SPARCQUANT: SPARSE ADAPTIVE REPAIR OF CRITICAL CHANNELS FOR MLLMS QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training quantization (PTQ) for multimodal large language models is challenging because heterogeneous modalities induce distinct activation distributions in the shared decoder. Smoothing-based quantization has proven effective and is widely adopted for large language model quantization, yet directly extending it to these models can cause smoothing factors estimated from mixed calibration data to be dominated by high-magnitude modalities, thereby over-smoothing other modalities and increasing activation quantization error. Against this backdrop, the analysis in this paper reveals a previously overlooked structural property: **high sensitivity to modality-specific smoothing mismatch is concentrated in a small subset of channels**. Motivated by this finding, this paper proposes **SPARCQuant**, a sensitivity-guided framework for **sparse adaptive repair of critical channels** in multimodal large language model quantization. SPARCQuant selects a layer- and module-adaptive base modality to define shared smoothing and ranks channels using robust activation-range discrepancy and activation-magnitude importance. A coarse-to-fine proxy reconstruction search with online scheduling then **automatically selects the sparse channel support and its size**, adapting the repair to each layer, module, and non-base modality **without manual channel selection or per-layer budget tuning**. Selected-channel ridge compensation corrects residual errors while preserving a single shared quantized weight. Experiments on vision-language and omni-modal models demonstrate **improved robustness under low-bit quantization**. A CUDA implementation with custom kernels enables efficient deployment, achieving **2.66×–2.87× end-to-end inference speedup over BF16**.

1 INTRODUCTION

Multimodal large language models (MLLMs) have evolved from vision-language assistants into general-purpose systems that process images, videos, audio, and text through a unified generative interface. Models such as Flamingo (Alayrac et al., 2022), BLIP-2 (Li et al., 2023), MiniGPT-4 (Zhu et al., 2024), InstructBLIP (Dai et al., 2023), and LLaVA (Liu et al., 2023) show that connecting pretrained modality encoders with large language models enables open-ended multimodal reasoning and instruction following. Newer systems further support high-resolution perception, long-video understanding, and omni-modal interaction (Bai et al., 2025b; Xu et al., 2025). These capabilities, however, bring high memory and compute costs, especially because MLLMs process heterogeneous and often long modality token sequences, complicating deployment on resource-constrained or latency-sensitive platforms (Wang et al., 2025).

Among post-training quantization (PTQ) techniques, smoothing-based quantization has proven highly effective and has become a widely adopted paradigm for large language model quantization. Its key idea is to exploit function-preserving transformations that redistribute quantization difficulty between activations and weights, thereby suppressing activation outliers without full model retraining. Outlier Suppression (Wei et al., 2022) pioneered equivalent transformations for reducing activation outliers, while SmoothQuant (Xiao et al., 2023) formalized channel-wise smoothing through the invariance $Y = XW = (XS^{-1})(SW)$, migrating activation outliers into weights to enable efficient weight–activation quantization. Subsequent methods, including Outlier Suppression+ (Wei et al., 2023) and OmniQuant (Shao et al., 2024), further improve these transformations through enhanced

