MAP-MoE: Modality-Aware Structured Channel Pruning for Multimodal Mixture-of-Experts Models

Anonymous authors
Paper under double-blind review

Abstract

Sparse Mixture-of-Experts (MoE) models improve performance by scaling up while activating only a few experts for each token. This scaling advantage requires all experts to be stored during deployment, leading to substantial storage and GPU memory costs. Existing solutions reduce these costs by pruning experts, merging experts, or pruning expert feed-forward network (FFN) channels. However, balancing model accuracy and inference speed remains challenging. We therefore introduce MAP-MoE, a training-free framework that combines modality-aware expert FFN channel pruning with our Ragged-Hybrid GPU executor. Using a small unlabeled calibration set, MAP-MoE groups experts into visual, mixed, and textual groups and assigns different pruning rates to them under a shared global budget. Within each MoE layer, MAP-MoE ranks FFN channels across experts in the same modality group and retains only the selected rows of the FFN gate and up projections and the corresponding columns of the down projection. Experts retain different numbers of channels after pruning, making standard execution much slower. Ragged-Hybrid addresses this with custom Triton kernels for small token batches and grouped GEMM for larger inputs. Our main evaluation covers two multimodal MoE models at FFN channel pruning ratios of 20%, 30%, and 50% across 10 benchmarks. At 50% pruning, MAP-MoE improves average accuracy by 1.34 points over the SOTA baseline (CAMERA-P) on Qwen3-VL while retaining 98.5% of the original Qwen3-VL average accuracy. For batched generation on one A800, our framework achieves up to $1.65\times$ end-to-end generation speedup over the original model and reduces peak allocated GPU memory by up to 46%.

1 Introduction

Recent vision–language models have adopted sparse Mixture-of-Experts (MoE) layers to scale up (Bai et al., 2025; Wang et al., 2025; Lin et al., 2024). An MoE layer contains many experts but routes each token to only a few of them (Shazeer et al., 2017; Fedus et al., 2022; Lepikhin et al., 2021). This limits the computation used by each token. All expert weights, however, remain part of the deployed model and lead to high storage and GPU memory costs. Expert compression is a direct way to reduce these costs.

Existing compression methods prune complete experts, merge similar experts, or remove FFN channels (Lasby et al., 2026; Jha et al., 2026; Xu et al., 2026). Pruning methods use different importance measures to rank experts or channels for removal. REAP uses router-weighted expert output norms, CAMERA combines channel activations with weight norms, and NAEE compares the reconstruction error of candidate expert subsets (Lasby et al., 2026; Xu et al., 2026; Lu et al., 2024). These measures capture different aspects of importance, but the resulting pruning rules do not assign separate budgets to expert groups with different modality preferences.

The model inserts visual-encoder outputs at image-token positions in the language sequence; we call these positions visual tokens and the remaining calibration positions textual tokens. Visual and textual tokens can differ in both the experts they use and the activation strength they produce (Dutt et al., 2026; Huang et al., 2026). We measure these differences at the expert level and infer visual, mixed, and textual groups. We then compare task accuracy after removing the same number of FFN

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.

channels from the visual or textual group. The observed difference motivates separate group budgets under a fixed total budget. Sec. 3 presents the measurements, and Sec. 5.2 evaluates the grouping and allocation methods.

Based on these observations, we introduce MAP-MoE, a training-free framework for modality-aware FFN channel pruning. It includes Ragged-Hybrid as the GPU execution engine for the resulting pruned experts. Our contributions are:

- We propose modality-aware grouping and budget allocation for FFN channel pruning. We group experts by how visual and textual tokens use them, then assign a separate pruning rate to each group. We select channels jointly across experts in the same group and MoE layer, leaving the router unchanged. In controlled Qwen-P50 tests (50% FFN channel pruning), inferred groups improve average accuracy by 2.51 points over random groups with the same layer-wise sizes; unequal group rates improve it by 1.87 points over uniform rates.
- We introduce the rarity-weighted energy (RWE) score to select channels within each group. It combines squared router-weighted activations, variation in their absolute values, and down-projection weight norms. With groups and budgets fixed, RWE improves P50 average accuracy over an activation-and-weight score by 0.17/0.40 points on Qwen/InternVL.
- We design Ragged-Hybrid to accelerate the pruned model when experts retain different numbers of FFN channels. It uses custom Triton kernels for small token batches entering an MoE layer and grouped GEMM for larger inputs. For the same pruned model, Ragged-Hybrid is 2.35–$5.28\times$ faster than the standard PyTorch path on the tested Qwen-P50 workloads.
- We evaluate Qwen3-VL-30B-A3B and InternVL3.5-30B-A3B at 20%, 30%, and 50% FFN channel pruning across 10 multimodal benchmarks. At 50%, MAP-MoE exceeds CAMERA-P by 1.34/2.65 points on Qwen/InternVL and retains 98.5%/96.1% of the average accuracy of the original models. In our MolmoE-P50 comparison, it gains 4.97 points over DERN (SOTA) and 7.55 points over CAMERA-P. For Qwen batch generation at the same pruning ratio, our system is 1.12–$1.65\times$ faster and uses 44–46% less peak allocated GPU memory than the original model.

2 Related Work

Sparse multimodal MoEs. Sparse MoE layers replace standard FFNs with multiple experts and activate only a few for each token (Shazeer et al., 2017; Fedus et al., 2022). MoE-LLaVA, Qwen3-VL, and InternVL3.5 extend this architecture to vision–language models, where visual and textual tokens share an MoE language backbone (Lin et al., 2024; Bai et al., 2025; Wang et al., 2025). Studies of these models identify experts used mainly by one modality and experts shared by both (Dutt et al., 2026). These differences also guide compression methods: MoDES skips experts during inference, while MODE and VEQ use modality information in quantization (Huang et al., 2026; Chen et al., 2026; Qin et al., 2026).

Expert merging. To reduce expert storage, expert merging combines the weights of several experts. Existing methods differ mainly in how they group experts and merge their weights. REAM uses router-weighted activation statistics to guide both steps (Jha et al., 2026). Sub-MoE groups experts by output similarity, uses joint singular value decomposition to obtain a shared subspace, and merges expert-specific components within that subspace (Li et al., 2025b). This approach preserves information from several experts instead of discarding the removed weights outright. Its decisions are nevertheless made at expert or expert-group level: the merge determines which expert functions share one parameter set, but it does not directly control which FFN channels remain inside each expert. Merging can also require router changes or weight reconstruction so that tokens can be routed to the merged experts.

Expert pruning. A more aggressive strategy is to remove complete experts. NAEE selects expert subsets that minimize layer-output reconstruction error (Lu et al., 2024). REAP measures importance through router-weighted expert output norms and removes low-scoring experts (Lasby et al., 2026). Shapley-MoE estimates the contribution of each expert with sampled Shapley values and prunes less important experts (Huang et al., 2025). These methods reduce storage while leaving the

retained experts intact. Weight reconstruction may reduce the resulting loss, but it cannot selectively preserve useful channels inside an expert that has been removed.

FFN channel pruning. For finer control, FFN channel pruning removes selected channels without deleting whole experts. CAMERA-P treats a gate row, an up row, and the corresponding down column as one micro-expert, and ranks these units across experts in each MoE layer (Xu et al., 2026). TENP keeps important experts intact and applies different channel pruning rates to other experts across layers (He et al., 2026). FlexMoE ranks channels within each expert and learns a retained-channel ratio for each expert to produce nested subnetworks, with custom kernels for execution (Mo et al., 2026). Rather than exporting one fixed pruned model, μ-MoE adapts structured channel selection to each prompt at test time (Koike-Akino et al., 2025). Channel pruning supports finer budget control than deleting complete experts and can preserve the original routing choices. It can also leave different experts with different channel counts, so physical parameter reduction does not automatically produce faster inference. The pruning rule and the execution path must therefore be evaluated separately.

Visual token reduction. Another way to reduce computation is to remove redundant visual tokens. TopV and ToDRE prune visual tokens during inference, while VisionZip selects tokens before they enter the language model (Yang et al., 2025a; Li et al., 2025a; Yang et al., 2025b). For multimodal MoEs, FastMMoE combines fewer expert activations per visual token with token pruning based on routing similarity (Xia et al., 2025). These methods can speed up inference by processing fewer visual tokens, but they leave model parameters unchanged. Their benefit depends on how many visual tokens can be removed and may be limited on text-heavy inputs.

3 MOTIVATION

We motivate MAP-MoE with two observations: visual and textual tokens use experts differently, and channel pruning needs a suitable GPU implementation to deliver actual speedups.

3.1 MODALITY DIFFERENCES GUIDE GROUP PRUNING BUDGETS

Prior analyses have found that visual and textual tokens use MoE experts differently (Dutt et al., 2026; Huang et al., 2026). We measure this difference from the routed FFN activations in our calibration set. In MoE layer l, expert e has hidden dimension d, intermediate dimension m, gate/up weights $W^g, W^u \in \mathbb{R}^{m\times d}$, and down weights $W^d \in \mathbb{R}^{d\times m}$. One FFN channel comprises a row from each gate/up matrix and the matching down-projection column.

