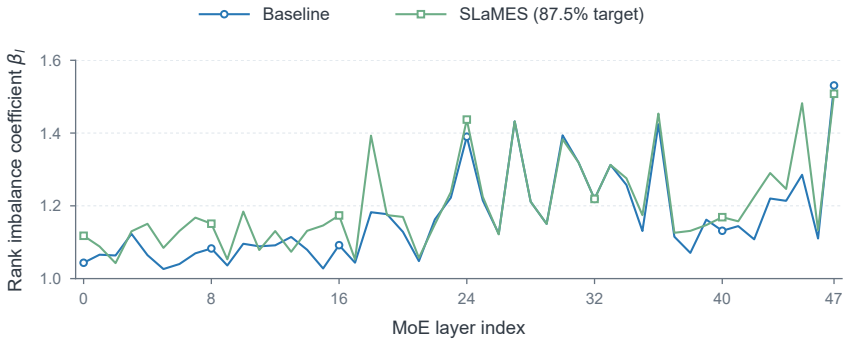


Figure 10: Layer-wise rank load-imbalance coefficient β_l for Qwen3.5-122B-A10B under the 87.5% target setting. SLaMES causes only a modest increase on average, without a sustained rise across layers. Layer indices are zero-based.

for the baseline. Thus, while the skipping decisions introduce some layer-specific imbalance, they do not produce sustained or severe concentration of workload on a few ranks. The modest balance cost is acceptable in view of the substantial reduction in expert assignments.

A.8 Limitations

All experiments use the same 1,024-example GQA source for calibration. While our evaluation spans three MoE MLLMs and both image and video tasks, we have not systematically varied the calibration data or covered every model family and deployment workload. Our results therefore characterize performance under the evaluated calibration protocol; the effect of alternative calibration sources and sample sizes on threshold quality remains to be studied. Extending the evaluation to additional multimodal domains is another direction for future work.

A.9 Related Work Supplement

Token pruning in MLLMs Visual-token removal mitigates redundancy by discarding tokens before or within the language model. VisionZip (Yang et al., 2025) removes redundant tokens before language-model inference, FastV (Chen et al., 2024) prunes them after shallow layers, and SparseVLM (Zhang et al., 2025) performs text-conditioned, layer-adaptive sparsification. Recent variants retain diverse tokens (Alvar et al., 2025) or allocate sparsity according to layer-wise attention (He et al., 2025). This eliminates later attention and feed-forward computation for discarded tokens. Because each decision removes an entire token, however, this remains the coarsest granularity of computation skipping: once removed, a token cannot participate in subsequent layers, and its potential later contribution is irreversibly lost.

Expert Pruning and Skipping in MoE LLMs Expert-level efficiency methods for MoE LLMs broadly fall into static compression and dynamic skipping. Static approaches include domain-specific pruning guided by few-shot demonstrations (Dong et al., 2026), gradient-free search for compact expert configurations (Liu et al., 2024a), and one-shot within-expert pruning from weight, activation, and router signals (Xie et al., 2024). DiEP (Bai et al., 2025) learns non-uniform layer-wise ratios, while MoNE (Zhang et al., 2026) replaces redundant experts with lightweight constant modules. Their compressed configurations are fixed and cannot adapt computation across inputs or tokens. Although these approaches reduce retained capacity, their realized latency still depends on the resulting routing pattern and system support. Dynamic methods instead vary the active set at inference time: NAEE (Lu et al., 2024) supports task-agnostic and task-specific post-training skipping, while MC-MoE (Huang et al., 2025) combines mixed-precision quantization with online pruning that protects important tokens and suppresses low-significance expert activations. These methods preserve an adjustable, token-dependent compute–accuracy trade-off, but were primarily evaluated on text-only MoE LLMs and leave modality-dependent routing behavior unaddressed.