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

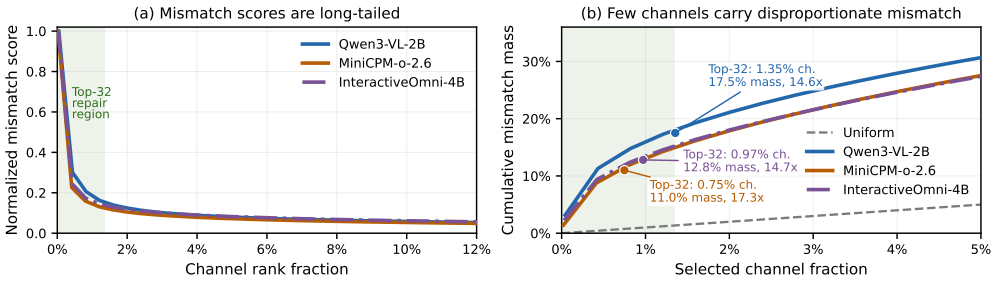


Figure 1: Calibration reveals long-tailed sensitivity-guided modality mismatch. For each layer, linear module, and non-base modality, channels are sorted by $C_i = A_i|\log R_i - \log R_{b,i}|$ (see Section 3.2 for definitions). Top-32 channels occupy a tiny input fraction but capture mismatch mass far above uniform allocation, motivating the repair of selected important channels.

shifting, scaling, or learnable smoothing parameters. Their broad adoption and strong performance establish smoothing as a practical foundation for low-bit LLM quantization.

MLLMs violate the single-distribution assumption behind many LLM-oriented PTQ methods. Text, vision, video, and audio tokens may have different activation ranges, variances, and quantization sensitivities inside the shared decoder. Figure 2 provides a concrete example: although text, vision, and audio tokens are processed by the same decoder, their channel-wise activation-range profiles are not aligned. Because smoothing factors are estimated channel by channel, this cross-modal heterogeneity can make a shared smoothing pattern favor one modality while poorly matching others. Recent work has therefore explored modality-aware quantization: QSLAW (Xie et al., 2024) learns quantization-aware scales, MBQ (Li et al., 2025a) balances reconstruction across vision and language tokens, MQuant (Yu et al., 2025) uses modality-specific static quantization, VLMQ (Xue et al., 2025) adds token-level importance to Hessian-based calibration, and LUQ (Bhatnagar et al., 2026), SPEED-Q (Guo et al., 2026), and VEQ (Qin et al., 2026) further highlight layer, component, expert, and modality heterogeneity under aggressive compression. MASQuant (Hu et al., 2026) mitigates cross-modal smoothing misalignment through modality-specific smoothing and low-rank compensation.

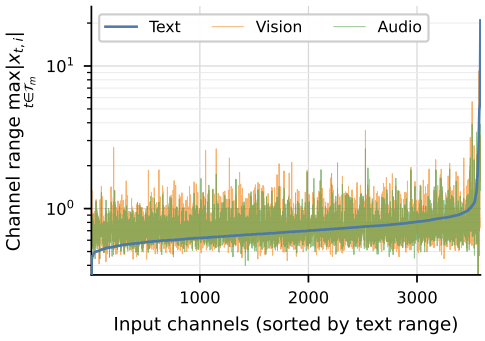


Figure 2: Cross-modal activation heterogeneity in MiniCPM-o-2.6 (OpenBMB, 2025) on six tri-modal OmniBench examples (Li et al., 2025b) spanning speech, sound events, and music. Curves show modality-wise per-channel maximum absolute activations at the first decoder `q_proj` input. Channels are sorted by text range; the non-aligned vision and audio profiles reveal modality-dependent distributions.

Against this backdrop, this paper uncovers a previously overlooked phenomenon: only a small subset of channels is sensitive to cross-modal smoothing mismatch, while most channels exhibit limited modality-dependent discrepancy. To characterize this sparse sensitivity, a calibration-based channel score serves as a proxy for vulnerability to shared smoothing: $C_i = A_i|\log R_i - \log R_{b,i}|$ (see Section 3.2 for definitions). Here, R_i and $R_{b,i}$ are robust activation radii for the current modality and a selected base (reference) modality, respectively, and A_i measures channel importance through normalized mean absolute activation. The absolute log-ratio captures relative activation-range mismatch, while the importance weight prioritizes channels that combine pronounced mismatch with large activation magnitudes. The score is computed for each layer, linear module, and non-base modality; Section 3.2 details the base-modality selection and channel-ranking procedure. As shown in Figure 1, the resulting scores follow a strongly long-tailed distribution. For Qwen3-VL-2B (Bai et al., 2025a), MiniCPM-o-2.6 (OpenBMB, 2025), and InteractiveOmni-4B (Tong et al., 2025), Top-32 channels capture 17.5%, 11.0%, and 12.8% of mismatch mass, with 14.6×, 17.3×, and

14.7× enrichment over uniform allocation, respectively. These results demonstrate that sensitivity to cross-modal smoothing mismatch is concentrated in a small set of important channels.