We run calibration inputs through the original model. For token t routed to expert e, let $\mathbf{h}_t \in \mathbb{R}^d$ be the layer input and $p_{t,e}$ the router weight. We record the router-weighted activations before the down projection:

$$\mathbf{z}_{l,e,t} = p_{t,e}\left[\mathrm{SiLU}(W^g_{l,e}\mathbf{h}_t) \odot (W^u_{l,e}\mathbf{h}_t)\right]. \tag{1}$$

For each expert, $A^v_{l,e}$ and $A^t_{l,e}$ sum $\|\mathbf{z}_{l,e,t}\|_2^2$ over routed visual and textual tokens, respectively. The corresponding route counts for the whole layer are N^v_l and N^t_l. We divide by these counts so that the different numbers of visual and textual routes do not alone determine the expert label. The visual preference score is

$$\bar{A}^v_{l,e} = \frac{A^v_{l,e}}{N^v_l + \epsilon}, \quad \bar{A}^t_{l,e} = \frac{A^t_{l,e}}{N^t_l + \epsilon}, \qquad b_{l,e} = \frac{\bar{A}^v_{l,e}}{\bar{A}^v_{l,e} + \bar{A}^t_{l,e} + \epsilon}. \tag{2}$$

The score $b_{l,e}$ measures the visual share of route-normalized activation energy for that expert. With thresholds $\tau_t < \tau_v$, we label experts with $b < \tau_t$ textual, those with $b > \tau_v$ visual, and the rest mixed. Experts with the same label form a modality group. Appendix C gives the thresholds, ϵ, and the rule for missing modality observations. Applying this rule to Qwen assigns 2,713 of 6,144 layer–expert pairs (44.2%) to the visual group. Figure 1(a) shows their distribution across MoE layers.

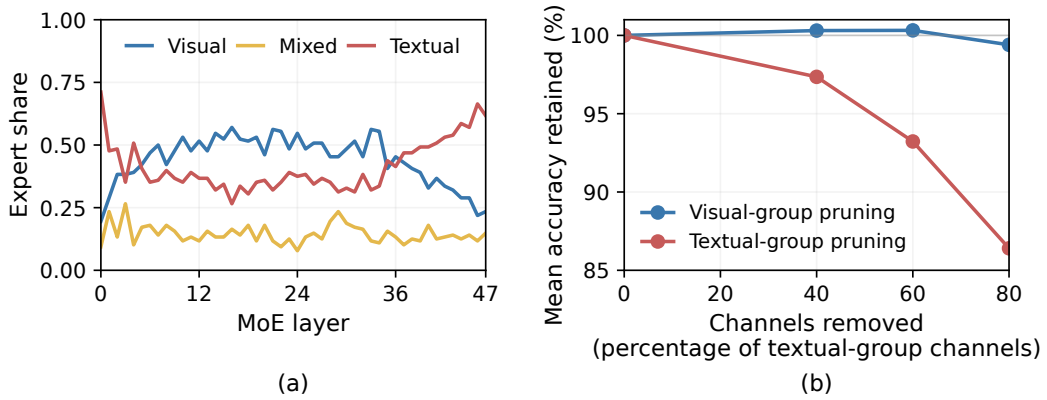


Figure 1: Expert modality groups and accuracy after FFN channel pruning in Qwen3-VL. (a) Share of visual, mixed, and textual experts by MoE layer. (b) Average accuracy retained across ten tasks after removing the same number of FFN channels from the visual or textual group. The x-axis expresses that common count as a percentage of textual-group channels.

Visual experts account for 44.2% of the experts in Qwen, so pruning them could substantially reduce model size. We test how channel removal affects accuracy in the visual and textual groups. Using RWE to rank channels, we remove the same number of channels from the visual or textual group and leave the other groups unchanged. At the largest tested removal count, the ten-task mean accuracy is 0.768 after pruning the visual group, close to 0.773 for the original model, but falls to 0.668 after pruning the textual group. Figure 1(b) shows this pattern across the tested counts. This comparison reveals that visual-group channels are less sensitive to removal in this setting. The result motivates separate group pruning rates under a fixed total budget; Sec. 5.2 tests grouping and allocation in the full method.

3.2 Efficient Inference After Channel Pruning

Existing channel-pruning methods such as CAMERA-P can leave different experts with different numbers of FFN channels. This reduces stored weights and computation but changes the matrix shapes used during inference. Padding experts back to the original FFN dimension cannot use the potential computation savings, while launching many small matrix multiplications adds overhead. Pruned models therefore need kernels that execute the retained channels efficiently.

We test P50 models from MAP-MoE and CAMERA-P with the standard PyTorch path on Qwen, using 8 requests per batch on one A800. CAMERA-P reaches only 0.255–$0.419\times$ the end-to-end generation speed of the original model implemented in the Hugging Face Transformers library; MAP-MoE reaches 0.307–$0.475\times$ on the same workloads (Table 7). Both are slower despite removing half of the FFN channels. Ragged-Hybrid addresses this execution cost with custom Triton kernels for small token batches and grouped GEMM for larger inputs. Sec. 5.3 compares both paths on the same pruned model.

4 MAP-MoE

MAP-MoE uses a small unlabeled calibration set to group experts by modality preference and assign different pruning rates to the groups. Within each MoE layer, it ranks FFN channels across experts in the same group. It retains the selected rows of the gate and up projections and the corresponding columns of the down projection, then runs the pruned experts with Ragged-Hybrid. The router remains unchanged. Let P denote the global fraction of expert FFN channels removed. The reported P20, P30, and P50 settings correspond to $P = 0.2$, 0.3, and 0.5.

4.1 Modality-Aware Budget Allocation

Across all MoE layers, let q_g be the proportion of experts assigned to group $g \in \{v, m, t\}$, and let r_g be its pruning rate. We determine one set of rates (r_v, r_m, r_t) for the whole model and apply

Table 1: Group pruning rates at each budget, rounded to 0.001.

Model	Budget	Visual	Mixed	Text
Qwen3-VL	P20	0.374	0.241	0
	P30	0.545	0.409	0
	P50	0.800	0.600	0.144
InternVL3.5	P20	0.446	0.388	0
	P30	0.661	0.600	0
	P50	0.800	0.600	0.292

each r_g to the corresponding group in every MoE layer (Sec. 4.2). For each layer and group, we rank channels by their summed squared activations and measure the proportion of activation energy removed from that group when the lowest-ranked r proportion of channels is pruned. $\mathcal{E}_g(r)$ averages this value across MoE layers. The upper bound u_g limits the proportion of group-g channels that can be removed.

Allocation has two steps. First, we choose the smallest textual pruning rate allowed by the global budget and the group limits: $r_t^* = [P - q_v u_v - q_m u_m]_+ / q_t$, where $[x]_+ = \max(x, 0)$. Second, with $r_t = r_t^*$ fixed, we choose the visual and mixed rates by solving

$$\min_{r_v, r_m} \sum_g q_g \mathcal{E}_g(r_g) \quad \text{s.t.} \quad \sum_g q_g r_g = P, \quad 0 \le r_g \le u_g, \quad r_t = r_t^*. \tag{3}$$

We assign more pruning to groups for which $\mathcal{E}_g(r)$ increases more slowly. Once a group reaches its maximum rate, the remaining budget shifts to the other groups. At P50, both models reach the visual and mixed group limits, and the textual group takes the remaining budget.

Appendix B.4 gives the group limits, curve construction, and solver. Table 1 lists the resulting group rates.

4.2 Selecting and Pruning FFN Channels

Once the group rates are fixed, we choose which channels to retain. For expert (l, e), channel j receives the rarity-weighted energy (RWE) score

$$S_{l,e,j} = A_{l,e,j} C_{l,e,j}^{\gamma} \|W_{l,e,:,j}^d\|_2^2, \qquad A_{l,e,j} = \sum_t z_{l,e,t,j}^2. \tag{4}$$

Here $A_{l,e,j}$ sums squared activations over all calibration tokens routed to that expert. The squared norm of down-projection column j weights this quantity by how strongly the channel contributes to the expert output. The factor $C_{l,e,j}$ measures relative variation in activation magnitude, and γ controls its effect on the score.

Let $N_{l,e}$ be the number of calibration tokens routed to expert (l, e) and $\widetilde{N} = \max(N_{l,e}, 1)$. For each channel, $\mu = \widetilde{N}^{-1} \sum_t |z_{l,e,t,j}|$ is its mean absolute activation and $q = \widetilde{N}^{-1} \sum_t z_{l,e,t,j}^2$ its mean squared activation. We compute

$$C_{l,e,j} = \mathrm{clip}_{[\epsilon,1]} \left(\sqrt{\frac{2}{\pi}} \frac{\sqrt{\max(q - \mu^2, 0)}}{\max(\mu, \epsilon)} \right). \tag{5}$$

The factor C compares variation in activation magnitude with its mean and limits the result to $[\epsilon, 1]$. Channels with greater relative variation receive a larger C, a higher RWE score, and are therefore more likely to be retained.

The allocation in Sec. 4.1 uses squared activations to set the group rates; RWE then adds output weights and activation variation to select channels within each group. For each layer l and modality group g, Modality-Group Top-k ranks channels across all experts in that group. If the group contains $M_{l,g}$ channels, it removes exactly $\mathrm{round}(r_g M_{l,g})$ lowest-scoring channels. Each expert can therefore retain a different number of channels, even though the group rates are shared across layers. We store only the selected gate/up rows and matching down columns. Appendix B.4 gives the rounding and equal-score rules.