Adaptive Expert Allocation in MoE MLLMs Adaptive expert allocation has also been studied in MoE MLLMs. MoDES (Huang et al., 2026) introduces training-free skipping using layer importance and separate visual and textual thresholds. Its global threshold per modality and skip-rate-invariant importance, however, do not jointly capture sensitivity across layers, modalities, and skip rates or the routing-mass shape of individual tokens. Nevertheless, it demonstrates the value of modality-aware post-training allocation for multimodal sequences. MACS (Li et al., 2026) addresses the complementary expert-parallel straggler problem through load estimation and modality-adaptive capacity scaling rather than directly reducing token–expert assignments under an accuracy budget. AnyExperts (Gao et al., 2026) learns a token-importance estimator for variable expert-slot allocation, whereas MoE-GRPO (Ko et al., 2026) optimizes selection through reinforcement learning and modality-aware guidance; both require updating the router or model. SLaMES instead requires no additional training: it calibrates rate-dependent layer–modality sensitivity, searches per-layer and per-modality thresholds under a target compute budget, and adapts retained experts to each token through cumulative routing mass. It neither permanently removes experts nor modifies the pretrained router, retaining an adjustable compute–accuracy trade-off at deployment.

A.10 Sensitivity-Guided Alternating Frontier Search

Algorithm 1 details the search in Section 4.2. Here, g_l^m indexes the current threshold for layer l and modality m, and $\rho(g_l^m \leftarrow g)$ denotes the lookup-estimated global skipping ratio after replacing that single index by g.

Algorithm 1 Sensitivity-guided alternating frontier search.

Require: Lookup table $\mathcal{U}$, threshold grid $\mathcal{T}$, target ρ_{tar}
Require: Calibration set $\mathcal{D}$, early-stop patience P
Ensure: Per-layer, per-modality thresholds $\boldsymbol{\tau}^\star$

$g_l^{\mathrm{visual}} \leftarrow G$, $g_l^{\mathrm{text}} \leftarrow 1$ for all $l \in \mathcal{L}$
$\ell^\star \leftarrow +\infty$, $c \leftarrow 0$
while $c < P$ **do**
  $\mathcal{F}_{\mathrm{vis}} \leftarrow \{l : g_l^{\mathrm{visual}} > 1,\ \rho(g_l^{\mathrm{visual}} \leftarrow g_l^{\mathrm{visual}} - 1) \geq \rho_{\mathrm{tar}}\}$
  while $\mathcal{F}_{\mathrm{vis}} \neq \emptyset$ **do**
    $l^\star_{\mathrm{vis}} \leftarrow \arg\max_{l \in \mathcal{F}_{\mathrm{vis}}} S_{l,\mathrm{visual}}(r_{l,\mathrm{visual}})$
    Decrease $g_{l^\star_{\mathrm{vis}}}^{\mathrm{visual}}$ while feasible, stopping at a new sensitivity bin
    Update $\mathcal{U}$ state and $\mathcal{F}_{\mathrm{vis}}$
  end while
  $\boldsymbol{\tau} \leftarrow \{\tau^{(g_l^{\mathrm{text}})}, \tau^{(g_l^{\mathrm{visual}})}\}_{l=1}^{L}$
  $\ell \leftarrow \mathcal{L}_{\mathrm{KL}}(\boldsymbol{\tau}; \mathcal{D})$
  if $\ell < \ell^\star$ **then**
    $(\ell^\star, \boldsymbol{\tau}^\star) \leftarrow (\ell, \boldsymbol{\tau})$; $c \leftarrow 0$
  else
    $c \leftarrow c + 1$
  end if
  if $c = P$ **then**
    break
  end if
  $\mathcal{F}_{\mathrm{text}} \leftarrow \{l : g_l^{\mathrm{text}} < G\}$
  if $\mathcal{F}_{\mathrm{text}} = \emptyset$ **then**
    break
  end if
  $l^\dagger_{\mathrm{text}} \leftarrow \arg\min_{l \in \mathcal{F}_{\mathrm{text}}} S_{l,\mathrm{text}}(r_{l,\mathrm{text}})$
  $g_{l^\dagger_{\mathrm{text}}}^{\mathrm{text}} \leftarrow g_{l^\dagger_{\mathrm{text}}}^{\mathrm{text}} + 1$
end while
return $\boldsymbol{\tau}^\star$

A.11 THE USE OF LARGE LANGUAGE MODELS

We used a large language model (LLM) solely as a general-purpose writing assistance tool. Specifically, the LLM was employed to check spelling, correct grammatical errors, and improve the clarity and style of the manuscript text. The LLM did not contribute to the conception of the research ideas, the design of the methodology, the execution of experiments, or the analysis and interpretation of results. All scientific content, claims, and conclusions are solely those of the authors.