Motivated by this sparse channel-sensitivity phenomenon, this paper proposes **SPARCQuant**, a sensitivity-guided framework for **SP**arse **A**daptive **R**epair of **C**ritical channels when quantizing MLLMs. SPARCQuant selects a layer- and module-adaptive base modality from calibration statistics, which defines a shared smoothing pattern for most channels while preserving a single shared quantized weight. For each non-base modality, SPARCQuant identifies sensitivity-critical channels using robust channel-range discrepancy and activation-magnitude importance, and applies modality-specific smoothing only to the selected channels.

The remaining issue is how many channels to repair. Mismatch varies across layers, modules, bit-widths, and architectures, motivating calibration-driven budget allocation. SPARCQuant chooses the budget through an **automatic coarse-to-fine search** based on proxy reconstruction: for each candidate budget, it estimates reconstruction quality after selected-channel compensation and adds an explicit complexity penalty. An **online layer-wise scheduler** adapts the candidate search space across early, middle, late, and tail layers using calibration-time reconstruction signals, so the sparse budget follows depth-dependent quantization difficulty **without manual per-layer budget tuning**.

Finally, SPARCQuant fits selected-channel ridge compensation to correct residual output errors while preserving a single shared quantized weight. The correction is restricted to the automatically selected support, so its effective rank is bounded by the chosen channel budget. The central benefit is calibration-driven selection of which channels to repair and how many to use, without manually specifying a budget for each layer and module. Figure 3 presents the complete SPARCQuant method. Experiments demonstrate improved robustness under low-bit quantization and ease of deployment, with custom CUDA kernels achieving 2.66×–2.87× end-to-end inference speedup over BF16.

The main contributions are summarized as follows:

- **This paper proposes SPARCQuant, a framework for sparse adaptive channel repair when quantizing MLLMs.** It applies modality-specific smoothing and compensation to sensitivity-critical channels while preserving a single shared quantized weight.
- **This paper reveals that cross-modal smoothing mismatch is concentrated in a small subset of channels.** Calibration analysis across vision-language and omni-modal models shows long-tailed sensitivity scores, motivating sparse modality-specific repair.
- **This paper develops automatic budget search without manual per-layer budget tuning.** Coarse-to-fine search and online scheduling select suitable channel supports using a regularized reconstruction objective.

2 Related Work

2.1 Smoothing-based Quantization for Large Language Models

Smoothing-based PTQ is widely used to mitigate activation outliers in large language models. Outlier Suppression (Wei et al., 2022) reduces outliers through equivalent transformations and clipping, while SmoothQuant (Xiao et al., 2023) exploits linear-layer invariance to migrate activation outliers into weights with channel-wise scaling. Outlier Suppression+ (Wei et al., 2023) improves the shifting and scaling transformations. AWQ (Lin et al., 2024) protects salient weight channels using activation statistics, and OmniQuant (Shao et al., 2024) learns quantization and smoothing parameters to reduce low-bit reconstruction error. These methods work well for text-only LLMs, but they usually estimate one smoothing pattern from one calibration distribution, making them less suitable for multimodal models with heterogeneous modality-specific activations.