4.3 RAGGED-HYBRID

Ragged-Hybrid stores the retained gate/up rows and matching down columns contiguously, with a channel count and offset for each expert, and runs them without padding back to the original FFN dimension. Channel counts are aligned to multiples of 16 for tensor-core execution. Based on our kernel benchmarks, we use custom Triton kernels when at most 64 tokens enter an MoE layer across the batch, before Top-k routing. For these small inputs, the kernels read expert IDs from the router. The routed expert output is

$$W^d_{l,e}\mathbf{z}_{l,e,t} = p_{t,e}W^d_{l,e}\big[\mathrm{SiLU}(W^g_{l,e}\mathbf{h}_t) \odot (W^u_{l,e}\mathbf{h}_t)\big].$$

The kernels sum these outputs without regrouping tokens by expert. For more than 64 tokens, Ragged-Hybrid groups routes by expert and applies grouped GEMM to experts with the same aligned channel count. Appendix A explains the 64-token switch, alignment overhead, numerical checks, and implementation details.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Models. We conduct the full evaluation on Qwen3-VL-30B-A3B-Instruct and InternVL3.5-30B-A3B. Both models contain 48 MoE layers with 128 experts per layer, Top-8 routing, a hidden size of 2048, and an expert FFN dimension of 768; their vision encoders differ. We further evaluate P50 on MolmoE-1B-0924 as an extension to a different MoE configuration with 16 MoE layers and 64 experts per layer.

Calibration data. The calibration image pool contains 64 Flickr8k training images and 192 training images from the OCRVQA, FigureQA, DVQA, CLEVR, and IconQA configurations of The Cauldron (Hodosh et al., 2013; HuggingFaceM4, 2024). These form four equal categories: natural images, OCR, charts, and visual reasoning. Each calibration draw takes 32 images from each category and pairs them with 128 WikiText-2 training segments of 256 tokens each (Merity et al., 2017). Appendix C gives the number of draws used for each model, dataset revisions, prompts, and sample details.

Evaluation datasets. We evaluate ten benchmarks: OK-VQA, ScienceQA, TextVQA, GQA, POPE, MMBench, AI2D, RealWorldQA, ChartQA, and MMVP (Marino et al., 2019; Lu et al., 2022; Singh et al., 2019; Hudson & Manning, 2019; Li et al., 2023; Liu et al., 2023; Kembhavi et al., 2016; xAI, 2024; Masry et al., 2022; Tong et al., 2024). Together, they cover knowledge-intensive VQA, OCR and chart understanding, perception, hallucination evaluation, and visual reasoning. We average their scores with equal weight, using pair accuracy for MMVP. All evaluations use greedy decoding with one beam.

Compared methods. REAP removes experts using router-weighted output norms (Lasby et al., 2026); DERN applies its published scoring and channel-segment recombination (Zhou et al., 2025). Both use our multimodal calibration. We adapt standalone CAMERA-P directly to the fused VLM MoE modules. At every budget, it starts from the original checkpoint and uses $\alpha = 1$ in its published score. After pruning one layer, CAMERA-P recalibrates the next layer on the output of the pruned prefix. It then selects from one layer-wide micro-expert pool without modality labels (Xu et al., 2026). Within each model, CAMERA-P and MAP-MoE use the same ordered calibration inputs. Whole-expert methods keep 103/90/64 experts per layer for P20/P30/P50; MAP-MoE removes the target fraction of FFN channels, subject only to integer rounding. All methods share evaluation examples, preprocessing, prompts, and decoding settings. InternVL DERN-P50 collapsed during generation, so its incomplete cells are reported without an average accuracy.

Efficiency setup. On one A800 80GB, we test the Qwen3-VL P50 accuracy mask under M1/M2/M3: eight requests per batch with input/output token counts 1024/16, 512/64, and 128/128. Both runs use identical encoded inputs and BF16 weights; the original uses the Hugging Face Transformers library, the pruned model Ragged-Hybrid. We report medians from three processes with

Table 2: Per-task accuracy and equal-weight 10-task averages. MMVP reports individual/pair accuracy and contributes pair accuracy to the average. Scores are rounded to 0.001; bold marks the best structured method using unrounded scores per budget; InternVL DERN-P50 collapsed and is not averaged.

Model	Budget	Method	OKV	SQA	TQA	GQA	POPE	MMB	AI2D	RWQA	CQA	MMVP	Avg
Qwen3-VL		Original (0% ref)	0.502	0.953	0.838	0.625	0.901	0.871	0.860	0.719	0.860	0.800/0.600	0.773
	P20	REAP	0.468	0.947	0.837	0.623	0.901	0.857	0.850	0.723	**0.876**	0.800/0.600	0.768
		DERN	0.493	0.952	0.829	0.622	0.902	0.862	0.856	0.723	0.875	0.800/0.600	0.771
		CAMERA-P	**0.526**	0.953	0.831	**0.627**	0.902	0.868	0.854	0.712	0.849	**0.817/0.633**	0.776
		MAP-MoE (ours)	0.510	**0.956**	**0.837**	0.626	**0.903**	**0.875**	**0.858**	**0.725**	0.870	0.803/0.607	**0.777**
	P30	REAP	0.437	0.937	0.826	0.619	0.898	0.839	0.833	0.719	**0.874**	0.807/0.613	0.759
		DERN	0.472	0.947	0.824	0.608	0.900	0.852	0.844	**0.731**	0.866	**0.820/0.640**	0.768
		CAMERA-P	**0.512**	0.948	0.827	0.624	0.903	0.854	0.849	0.720	0.830	0.807/0.613	0.768
		MAP-MoE (ours)	0.496	**0.952**	**0.834**	**0.625**	**0.906**	**0.860**	**0.854**	0.720	0.870	0.813/0.627	**0.774**
	P50	REAP	0.372	0.896	0.804	0.595	0.885	0.823	0.805	0.715	**0.852**	0.810/0.620	0.737
		DERN	0.308	0.902	0.774	0.573	0.869	0.805	0.795	**0.719**	0.833	0.803/0.613	0.719
		CAMERA-P	0.402	**0.927**	0.796	0.608	**0.900**	**0.833**	0.826	0.710	0.834	**0.810/0.640**	0.748
		MAP-MoE (ours)	**0.521**	0.912	**0.817**	**0.616**	0.899	**0.833**	**0.833**	0.709	0.850	0.803/0.620	**0.761**
InternVL3.5		Original (0% ref)	0.521	0.993	0.783	0.632	0.880	0.859	0.856	0.711	0.881	0.820/0.647	0.776
	P20	REAP	0.482	0.989	0.783	0.630	0.888	0.845	0.845	0.702	0.882	**0.823/0.647**	0.769
		DERN	0.455	0.982	0.768	0.628	0.878	0.834	0.821	0.692	0.785	0.787/0.587	0.743
		CAMERA-P	0.512	0.992	**0.783**	**0.638**	0.886	0.857	0.850	0.707	**0.884**	0.813/0.640	0.775
		MAP-MoE (ours)	**0.520**	**0.992**	0.783	0.632	**0.889**	**0.858**	**0.853**	**0.711**	0.878	0.813/0.633	**0.775**
	P30	REAP	0.449	0.990	0.775	0.624	0.886	0.839	0.838	0.701	**0.889**	0.807/0.620	0.761
		DERN	0.419	0.960	0.753	0.617	0.881	0.793	0.792	0.658	0.726	0.757/0.513	0.711
		CAMERA-P	0.502	**0.991**	**0.780**	**0.636**	**0.894**	**0.851**	**0.853**	**0.709**	0.882	0.813/0.633	**0.773**
		MAP-MoE (ours)	**0.509**	0.982	0.773	0.631	0.892	0.845	0.850	0.705	0.868	**0.820/0.647**	0.770
	P50	REAP	**0.425**	0.960	0.755	0.601	0.887	0.808	0.821	**0.695**	0.788	**0.810/0.627**	0.737
		DERN	-	0.000	-	0.000	0.000	-	0.001	0.000	0.000	0.063/0.000	–
		CAMERA-P	0.244	**0.985**	0.701	0.571	0.885	**0.837**	0.836	0.684	**0.840**	0.800/0.613	0.720
		MAP-MoE (ours)	0.390	0.972	**0.766**	**0.626**	**0.892**	0.827	**0.840**	0.694	0.833	0.803/0.620	**0.746**

three warmups and five synchronized runs each. Appendix A gives software versions, timing boundaries, and the peak-memory definition.

5.2 Accuracy Results

MAP-MoE settings. We use grouping thresholds $(\tau_t, \tau_v) = (0.4, 0.6)$, group pruning limits $(u_v, u_m, u_t) = (0.8, 0.6, 1)$, and RWE exponent $\gamma = 2$ for every model and budget. In local Qwen sweeps, tested threshold pairs and γ values change the ten-task average by at most 0.48 and 0.17 points, respectively. Shifting both pruning limits by ± 0.05 changes it by +0.28 or -0.95 points. Appendix Tables 18, 13, and 15 give the full results.

Table 2 reports all per-task and average results for Qwen and InternVL. MAP-MoE exceeds REAP in all six model–budget settings. On Qwen, its improvement over CAMERA-P grows from 0.12 points at P20 to 0.62 at P30 and 1.34 at P50. At P50, MAP-MoE retains 98.5% and 96.1% of the average accuracy of the original Qwen and InternVL models, respectively, and improves over CAMERA-P by 1.34 and 2.65 points. Appendix B.2 analyzes how these gains vary across tasks.

We compute paired confidence intervals using 2,000 resamples within each task. For every bootstrap replicate, we recompute the equal-weight 10-task mean difference. The resampling unit is a circular-question group for MMBench, a question pair for MMVP, and one example otherwise. The Qwen/InternVL P50 gains over CAMERA-P have 95% confidence intervals of [0.68, 1.99] and [2.03, 3.28] percentage points; the 0.92-point InternVL gain over REAP has an interval of [0.31, 1.57] percentage points. Differences and intervals use unrounded per-example scores; Table 2 rounds displayed values to 0.001. These intervals condition on fixed calibration and masks. Across three independent single-draw Qwen-P50 calibration-to-mask runs, the average accuracy is 0.759 ± 0.003 (mean $\pm$ standard deviation) (Appendix B.5).

Table 3: Qwen-P20 selection control with fixed calibration and RWE scores.

Setting	OKV	SQA	TQA	GQA	POPE	MMB	AI2D	RWQA	CQA	MMVP	Avg.
Per-expert	0.496	0.938	0.818	0.616	0.907	0.854	0.842	0.723	0.858	0.607	0.766
Layer-global	0.492	0.957	0.835	0.622	0.901	0.869	0.854	0.719	0.864	0.607	0.772
MAP-MoE	0.510	0.956	0.837	0.626	0.903	0.875	0.858	0.725	0.870	0.607	**0.777**

Controlled ablations. Tables 3 and 4 test selection at Qwen-P20 and allocation at P50. MAP-MoE gains 0.49 points over layer-global Top-k at P20, where both grouping and group rates change. With groups and scores fixed, its P50 rates gain 1.87/0.29 points over uniform rates on Qwen/InternVL. At P50, the pruning limits and textual protection set the rates; at P20/P30, Eq. (3) uses $\mathcal{E}_g(r)$ to select rates within those limits. The combined use of modality grouping, unequal group pruning rates, and channel selection improves P50 accuracy.

Table 4: P50 allocation control.

Model	Rates	OKV	SQA	TQA	GQA	POPE	MMB	AI2D	RWQA	CQA	MMVP	Avg.
Qwen3-VL	Uniform	0.373	0.931	0.800	0.612	0.902	0.850	0.831	0.702	0.809	0.613	0.742
	MAP-MoE	0.521	0.912	0.817	0.616	0.899	0.833	0.833	0.709	0.850	0.620	0.761
InternVL3.5	Uniform	0.385	0.973	0.764	0.630	0.894	0.833	0.832	0.697	0.831	0.593	0.743
	MAP-MoE	0.390	0.972	0.766	0.626	0.892	0.827	0.840	0.694	0.833	0.620	0.746

Two further P50 controls test labels and scores separately. With per-layer group sizes, group rates, and RWE fixed, inferred Qwen groups reach 0.761, compared with 0.736 ± 0.005 for three random label permutations. With groups, rates, and retained counts fixed, RWE gives 0.761/0.746 on Qwen/InternVL, compared with 0.759/0.742 for base energy. A Qwen-P20 sweep over $\gamma \in \{0.5, 1, 2, 4\}$ spans 0.17 points. Appendix B.5 reports the complete per-task controls. These experiments change different factors, so their gains are not additive.

Table 5: MolmoE P50 per-task accuracy and equal-weight 10-task averages. MMVP uses pair accuracy. Scores are rounded to 0.001; bold uses unrounded scores to mark the best pruned model per task; the original model is a reference.

Method	**OKV**	**SQA**	**TQA**	**GQA**	**POPE**	**MMB**	**AI2D**	**RWQA**	**CQA**	**MMVP**	**Avg**
Original (0% ref)	0.648	0.899	0.780	0.536	0.897	0.632	0.723	0.600	0.781	0.663	0.716
REAP	0.291	0.336	0.316	0.082	0.772	0.078	0.247	0.392	0.297	0.280	0.309
DERN	0.430	0.761	0.640	0.304	0.804	0.489	**0.600**	0.490	**0.656**	0.540	0.571
CAMERA-P	0.480	0.511	0.659	0.445	0.847	0.345	0.497	0.507	0.585	0.580	0.546
MAP-MoE (ours)	**0.517**	**0.768**	**0.701**	**0.482**	**0.884**	**0.502**	0.583	**0.556**	0.636	**0.583**	**0.621**

Extension to MolmoE. We also test P50 on MolmoE-1B-0924 (Deitke et al., 2024), which has 16 MoE layers and 64 experts per layer, compared with 48 layers and 128 experts in our primary models. Across the same ten tasks, MAP-MoE reaches 0.621 average accuracy, compared with 0.546 for CAMERA-P and 0.571 for DERN. MAP-MoE gains 4.97 points over DERN (SOTA) and 7.55 points over CAMERA-P, supporting the high-pruning result on a different MoE configuration. Table 5 reports every task; Appendix B.1 gives the model-specific settings.

5.3 Inference Efficiency Results

Latency and memory. We report end-to-end generation time (E2E), time to first token (TTFT), and mean time per subsequent output token (TPOT). A speedup is original-model time divided by pruned-model time. At P50, Table 6 reports E2E generation speedups of 1.12–$1.65\times$ and a 44–46% reduction in peak allocated memory. During prefill, Ragged-Hybrid groups routes and runs GEMMs for experts with different retained channel counts, which may contribute to the slower TTFT. The 1.51–$1.76\times$ faster decoding is the main source of the end-to-end speedup in these batched workloads. These figures measure the combined effect of physical pruning and Ragged-Hybrid relative to the original model run with the Hugging Face Transformers library. Alignment to multiples of 16 adds 1.70% to the logically retained channel count and still reduces stored expert FFN weights by 49.15%.

Table 6: Qwen3-VL P50 efficiency over static workloads M1–M3 with eight requests per batch. Ratios are original-model time divided by pruned-model time; values below one indicate slower execution. Peak memory reports the reduction from the original model.

Input	Workload	E2E	TTFT	TPOT	Peak memory ↓
Multimodal	M1	1.124×	0.827×	1.508×	44.5%
	M2	1.419×	0.729×	1.598×	45.1%
	M3	1.579×	0.628×	1.679×	45.7%
Text	M1	1.117×	0.753×	1.737×	44.4%
	M2	1.519×	0.653×	1.753×	45.1%
	M3	1.654×	0.488×	1.760×	46.1%

Effect of the GPU executor. To measure the executor effect on an already-pruned model, we run each P50 pruned model through standard PyTorch grouped GEMM and Ragged-Hybrid while holding channels and weights fixed. The standard path uses `torch.nn.functional.grouped_mm`, sorts routes by expert, and groups experts with similar channel counts. Ragged-Hybrid uses custom routed-expert kernels for at most 64 input tokens and grouped GEMM for more than 64. For the same MAP-MoE weights, Ragged-Hybrid runs 2.35–5.28× faster than the standard PyTorch path. Compared with the original model, the E2E generation speedup increases from 0.307–0.475× with PyTorch to 1.113–1.637× with Ragged-Hybrid. The CAMERA-P mask gives a corresponding 2.64–6.56× executor gain. Table 7 reports every workload and shows that, among the two tested paths, only Ragged-Hybrid makes the pruned model faster than the original model. The same-mask control and Table 6 use separate timing runs, which explains the small differences between their medians. Appendix A gives absolute latency, timing definitions, alignment details, and numerical checks.

Table 7: Qwen3-VL P50 executor control. Values are E2E generation speedups, computed as the latency of the standard Hugging Face Transformers implementation divided by measured latency under identical inputs. For each pruning method, both execution paths retain identical tensors.

		Multimodal			Text		
Pruned mask	Execution path	M1	M2	M3	M1	M2	M3
Original	Hugging Face Transformers	1.000×	1.000×	1.000×	1.000×	1.000×	1.000×
MAP-MoE	Standard PyTorch (no custom kernel)	0.475×	0.337×	0.307×	0.472×	0.330×	0.310×
MAP-MoE	Ragged-Hybrid (ours)	1.117×	1.407×	1.563×	1.113×	1.502×	1.637×
CAMERA-P	Standard PyTorch (no custom kernel)	0.419×	0.313×	0.268×	0.370×	0.263×	0.255×
CAMERA-P	Ragged-Hybrid (ours)	1.106×	1.397×	1.562×	1.073×	1.495×	1.673×

Kernel-switch analysis. We compare the custom kernel with grouped GEMM at 64–128 flattened input tokens, using the retained-channel patterns from P20/P30/P50 and three calibration-derived routing distributions. At 64 tokens, the custom kernel gives a 1.128× geometric-mean speedup across the nine cases, with a lowest-case ratio of 0.994×. At 80 tokens, the geometric mean falls to 0.971× and the lowest ratio to 0.876×. This crossover supports the shared 64-token threshold used in all end-to-end measurements; Appendix Table 8 reports the complete sweep.

6 Conclusion

MAP-MoE uses separate calibration statistics for visual and textual tokens to assign FFN channel pruning rates, then removes the selected channels while preserving the router. Across our two primary 30B-A3B models, it exceeds REAP at all three budgets and improves over CAMERA-P at P50. A P50 extension to MolmoE also improves over CAMERA-P. Ragged-Hybrid runs the pruned Qwen-P50 model faster and with less GPU memory in our batch-generation tests on one A800. Together, the accuracy and runtime results show that modality-aware channel allocation can reduce stored expert parameters while retaining 96.1–98.5% of the average accuracy of the original models at P50.

AI Use Statement

Generative AI tools assisted with experiment planning, code development, literature search, analysis, drafting, translation, and language editing. The authors reviewed and revised AI-assisted code and text, checked cited sources and experimental artifacts, and take responsibility for the final paper.