2.2 Quantization Optimization for Multimodal Large Language Models

Recent studies on quantizing MLLMs explicitly consider modality heterogeneity. QSLAW (Xie et al., 2024) learns quantization-aware scales for multimodal adaptation; MBQ (Li et al., 2025a) balances quantization loss across visual and textual tokens; MQuant (Yu et al., 2025) introduces modality-specific static quantization; VLMQ (Xue et al., 2025) improves PTQ with Hessian-augmented

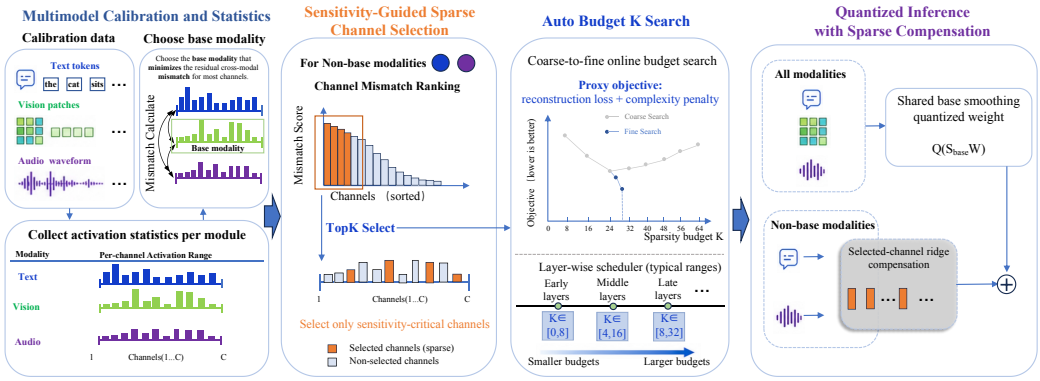


Figure 3: Overview of SPARCQuant. The pipeline collects modality-specific statistics, selects a layer- and module-adaptive base modality, ranks non-base channels by sensitivity-guided mismatch, searches sparse budget K online, and performs quantized inference with selected-channel compensation.

calibration; and MASQuant (Hu et al., 2026) mitigates cross-modal smoothing misalignment through modality-aware smoothing and low-rank compensation.

The analysis in this paper further reveals that cross-modal smoothing sensitivity is concentrated in only a small subset of channels. Based on this finding, SPARCQuant combines sensitivity-guided channel ranking with automatic budget search to determine where modality-specific smoothing and compensation are needed, while sharing a base smoothing pattern across the remaining channels.

3 METHOD

This paper proposes **SPARCQuant**, a sensitivity-guided framework for automatically selecting sparse repair channels for PTQ of MLLMs. The key observation is that smoothing sensitivity and modality mismatch are not uniform across hidden channels. SPARCQuant uses one base smoothing pattern for most channels and activates modality-specific smoothing on a selected support. Sensitivity scores determine the channel ranking, and online proxy reconstruction search chooses the support size during calibration, removing the need to manually select channels or tune each layer's budget.

Figure 3 summarizes the pipeline: modality-specific calibration statistics are collected, a shared base modality and sparse high-mismatch support are selected, the budget K is searched online, and quantized inference uses selected-channel compensation.

3.1 PRELIMINARIES AND MOTIVATION

Consider a linear module in an MLLM decoder: $Y = XW + b$, where $X \in \mathbb{R}^{T\times d_{\text{in}}}$ and $W \in \mathbb{R}^{d_{\text{in}}\times d_{\text{out}}}$. Smoothing-based quantization introduces a diagonal matrix $S = \text{Diag}(s)$ and rewrites $XW = (XS^{-1})(SW)$, which migrates activation outliers into weights before quantization (Xiao et al., 2023). In MLLMs, text, vision, and audio tokens can have very different activation ranges, so a unified smoothing factor may be dominated by one modality and over-smooth the others, causing smoothing misalignment (Hu et al., 2026).