Reproducibility Statement

The main text defines the calibration statistics, allocation problem, channel score, and execution path. The appendix gives the numerical solver, dataset and decoding settings, complete controlled results, timing protocol, numerical checks, and artifact digests. The accompanying code regenerates the reported masks and audits their budgets and tensor shapes.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Yuanteng Chen, Nanxin Zeng, Peisong Wang, Zhilei Liu, Yuantian Shao, Shiqiang Lang, Tao Liu, Chuangyi Li, Qinghao Hu, Gang Li, Jing Liu, and Jian Cheng. MODE: Modality-Decomposed Expert-Level Mixed-Precision Quantization for MoE Multimodal LLMs. *arXiv preprint arXiv:2606.17118*, 2026. URL https://arxiv.org/abs/2606.17118.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, et al. Molmo and PixMo: Open weights and open data for state-of-the-art vision-language models. *arXiv preprint arXiv:2409.17146*, 2024. URL https://arxiv.org/abs/2409.17146.

Raman Dutt, Harleen Hanspal, Guoxuan Xia, Petru-Daniel Tudosiu, Alexander Black, Yongxin Yang, Steven McDonagh, and Sarah Parisot. Exploiting mixture-of-experts redundancy unlocks multimodal generative abilities. In *ICLR 2026 Workshop on Multimodal Intelligence*, 2026.

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

Jiangyang He, Shaolin Zhu, and Deyi Xiong. TENP: Trapezoidal Expert Neuron Pruning For Mixture-of-Experts. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 20911–20925, 2026. doi: 10.18653/v1/2026.findings-acl.1049. URL https://aclanthology.org/2026.findings-acl.1049/.

Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013. doi: 10.1613/jair.3994. URL https://jair.org/index.php/jair/article/view/10833.

Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Fei Chao, Rongrong Ji, and Liujuan Cao. Discovering Important Experts for Mixture-of-Experts Models Pruning Through a Theoretical Perspective. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c66a9db149261435664284a20b6f1d42-Abstract-Conference.html.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2511.15690.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019.

HuggingFaceM4. The Cauldron, 2024. URL https://huggingface.co/datasets/HuggingFaceM4/the_cauldron.

Saurav Jha, Maryam Hashemzadeh, Ali Saheb Pasand, Ali Parviz, Min-Joong Lee, and Boris Knyazev. REAM: Merging improves pruning of experts in LLMs (router-weighted expert activation merging). *arXiv preprint arXiv:2604.04356*, 2026.

Aniruddha Kembhavi, Mike Salvato, Ellen Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *ECCV*, 2016.

Toshiaki Koike-Akino, Jing Liu, and Ye Wang. μ-MoE: Test-time pruning as micro-grained mixture-of-experts. *arXiv preprint arXiv:2505.18451*, 2025.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. REAP the Experts: Why Pruning Prevails for One-Shot MoE Compression. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ukGxWd2aDG.

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. In *International Conference on Learning Representations (ICLR)*, 2021.

Duo Li, Zuhao Yang, Xiaoqin Zhang, Ling Shao, and Shijian Lu. ToDRE: Effective Visual Token Pruning via Token Diversity and Task Relevance. *arXiv preprint arXiv:2505.18757*, 2025a. URL https://arxiv.org/abs/2505.18757.

Lujun Li, Qiyuan Zhu, Jiacheng Wang, Wei Li, Hao Gu, Sirui Han, and Yike Guo. Sub-MoE: Efficient mixture-of-expert LLM compression via subspace expert merging. *arXiv preprint arXiv:2506.23266*, 2025b. URL https://arxiv.org/abs/2506.23266.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023.

Bin Lin, Zhenyu Tang, Yang Ye, Jinfa Huang, Junwu Zhang, Yatian Pang, Peng Jin, Munan Ning, Jiebo Luo, and Li Yuan. MoE-LLaVA: Mixture of Experts for Large Vision-Language Models. *arXiv preprint arXiv:2401.15947*, 2024. URL https://arxiv.org/abs/2401.15947.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*, 2022.

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, 2024. doi: 10.18653/v1/2024.acl-long.334. URL https://aclanthology.org/2024.acl-long.334/.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3195–3204, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Marino_OK-VQA_A_Visual_Question_Answering_Benchmark_Requiring_External_Knowledge_CVPR_2019_paper.html.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of ACL*, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/pdf?id=Byj72udxe.

Fan Mo, Yuxuan Han, Geng Zhang, Wangbo Zhao, and Yang You. FlexMoE: One-for-All Nested Intra-Expert Pruning for MoE Language Models. *arXiv preprint arXiv:2606.27866*, 2026. URL https://arxiv.org/abs/2606.27866.

Guangshuo Qin, Zhiteng Li, Zheng Chen, Weihang Zhang, Linghe Kong, and Yulun Zhang. VEQ: Modality-Adaptive Quantization for MoE Vision-Language Models. *arXiv preprint arXiv:2602.01037*, 2026. URL https://arxiv.org/abs/2602.01037.

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. In *International Conference on Learning Representations (ICLR)*, 2017.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, 2019.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal LLMs. *arXiv preprint arXiv:2401.06209*, 2024.

Weiyun Wang et al. InternVL3.5: Advancing Open-Source Multimodal Models in Versatility, Reasoning, and Efficiency. *arXiv preprint arXiv:2508.18265*, 2025. URL https://arxiv.org/abs/2508.18265.

xAI. Realworldqa. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/xai-org/RealworldQA.

Guoyang Xia, Yifeng Ding, Fengfa Li, Lei Ren, Wei Chen, Fangxiang Feng, and Xiaojie Wang. FastMMoE: Accelerating Multimodal Large Language Models through Dynamic Expert Activation and Routing-Aware Token Pruning. *arXiv preprint arXiv:2511.17885*, 2025. URL https://arxiv.org/abs/2511.17885.

Yuzhuang Xu, Xu Han, Yuanchi Zhang, Yixuan Wang, Yijun Liu, Shiyu Ji, Qingfu Zhu, and Wanxiang Che. CAMERA: Multi-Matrix Joint Compression for MoE Models via Micro-Expert Redundancy Analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27395–27404, 2026. doi: 10.1609/aaai.v40i32.39957. URL https://ojs.aaai.org/index.php/AAAI/article/view/39957. Issue 32.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, et al. Topv: Compatible token pruning with inference time optimization for fast and low-memory multimodal vision language model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

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 (CVPR)*, pp. 19792–19802, 2025b. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yang_VisionZip_Longer_is_Better_but_Not_Necessary_in_Vision_Language_CVPR_2025_paper.html.

Yixiao Zhou, Ziyu Zhao, Dongzhou Cheng, Zhiliang Wu, Jie Gui, Yi Yang, Fei Wu, Yu Cheng, and Hehe Fan. Dropping Experts, Recombining Neurons: Retraining-Free Pruning for Sparse Mixture-of-Experts LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 15169–15186. Association for Computational Linguistics, November 2025. doi: 10.18653/v1/2025.findings-emnlp.820. URL https://aclanthology.org/2025.findings-emnlp.820/.

A System and Efficiency Details

Measurement setup. The Qwen3-VL timing runs at P20/P30/P50 use the same pruning masks as their corresponding accuracy evaluations in Table 2. Ragged-Hybrid runs the retained expert weights with grouped GEMM during prefill and custom Triton kernels during decoding (Sec. 4.3). Each run uses one exclusive A800. We report the median of three independent processes for each workload.

M1/M2/M3 are `bs8/seq1024/gen16`, `bs8/seq512/gen64`, and `bs8/seq128/gen128`; each is a static batch that includes prefill and decoding. Here `seq` is the target number of text tokens. The encoded prompt also contains chat-template tokens and, for multimodal inputs, image tokens. We build fixed batches from the frozen training-image pool and WikiText-2 training text, truncating or concatenating text to reach the target. All methods and timing runs use the same encoded batches. Table 9 reports P20/P30/P50 for M1–M3; Table 6 shows the P50 results in the main text.

The CAMERA-P timing controls use the P50 accuracy mask from Table 2, obtained with the same three-draw calibration recipe as MAP-MoE.

When to switch kernels. We compare the custom Triton kernel with grouped GEMM on one A800. The test covers the retained-channel patterns from P20/P30/P50 and the combined, visual-token, and textual-token routing distributions recorded during calibration. We count input tokens across the batch before Top-k routing. At 64 input tokens, the custom kernel has a $1.128\times$ geometric-mean speedup across the nine cases; the lowest ratio is $0.994\times$. At 80 tokens, the geometric mean falls to $0.971\times$ and the lowest ratio to $0.876\times$. We therefore switch to grouped GEMM above 64 input tokens in every reported workload.

Table 8: Custom Triton kernel compared with grouped GEMM on one A800. Each entry divides grouped-GEMM latency by custom-kernel latency; values above one favor the custom kernel. Routing patterns come from the recorded combined, visual-token, and textual-token calibration statistics.