A dense remedy is to learn modality-specific smoothing factors $\{S_m\}_{m\in\mathcal{M}}$, but this requires modality-dependent quantized weights or extra dense compensation. SPARCQuant instead uses the sparse parameterization

$$\log s_{m,i} = \log s_i^0 + z_{m,i}\delta_{m,i}, \qquad \|z_m\|_0 = K_m \ll d_{\text{in}}, \tag{1}$$

where s^0 is the shared base scale, z_m is a selected-channel mask, and δ_m is a modality-specific residual active only on selected channels. Thus most channels share the base pattern, while only sensitivity-critical high-mismatch channels are corrected.

3.2 Base Modality and Sparse Channel Selection

This subsection implements the calibration/statistics and sparse-selection stages in Figure 3: modality-specific activation ranges are estimated and converted into a base-modality decision plus a ranked high-mismatch support.

For each layer and linear module, SPARCQuant first selects a *base modality* rather than fixing text as reference. Let $R_{m,i}$ be the robust activation radius of modality m on channel i, and $A_{m,i}$ its activation-magnitude importance; Appendix A.3.1 gives the estimators. The sensitivity-guided mismatch from modality m to candidate base b is

$$D_{m\to b,i} = A_{m,i}\,|\log R_{m,i} - \log R_{b,i}|\,. \tag{2}$$

The base modality minimizes the mismatch left after a small number of sensitivity-critical channels are assigned to the sparse branch:

$$b^\star = \arg\min_{b\in\mathcal{M}} \sum_{m\neq b} \mathrm{TailSum}_\kappa\big(D_{m\to b,:}\big), \tag{3}$$

where $\mathrm{TailSum}_\kappa$ removes the largest κ entries before summation. This favors a base that represents most channels across modalities, rather than the modality with the largest activation magnitude.

After selecting $b^\star$, the base smoothing scale is initialized as $s_i^0 = R_{b^\star,i}^{\alpha}/P_i^{1-\alpha}$, where P_i is the input-channel weight range and α controls activation-to-weight migration. For each non-base modality m, SPARCQuant ranks channels by $D_{m\to b^\star,i}$ and selects the Top-K_m channels. The residual scale is initialized as the log-ratio between the modality-specific scale and the base scale; during calibration, only s^0 and active residuals δ_m are optimized.

3.3 Online Auto-Budget Search

The auto-budget block in Figure 3 performs a coarse-to-fine search over K with a layer-wise scheduler, using smaller early-layer budgets and larger budgets when later layers need stronger repair.

The sensitivity score ranks channels but does not determine how many to select. Since K_m should vary across layers, modules, modalities, and bit-widths, SPARCQuant chooses it online with a proxy reconstruction objective. For a candidate budget K and selected set $\mathcal{I}_K$, SPARCQuant evaluates

$$K_m^\star = \arg\min_{K\in\mathcal{C}_l} \left[\mathcal{L}^{\mathrm{proxy}}(K) + \mu_l \frac{K}{K_{\max,l}}\right], \tag{4}$$

where $\mathcal{L}^{\mathrm{proxy}}(K)$ is the output reconstruction loss after fitting selected-channel ridge compensation, and the second term regularizes the support size. Combined with sensitivity ranking, this objective selects a suitable channel support from the searched candidates without manual per-layer budget tuning. The candidate set $\mathcal{C}_l$ and penalty μ_l are scheduled by layer depth and reconstruction difficulty: early layers use smaller budgets, middle layers moderate budgets, and late layers larger budgets when quantization is harder. Appendix A.3.2 details the coarse-to-fine search and late-tail scheduler.

3.4 Selected-Channel Ridge Compensation

The rightmost stage of Figure 3 gives the inference form: all modalities use the shared base-smoothed quantized weight, while only non-base modalities invoke selected-channel compensation.

All modalities share one base quantized weight, $\widehat{W}^0 = Q_w\big(\mathrm{Diag}(s^0)W\big)$, so the modality-specific smoothing in Eq. (1) no longer gives exact invariance. SPARCQuant therefore corrects residual output error only on the selected support.