Budget	Routing profile	64	80	96	112	128
P20	Combined	1.095	0.955	0.818	0.722	0.633
	Visual	1.147	0.973	0.833	0.719	0.637
	Textual	0.994	0.876	0.717	0.609	0.573
P30	Combined	1.131	0.983	0.845	0.735	0.643
	Visual	1.204	1.022	0.896	0.753	0.664
	Textual	1.014	0.884	0.727	0.617	0.582
P50	Combined	1.221	1.041	0.895	0.783	0.683
	Visual	1.292	1.086	0.928	0.803	0.708
	Textual	1.084	0.938	0.785	0.664	0.620
Geometric mean		**1.128**	0.971	0.824	0.709	0.637
Lowest case		**0.994**	0.876	0.717	0.609	0.573

Timing procedure. The original Qwen3-VL model runs with the Hugging Face Transformers library in BF16 evaluation mode on one GPU. We use PyTorch 2.11.0+cu128, CUDA 12.8, and Hugging Face Transformers 5.15.0.dev0. Original and pruned runs use the same inputs, static batch sizes, three warmups, and five timed repetitions per workload in each process. We synchronize CUDA around each timed call. Model loading, input construction, and host-to-device transfer occur before timing. TTFT is the mean latency of separate one-token generation calls; we measure E2E in separate calls with both minimum and maximum output length set to G tokens. For $G > 1$, we compute $\mathrm{TPOT} = (\mathrm{E2E} - \mathrm{TTFT})/(G-1)$ within each process, then report medians across processes. Peak memory is `torch.cuda.max_memory_allocated` during the E2E calls, reported in decimal GB. In the executor comparison, retained channel indices and weights stay fixed. The standard PyTorch path uses `torch.nn.functional.grouped_mm`, sorts routes by expert, and groups experts with similar channel counts; it uses no custom kernel. We measure this comparison in separate matched runs, so its medians can differ slightly from Table 6.

Table 9: Qwen3-VL efficiency results for MAP-MoE at P20/P30/P50 over static workloads M1–M3 with eight requests per batch. Speedup is original-model time divided by pruned-model time; values above one indicate faster execution. Each budget uses the pruning mask from its corresponding accuracy evaluation. Values are three-process medians on one A800.

Budget	Input	Workload	E2E	TTFT	TPOT	Peak memory ↓
P20	Multimodal	M1	1.099×	0.832×	1.423×	17.3%
		M2	1.346×	0.751×	1.488×	17.7%
		M3	1.470×	0.648×	1.547×	18.4%
	Text	M1	1.112×	0.777×	1.641×	17.4%
		M2	1.447×	0.681×	1.636×	17.9%
		M3	1.570×	0.528×	1.652×	18.8%
P30	Multimodal	M1	1.103×	0.828×	1.450×	26.2%
		M2	1.367×	0.748×	1.515×	26.8%
		M3	1.498×	0.638×	1.583×	27.5%
	Text	M1	1.108×	0.768×	1.664×	26.3%
		M2	1.478×	0.668×	1.685×	27.0%
		M3	1.603×	0.517×	1.691×	28.0%
P50	Multimodal	M1	1.124×	0.827×	1.508×	44.5%
		M2	1.419×	0.729×	1.598×	45.1%
		M3	1.579×	0.628×	1.679×	45.7%
	Text	M1	1.117×	0.753×	1.737×	44.4%
		M2	1.519×	0.653×	1.753×	45.1%
		M3	1.654×	0.488×	1.760×	46.1%

Table 10: Absolute latency for Qwen3-VL P50 with the same accuracy mask. TTFT and TPOT use ms and ms/token; E2E uses seconds. Values are three-process medians on one A800.

		TTFT		TPOT		E2E	
Input	Load	Original	MAP-MoE	Original	MAP-MoE	Original	MAP-MoE
Multimodal	M1	936.6	1131.9	88.11	58.42	2.2573	2.0083
	M2	598.5	820.5	80.16	50.17	5.6483	3.9815
	M3	372.7	593.2	74.48	44.37	9.8315	6.2283
Text	M1	688.9	914.8	62.14	35.78	1.6220	1.4515
	M2	408.7	625.8	64.75	36.94	4.4878	2.9541
	M3	208.2	426.8	65.44	37.19	8.5192	5.1519

Kernel numerical check. A BF16 numerical check uses FFN input/output dimension $d = 256$ (the evaluated model uses $d = 2048$), FFN channel count $m = 768$, six experts with logical retained-channel counts $[0, 1, 17, 127, 511, 768]$, and Top-2 routing. Against the reference computation, relative RMSE is 3.14×10^{-3} for eight tokens (custom Triton kernel) and 1.39×10^{-4} for 80 tokens (grouped GEMM). The empty expert produces zero output in both cases.

Alignment storage overhead. For an expert retaining $k_{l,e}$ FFN channels, the executor stores $\widetilde{k}_{l,e} = \max(16, 16\lceil k_{l,e}/16 \rceil)$ channels, including a zero tile for an empty expert. Table 11 reports the selected channel count and the extra storage needed for alignment. Each retained FFN channel contributes one gate row, one up row, and one down column, all spanning the unchanged dimension d. The percentage overhead in channel count therefore also applies to expert FFN weight storage. For MAP-MoE the aligned expert FFN weight reduction is 49.15% at P50. Execution cost also depends on routing and kernel tiling; measured latency and peak memory are reported separately.

Table 11: Qwen-P50 alignment audit for the masks in Table 7. Zero-tile entries are included in aligned channel counts and overhead; percentages use the logical retained-channel count as the denominator.

Quantity	MAP-MoE	Standalone CAMERA-P
Logical retained channels	2,359,300	2,359,296
Aligned stored channels	2,399,440	2,405,904
Alignment overhead	1.70%	1.98%
Empty experts	235	38
Zero-tile entries	3,760	608

B Additional Analyses

B.1 MolmoE P50 Extension

MolmoE has 16 MoE layers, 64 experts per layer, and Top-8 routing. Calibration uses 128 WikiText-2 training segments, 64 ScienceQA-IMG validation images, and 64 TextVQA training images, without answer labels. We evaluate on ScienceQA test and TextVQA validation. CAMERA-P and MAP-MoE use the same prepared calibration inputs. CAMERA-P prunes layers sequentially; MAP-MoE keeps the grouping thresholds, pruning caps, and $\gamma = 2$ used in the main experiments. MAP-MoE uses P50 pruning rates of $0.8/0.6/0.4191744$ for the visual, mixed, and textual groups.

All methods use unquantized BF16 weights, the model processor with up to 36 crops, greedy decoding with at most 12 new tokens, and task-specific Molmo prompts. We score each method with the same ten lmms-eval tasks. Table 5 reports the full comparison. At 50% pruning, MAP-MoE improves average accuracy over CAMERA-P by 7.55 percentage points.

B.2 Task-wise Accuracy Gains

For Qwen, the largest improvement over CAMERA-P is on OK-VQA (+11.89 points). Across the other nine tasks, the average difference is approximately +0.17 points. For InternVL, the average difference across those nine tasks is approximately +1.33 points. Improvements also appear on TextVQA, GQA, AI2D, and ChartQA, which test OCR, perception, and chart understanding.

B.3 Modality-Label Control

Table 12 tests modality labels at Qwen-P50 while keeping group sizes, group rates, calibration, and RWE fixed. It reports each of the three random label permutations and their mean score on each task.

Table 12: Qwen-P50 modality-label control with group sizes, rates, calibration, and RWE fixed. Random labels use three permutations; MMVP uses pair accuracy.

Setting	OKV	SQA	TQA	GQA	POPE	MMB	AI2D	RWQA	CQA	MMVP	Avg.
Random 1	0.371	0.897	0.811	0.597	0.895	0.816	0.819	0.723	0.828	0.607	0.736
Random 2	0.397	0.900	0.803	0.603	0.888	0.824	0.803	0.703	0.808	0.580	0.731
Random 3	0.389	0.903	0.802	0.596	0.897	0.833	0.816	0.706	0.837	0.627	0.741
Random mean	0.386	0.900	0.805	0.598	0.893	0.824	0.813	0.711	0.824	0.604	0.736
Inferred labels	0.521	0.912	0.817	0.616	0.899	0.833	0.833	0.709	0.850	0.620	0.761

B.4 Capacity Allocation: Protection Policy and Equal-Marginal Solver

Group activation curves. For MoE layer l and modality group g, let $M_{l,g} = n_{l,g}m$ and sort the per-channel squared-activation sums $A_{l,e,j} = \sum_t z^2_{l,e,t,j}$ in ascending order: $a_{l,g,(1)} \leq \cdots \leq a_{l,g,(M_{l,g})}$. Define prefix sums $P_{l,g}(k) = \sum_{i=1}^{k} a_{l,g,(i)}$, with $P_{l,g}(0) = 0$, and total $T_{l,g} = P_{l,g}(M_{l,g})$. For $0 \leq r < 1$, set $k = \lfloor rM_{l,g} \rfloor$ and $\theta = rM_{l,g} - k$. The proportion of

squared activation removed by pruning the lowest-ranked r proportion of channels, and its average across MoE layers, are

$$\phi_{l,g}(r) = \frac{P_{l,g}(k) + \theta a_{l,g,(k+1)}}{T_{l,g}}, \qquad \mathcal{E}_g(r) = \frac{1}{L}\sum_{l=1}^{L} \phi_{l,g}(r), \tag{6}$$

with $\phi_{l,g}(1) = 1$. The two reported calibration caches have nonempty pools with positive squared-activation sums for every layer and group; the reference solver checks these conditions before optimization. We normalize within each layer and group before averaging across layers. Let $M_g = \sum_l M_{l,g}$ and $q_g = M_g / \sum_h M_h$. This weights each group curve by its share of all expert channels. Because the evaluated experts all start with the same m, q_g also equals $n_g / \sum_h n_h$. The allocation objective thus weights the normalized squared-activation curves by group size. These curves use squared activations only; RWE also uses output weights and activation variation when selecting channels. Since the sorted values are nonnegative and nondecreasing, $\phi_{l,g}$ and $\mathcal{E}_g$ are nondecreasing convex piecewise-linear functions.