For modality m, define $\widetilde{X}_m = Q_a\big(X_m\mathrm{Diag}(s_m)^{-1}\big)$. The final output is

$$\widehat{Y}_m = \widetilde{X}_m\widehat{W}^0 + \widetilde{X}_{m,:,\mathcal{I}_m}B_m + b + c_m, \tag{5}$$

where $B_m \in \mathbb{R}^{K_m\times d_{\mathrm{out}}}$ and c_m are fitted by ridge regression (Hoerl & Kennard, 1970) on calibration data; Appendix A.3.3 gives the closed form. For the base modality, $\mathcal{I}_m$ is empty and the branch is skipped.

This preserves a single shared quantized weight, with $K_m d_{\mathrm{out}}$ entries in the compensation matrix B_m and a bias vector c_m for each non-base modality. The automatically selected K_m determines the scope of correction; when $K_m = 0$, the selected-channel branch is skipped. Sparsity controls where correction is applied, while calibration still includes reconstruction search and parameter fitting.

Table 1: Main results on Qwen3-VL. Metrics are percentages. Only the Thinker/LLM is quantized; BF16 is the dense reference. Best PTQ results are **bold**, and second-best results are underlined.

Model	Bits	Method	MMMU↑	TextVQA↑	VizWiz↑	ScienceQA↑	OCRBench↑	Avg.↑
Qwen3-VL -2B	BF16	Dense	37.33	79.48	67.37	82.43	74.30	68.18
	W8A8	RTN	**41.33**	79.26	66.39	80.81	74.90	68.54
		SQ (ICML2023)	39.89	**79.45**	65.72	81.23	74.00	68.06
		MBQ (CVPR2025)	40.89	79.15	**67.38**	80.81	**75.00**	68.64
		MASQ (CVPR2026)	38.44	77.40	66.80	70.95	74.50	65.62
		SPARCQuant	40.78	79.43	66.98	**82.43**	**75.00**	**68.93**
	W4A8	RTN	31.44	62.89	55.90	63.69	61.90	55.16
		SQ (ICML2023)	26.44	0.00	0.12	2.78	11.70	8.21
		MBQ (CVPR2025)	35.11	69.31	64.42	**68.66**	64.00	60.30
		MASQ (CVPR2026)	32.22	44.41	46.27	43.43	61.60	45.59
		SPARCQuant	**35.89**	**71.80**	**65.10**	67.46	**65.80**	**61.21**
	W4A6	RTN	27.44	15.49	25.65	14.64	23.10	21.26
		SQ (ICML2023)	25.89	0.04	1.21	3.56	5.20	7.18
		MBQ (CVPR2025)	30.89	62.36	56.88	45.08	58.00	50.64
		MASQ (CVPR2026)	29.33	53.12	35.05	9.46	52.70	35.93
		SPARCQuant	**33.33**	**64.24**	**57.01**	**61.07**	**58.20**	**54.77**
Qwen3-VL -8B	BF16	Dense	51.78	80.72	68.84	92.74	75.50	73.91
	W8A8	RTN	**52.22**	80.30	68.19	92.29	74.90	73.58
		SQ (ICML2023)	51.67	79.92	67.99	92.48	74.80	73.37
		MBQ (CVPR2025)	51.67	80.70	68.18	92.57	75.80	73.78
		MASQ (CVPR2026)	49.33	80.06	67.58	90.78	72.00	71.95
		SPARCQuant	51.89	**80.83**	**68.30**	**92.74**	**75.90**	**73.93**
	W4A8	RTN	**49.22**	71.70	64.30	86.04	64.60	67.17
		SQ (ICML2023)	29.67	43.03	51.26	54.77	40.70	43.89
		MBQ (CVPR2025)	48.56	76.24	66.54	88.06	61.90	68.26
		MASQ (CVPR2026)	43.78	76.79	65.98	84.51	**66.10**	67.43
		SPARCQuant	47.00	**77.96**	**67.05**	**88.16**	65.20	**69.07**
	W4A6	RTN	26.00	38.02	47.83	19.62	**67.00**	39.69
		SQ (ICML2023)	28.44	23.24	35.01	44.26	32.30	32.65
		MBQ (CVPR2025)	41.44	68.49	51.98	77.60	56.30	59.16
		MASQ (CVPR2026)	32.11	53.95	50.86	58.64	51.10	49.33
		SPARCQuant	**43.56**	**73.33**	**64.73**	**81.47**	59.80	**64.58**

3.5 Layer-Wise Calibration Objective

SPARCQuant follows block-wise PTQ. For layer l, let H_{l+1}^{fp} and $\widehat{H}_{l+1}$ be floating-point and quantized block outputs. The base scale and active sparse residuals are optimized with the modality-balanced objective

$$\mathcal{L}_l = \sum_{m\in\mathcal{M}} \omega_m \frac{\left\| M_l^{(m)} \odot \left(\widehat{H}_{l+1} - H_{l+1}^{\mathrm{fp}}\right)\right\|_1}{\|M_l^{(m)}\|_1 + \epsilon} + \lambda_\delta \sum_{m\in\mathcal{M}} \|z_m \odot \delta_m\|_1, \tag{6}$$

where $M_l^{(m)}$ is the modality token mask, ω_m the modality weight, and the second term discourages unnecessary deviation from the base scale. After one layer is optimized, SPARCQuant solves selected-channel compensation and moves to the next layer, using only calibration data and no full-model retraining.

4 Experiments

4.1 Experimental Setup

Models. SPARCQuant is evaluated on vision-language and omni-modal MLLMs. Vision-language experiments use Qwen3-VL-2B and Qwen3-VL-8B (Bai et al., 2025a); omni-modal experiments use MiniCPM-o-2.6 (OpenBMB, 2025) and InteractiveOmni-4B (Tong et al., 2025). Following a deployment-oriented setting, only the shared understanding LLM/decoder is quantized. For Qwen3-VL, image/video tokens are treated as vision modality during calibration; for the omni-modal models, text, vision, and audio masks are built from preprocessed multimodal inputs.

Quantization and calibration settings. SPARCQuant is compared with RTN, SmoothQuant (SQ) (Xiao et al., 2023), MBQ (Li et al., 2025a), and MASQuant (MASQ) (Hu et al., 2026) under

Table 2: Main results on omni-modal models (%). Only the understanding decoder is quantized. Best and second-best PTQ results are **bold** and underlined, respectively.