Protection policy and feasible budgets. Fix $u_v = 0.8$, $u_m = 0.6$, and $u_t = 1$. We fix these cap values for all reported models and budgets. They are inputs to the allocation policy, not fitted outputs of the calibration-curve optimizer. The policy first minimizes textual pruning. Its smallest feasible value is

$$r_t^*(P) = \frac{[P - (q_v u_v + q_m u_m)]_+}{q_t}. \tag{7}$$

Equivalently, impose the retention floor $1 - r_t \geq 1 - r_t^*(P)$ together with the original budget and caps. Feasibility then forces $r_t = r_t^*(P)$, giving the two-step allocation in Eq. 3. The allowable global range is $0 \leq P \leq P_{\max} = q_v u_v + q_m u_m + q_t$; a request outside this range is infeasible under the chosen caps. The reported setting has $q_v, q_m, q_t > 0$.

For Qwen, $P_{vm} \approx 0.44$; for InternVL, $P_{vm} \approx 0.34$. P20 and P30 lie below these thresholds in both models, so $r_t = 0$; the calibration curves determine the allocation within the remaining constraints, and InternVL-P30 reaches the mixed-group cap. P50 exceeds both thresholds: $r_v = 0.8$ and $r_m = 0.6$ are then forced by maximal text retention, and the residual gives $r_t \approx 0.14$ and 0.29, respectively. Thus the textual retention floors at P50 are approximately 85.57% and 70.79%.

Rationale for the caps. The caps specify a minimum retained capacity for each nontextual group, while the optimizer determines the actual rates below them. The visual group contains more channels with small measured squared activations, so it can use a higher pruning cap. The mixed group receives stronger protection because its calibration responses are shared across modalities. On the allocation curves, $r_v = 0.8$ retains 78.33% and 73.25% of the normalized squared activation of the visual group in Qwen and InternVL. At $r_m = 0.6$, the mixed group retains 84.66% and 83.84%; increasing its rate to 0.8 would instead remove 31.53% and 30.85%. Together, these measurements motivate the fixed pruning caps for visual and mixed experts. Keeping the caps fixed across budgets and models preserves the same minimum-capacity policy. Below P_{vm}, the calibrated curves determine the remaining allocation freedom; above P_{vm}, the group sizes and budget determine the residual textual rate. The numerical cap values are policy choices; the calibrated optimization operates under these constraints.

Cap sensitivity. We vary the Qwen-P50 visual/mixed caps while keeping the global budget, calibration statistics, expert groups, RWE selector, and Group Top-k selection fixed. Budget conservation determines the textual rate in every setting. Table 13 reports all ten tasks. Relative to the default, changing both caps by -0.05 or $+0.05$ changes the 10-task average by +0.28 or -0.95 percentage points, respectively; the largest difference among the three settings is 1.24 points. These results characterize local sensitivity, and individual tasks respond differently to the capacity shifts.

Table 13: Qwen-P50 cap sensitivity. The total pruning budget remains 50%; only group rates change. Rates and accuracy are shown to 0.001.

Rates (r_v, r_m, r_t)	OKV	SQA	TQA	GQA	POPE	MMB	AI2D	RWQA	CQA	MMVP	Avg.
$(0.75, 0.55, 0.215)$	0.505	0.925	0.816	0.617	0.902	0.839	0.835	0.705	0.855	0.640	0.764
$(0.80, 0.60, 0.144)$	0.521	0.912	0.817	0.616	0.899	0.833	0.833	0.709	0.850	0.620	0.761
$(0.85, 0.65, 0.073)$	0.508	0.893	0.811	0.609	0.898	0.820	0.830	0.697	0.841	0.607	0.751

Constrained optimization and equal marginals. When $P \le P_{vm}$, substitute $r_m = (P - q_v r_v)/q_m$ and solve the one-dimensional convex problem

$$\begin{aligned} \min_{r_v \in [a,b]} \quad & F_P(r_v) = q_v \mathcal{E}_v(r_v) + q_m \mathcal{E}_m\left(\frac{P - q_v r_v}{q_m}\right), \\ & a = \max\left(0, \frac{P - q_m u_m}{q_v}\right), \qquad b = \min\left(u_v, \frac{P}{q_v}\right). \end{aligned} \tag{8}$$

At differentiable interior optima, $F_P'(r_v) = q_v[\mathcal{E}_v'(r_v) - \mathcal{E}_m'(r_m)] = 0$. More generally, the optimality conditions allow subgradients $s_g \in \partial \mathcal{E}_g(r_g)$ and multipliers $\alpha_g, \beta_g \ge 0$ such that

$$\begin{aligned} q_g(s_g - \lambda) - \alpha_g + \beta_g = 0, \quad g \in \{v, m\}, \\ \alpha_g r_g = 0, \qquad \beta_g(u_g - r_g) = 0. \end{aligned} \tag{9}$$

Thus free groups share $s_g = \lambda$; a group at its lower bound permits $s_g \ge \lambda$, and one at its upper bound permits $s_g \le \lambda$. These conditions include kinks in the empirical curves. In particular, InternVL-P30 activates the mixed cap, so equality of the two ordinary derivatives is not required. The equality concerns marginal proxy cost per unit of global pruning budget, $d(q_g \mathcal{E}_g)/d(q_g r_g)$, which explains why the q_g factors cancel.

Numerical algorithm. We use the following grid-and-refinement procedure for all reported budgets.

1. Compute the expert group assignments, M_g, q_g, and sorted squared-activation prefix sums from the calibration cache. Construct Eq. 6, with caps $(0.8, 0.6, 1)$ as fixed inputs.
2. Check $0 \le P \le P_{\max}$ and calculate $P_{vm} = q_v u_v + q_m u_m$. If $P > P_{vm}$, return $(u_v, u_m, (P - P_{vm})/q_t)$.
3. Otherwise set $r_t = 0$ and form $[a, b]$ in Eq. 8. If $a = b$, return $(a, (P - q_v a)/q_m, 0)$.
4. Evaluate F_P at 400,001 equally spaced points on $[a, b]$. Select the first smallest sampled value and bracket it by its adjacent grid points, clipped to $[a, b]$.
5. Refine this bracket with 150 ternary-search iterations. At trisection points $x_1 < x_2$, retain $[\mathrm{lo}, x_2]$ when $F_P(x_1) < F_P(x_2)$ and $[x_1, \mathrm{hi}]$ otherwise. Set r_v to the final midpoint, recover r_m from budget conservation, and return $(r_v, r_m, 0)$.
6. Use the unrounded rates for channel selection. Apply $D_{l,g} = \mathrm{round}(r_g M_{l,g})$ independently to each layer–group pool, using half-to-even rounding, and remove exactly that many lowest-scoring channels.

The curve stage permits fractional interpolation to define a continuous allocation proxy; the last step realizes it as integer channel counts. There is no additional layerwise reallocation and no redistribution of rounding remainders. With at most $3L$ nonempty layer–group sets, the difference between the realized and continuous global fractions is bounded by $3L/(2\sum_l E_l m)$, approximately 1.53×10^{-5} for the evaluated architectures. An individual expert can retain zero channels; the alignment rule applies only after channel selection. Rounded values in Table 1 are for display and are not used to generate masks. Equal scores are handled by `torch.topk`, without a separate tie-breaking rule.

B.5 Selector and Calibration Analyses

All accuracy averages in this subsection use the 10-task suite in Table 2; RWE follows Eq. 4 without modification.

Table 14: P50 selector control. All rows fix the groups, rates, calibration, and retained-channel counts. "CAMERA-style" uses the CAMERA-P score within MAP-MoE rather than the standalone method.

Model	Selector	OKV	SQA	TQA	GQA	POPE	MMB	AI2D	RWQA	CQA	MMVP	Avg.
Qwen	Energy	0.497	0.918	0.817	0.614	0.903	0.827	0.835	0.712	0.850	0.620	0.759
	RWE	0.521	0.912	0.817	0.616	0.899	0.833	0.833	0.709	0.850	0.620	0.761
	CAMERA-style	0.478	0.903	0.805	0.616	0.903	0.832	0.815	0.720	0.826	0.620	0.752
InternVL	Energy	0.370	0.965	0.766	0.620	0.889	0.820	0.835	0.695	0.840	0.620	0.742
	RWE	0.390	0.972	0.766	0.626	0.892	0.827	0.840	0.694	0.833	0.620	0.746
	CAMERA-style	0.397	0.974	0.749	0.617	0.893	0.829	0.836	0.671	0.854	0.640	0.746

RWE-exponent sensitivity. Table 15 varies γ at Qwen-P20 while fixing the expert groups, group rates, calibration statistics, and retained-channel count. Across $\gamma \in \{0.5, 1, 2, 4\}$, the displayed 10-task averages range from 0.775 to 0.777; the difference is 0.17 percentage points using unrounded scores. No value gives the highest score on every task. All main experiments use $\gamma = 2$; this test measures sensitivity near the default rather than choosing a separate exponent for each model or budget.

Table 15: Qwen-P20 RWE-exponent sensitivity. Groups, rates, calibration, and retained-channel count are fixed. Bold marks the highest unrounded score in each column.

γ	OKV	SQA	TQA	GQA	POPE	MMB	AI2D	RWQA	CQA	MMVP	Avg.
0.5	0.510	0.955	**0.838**	**0.626**	0.902	0.872	0.859	0.716	**0.871**	**0.613**	0.776
1	0.505	**0.956**	0.836	0.624	0.901	0.869	0.859	0.722	0.870	**0.613**	0.776
2	0.510	**0.956**	0.837	0.626	0.903	**0.875**	0.858	**0.725**	0.870	0.607	**0.777**
4	**0.520**	**0.956**	0.837	0.625	**0.905**	0.868	**0.860**	0.714	0.866	0.600	0.775

Variation-factor audit. Recomputing C from the stored sufficient statistics reproduces the stored variation factors elementwise in both frozen calibration caches. The factor reaches its upper bound on 2,899,640/4,718,592 channels (61.45%) in Qwen and 3,001,104/4,718,592 (63.60%) in InternVL, counting each expert–channel coordinate once. On these coordinates, the RWE score equals $A\|W^d\|_2^2$; squared activations and output-weight norms still distinguish the scores. The bounded factor encodes relative activation variability; Table 14 isolates its effect with group assignments and budgets fixed.

Calibration stability. Growing each Qwen draw from $64{+}64$ to $128{+}128$ raises pairwise agreement on expert group assignments from 86.6–88.7% to 90.2–90.7% and lowers the mean standard deviation of the modality bias from 0.0433 to 0.0328; three-draw aggregation reaches 94.7–95.2% leave-one-draw agreement. Remaining assignment changes are concentrated among mixed experts near the thresholds.

Independent calibration-to-mask repeats. Independent Qwen-P50 calibration draws regenerate the expert group assignments, allocation, and mask before evaluation (Table 16). The resulting 10-task average is 0.759 ± 0.003 (sample standard deviation over three draws), complementing the three-draw aggregate used for the main result.

Table 16: Independent Qwen-P50 calibration-to-mask repeats. Each draw regenerates expert group assignments, group rates, and the mask before evaluating all ten tasks.

Draw	OKV	SQA	TQA	GQA	POPE	MMB	AI2D	RWQA	CQA	MMVP	Avg.
First	0.504	0.918	0.819	0.612	0.891	0.825	0.834	0.712	0.854	0.607	0.758
Second	0.529	0.906	0.812	0.617	0.903	0.836	0.829	0.712	0.856	0.613	0.762
Third	0.524	0.905	0.815	0.610	0.901	0.820	0.823	0.689	0.848	0.627	0.756
Mean	0.519	0.910	0.816	0.613	0.899	0.827	0.829	0.705	0.853	0.616	0.759
Std.	0.013	0.007	0.003	0.004	0.006	0.008	0.005	0.014	0.004	0.010	0.003

Paired confidence intervals. We draw 2,000 paired bootstrap resamples within each task and report 95% percentile confidence intervals for the mean difference across the 10 tasks. MMBench uses circular-question groups and MMVP uses question pairs; the other tasks use individual questions. Using the unrounded per-example scores, at P50 MAP-MoE improves over CAMERA-P by 1.34 percentage points on Qwen (95% CI [0.68, 1.99]), and improves over REAP and CAMERA-P by 0.92 percentage points ([0.31, 1.57]) and 2.65 percentage points ([2.03, 3.28]), respectively, on InternVL. These intervals assume fixed calibration and masks; they measure uncertainty from the evaluation examples.

B.6 Optional Matched-Storage W8 Comparison

Setup. This optional study compares retaining all FFN channels of text-associated experts with W8 storage against removing half their channels in BF16. W8 uses INT8 weights with group size 128 and FP16 scales. With visual/mixed structure fixed, W8 text experts with all channels (29.325 GB MoE storage) achieve higher accuracy on all three reported tasks than BF16 text experts with half the channels (29.137 GB; Table 17). This comparison illustrates an additional way to retain text-expert channels under a storage budget.

Table 17: Matched-storage comparison on text experts (Qwen).

Method	OK-VQA	TextVQA	GQA
Text: half the channels, BF16	0.306	0.776	0.576
Text: all channels, W8	**0.483**	**0.820**	**0.600**

C Evidence and Audit

Calibration draws. Each draw contains 128 training images, sampled evenly from the four categories in Sec. 5, and 128 WikiText-2 training segments of 256 tokens each. For Qwen, we sum the activation and routing statistics from three fixed draws before generating one set of expert labels, rates, and pruning masks per budget. InternVL uses one fixed draw across the reported budgets. These pooled Qwen statistics are distinct from the three independent calibration-to-mask repeats in Sec. B.5. Calibration does not use answer labels. The source manifests record the exact splits, revisions, image hashes, and prompts.

Grouping thresholds and numerical settings. We use $\tau_t = 0.4$, $\tau_v = 0.6$, and $\epsilon = 10^{-12}$ across models and budgets. Experts without a strong preference fall into the mixed group. We test six threshold pairs on Qwen-P50 while keeping the calibration cache, total budget, pruning caps, RWE scores, and evaluation settings fixed (Table 18). The ten-task average ranges from 0.756 to 0.765; the default reaches 0.761. The highest point estimate, at $(0.35, 0.65)$, is 0.40 percentage points above the default, with a paired 95% confidence interval of $[-0.07, 0.88]$ percentage points. This test does not distinguish it from the default or establish a best choice for other models.

Table 18: Qwen-P50 grouping-threshold sensitivity. The calibration cache, 50% budget, pruning caps, RWE scores, and ten evaluation tasks are fixed.

(τ_t, τ_v)	Average	Change from default (pp)
$(0.30, 0.70)$	0.759	-0.20
$(0.35, 0.60)$	0.762	$+0.09$
$(0.35, 0.65)$	0.765	$+0.40$
$(0.40, 0.60)$ (default)	0.761	0.00
$(0.40, 0.65)$	0.762	$+0.10$
$(0.45, 0.55)$	0.756	-0.48

Visual-position masks. Qwen marks positions whose token ID equals the model image-token ID as visual. InternVL marks positions whose token ID equals the <IMG_CONTEXT> ID as visual. All remaining positions are textual.

Experts with missing modality observations. If either the visual or textual routed-token count is zero for an expert, we assign $b = 0.5$ and place it in the mixed group. This conservative rule avoids inferring modality specialization from an absence of observations in a small calibration sample. It affects 30 of 6,144 Qwen layer–expert pairs and 53 of 6,144 InternVL pairs.

DERN implementation settings. Both model adapters use similarity threshold 0.4 and gate/up/down component weights $(0, 0.5, 0.5)$ for channel-segment recombination. Weighted spherical k-means runs for at most 100 iterations with tolerance 10^{-4}. These settings are fixed across budgets.

Dataset splits and generation settings. Table 19 lists the dataset splits and generation limits stored with the evaluated runs. For tasks without an explicit cap, the recorded lmms-eval revision `6619ef61` supplies adapter defaults of 128 new tokens for Qwen and 1024 for InternVL. Greedy decoding with one beam is used. Qwen uses batch size eight for MAP-MoE and CAMERA-P and four for REAP and DERN, with `enable_thinking=False` and the Qwen3-VL chat adapter; InternVL uses batch size one, `device_map=cuda:0`, `max_num=12`, and `use_flash_attn=True`. The model-specific preprocessing and decoding settings are shared across pruning methods. The Qwen processor supplies its chat template; InternVL uses the chat interface of the corresponding model. Open-ended tasks request short answers and multiple-choice tasks request the option letter using the task-specific prompt configuration. The image processing path is model-specific: Qwen receives `max_pixels=602112` as an adapter argument, while InternVL uses 448-pixel dynamic tiles, up to 12 base tiles and a thumbnail for multi-tile inputs.

Table 19: Evaluated dataset configurations and explicit task-level generation caps, extracted from saved result configurations. "Adapter default" means the task configuration does not set a token cap: the recorded adapter defaults are 128 for Qwen and 1024 for InternVL. The MMVP dataset split is named `train`; the evaluation contains 300 individual questions grouped into 150 pairs.

Task	Dataset / subset	Split	Maximum new tokens
OK-VQA	lmms-lab/OK-VQA	val2014	Adapter default
ScienceQA	lmms-lab/ScienceQA / ScienceQA-IMG	test	16
TextVQA	lmms-lab/textvqa	validation	Adapter default
GQA	lmms-lab/GQA / testdev_balanced_instructions	testdev	16
POPE	lmms-lab/POPE / Full	popular	128
MMBench	lmms-lab/MMBench / en	dev	1024
AI2D	lmms-lab/ai2d	test	16
RealWorldQA	lmms-lab/real_world_qa	test	16
ChartQA	lmms-lab/chart_qa	test	16
MMVP	lmms-lab-eval/MMVP	train	16

The accompanying code can regenerate the reported results from saved calibration caches, mask hashes, per-task summaries, and run logs with SHA-256 records. We check budgets, keep-mask hashes, and tensor shapes before reporting results. OK-VQA is evaluated on the same fixed 4,946-example subset of `val2014` for all methods.

Table 20: Evaluation sample counts in paper order. Splits and decoding limits appear in Table 19.

Task	Evaluated N
OKV	4946
SQA	2017
TQA	5000
GQA	12578
POPE	3000
MMB	4329
AI2D	3088
RWQA	765
CQA	2500
MMVP	300