Model	Bits	Method	LibriSpeech WER↓	FLEURS-English WER↓	TED-LIUM 3 WER↓	MMMU-val↑	OmniBench↑
MiniCPM-o-2.6	BF16	Dense	11.56	11.76	43.63	48.11	44.87
	W8A8	RTN	13.86	17.61	51.72	47.00	44.77
		SQ (ICML2023)	15.80	19.21	56.20	46.00	44.58
		MBQ (CVPR2025)	12.11	**11.22**	48.72	47.10	44.96
		MASQ (CVPR2026)	11.40	13.29	48.19	47.11	44.60
		SPARCQuant	**10.48**	11.41	**44.19**	**47.33**	**45.31**
	W4A8	RTN	33.03	58.32	58.91	42.88	41.77
		SQ (ICML2023)	73.00	73.46	99.51	32.33	23.98
		MBQ (CVPR2025)	58.35	70.94	80.46	41.67	39.03
		MASQ (CVPR2026)	23.86	31.87	59.15	42.56	44.25
		SPARCQuant	**15.81**	**22.05**	**47.60**	**43.44**	**44.34**
	W4A6	RTN	58.81	81.01	87.90	32.33	36.11
		SQ (ICML2023)	103.69	104.65	115.25	25.44	4.60
		MBQ (CVPR2025)	56.52	95.48	71.10	33.44	36.02
		MASQ (CVPR2026)	98.84	182.20	165.32	35.78	36.02
		SPARCQuant	**22.49**	**29.70**	**64.94**	**40.89**	**42.12**
Interactive Omni-4B	BF16	Dense	4.02	6.62	3.36	53.78	46.28
	W8A8	RTN	4.02	6.76	3.47	53.33	**47.35**
		SQ (ICML2023)	4.11	6.68	3.35	**54.78**	46.99
		MBQ (CVPR2025)	4.04	**4.77**	3.41	53.89	46.28
		MASQ (CVPR2026)	4.07	6.58	**3.25**	**54.78**	45.75
		SPARCQuant	**4.01**	6.52	3.47	53.89	46.46
	W4A8	RTN	18.00	22.41	62.33	45.67	41.15
		SQ (ICML2023)	12.17	20.67	51.49	46.33	42.12
		MBQ (CVPR2025)	10.64	12.57	27.76	**50.44**	38.58
		MASQ (CVPR2026)	**5.63**	5.72	5.65	47.89	41.77
		SPARCQuant	5.67	**5.42**	**4.74**	**50.44**	**45.93**
	W4A6	RTN	40.08	47.60	58.72	34.56	28.05
		SQ (ICML2023)	13.49	12.12	27.96	40.11	36.90
		MBQ (CVPR2025)	21.84	27.14	48.86	43.67	35.84
		MASQ (CVPR2026)	7.62	8.62	11.65	38.67	38.14
		SPARCQuant	**6.35**	**6.29**	**5.08**	**46.22**	**41.24**

W4A6, W4A8, and W8A8, with BF16 as the dense reference. All methods use 128 calibration samples: COCO-2017 image-text examples (Lin et al., 2014) for Qwen3-VL and OmniBench samples (Li et al., 2025b) with visual, acoustic, and textual inputs for MiniCPM-o-2.6 and InteractiveOmni-4B. These samples are used only for PTQ parameter search, without task-specific training or full-model finetuning. As native INT6 GEMM is unavailable on the target GPU, A6 uses six-bit quantization bounds in an INT8 container and is evaluated as an accuracy configuration.

Deployment protocol. CUDA W8A8, W4A8, and W4A6 are benchmarked against BF16 on Qwen3-VL-2B/8B using one RTX 5090, batch size one, a 319-token image–text prompt, and 32 generated tokens. Timings are averaged over five runs after warm-up; the embedding, LM head, and KV cache remain in BF16. Runtime and measurement details are summarized in Appendix A.2.

Evaluation benchmarks. Qwen3-VL is evaluated on MMMU (Yue et al., 2024), TextVQA (Singh et al., 2019), VizWiz (Gurari et al., 2018), ScienceQA (Lu et al., 2022), and OCRBench (Liu et al., 2024). MiniCPM-o-2.6 and InteractiveOmni-4B are evaluated on LibriSpeech `test-other` (Panayotov et al., 2015), FLEURS-English `en_us/test` (Conneau et al., 2023), TED-LIUM 3 `legacy/test` (Hernandez et al., 2018), MMMU, and OmniBench (Li et al., 2025b), using the complete evaluation splits.

4.2 Main Results

Tables 1 and 2 compare SPARCQuant with four PTQ baselines across vision-language and omni-modal models. We discuss the results from W8A8 to W4A8 and W4A6 to examine how the advantage changes as the quantization budget decreases.

At W8A8, most methods remain close to BF16 on many tasks, with relatively small performance gaps; SPARCQuant nevertheless retains an overall advantage. On Qwen3-VL-2B/8B, it achieves the highest average scores of 68.93/73.93 versus 68.64/73.78 for the strongest baseline, while matching BF16 ScienceQA accuracy at 82.43/92.74. On MiniCPM-o-2.6, it leads four of five tasks, reducing LibriSpeech WER from the best baseline's 11.40 to 10.48. It also achieves the lowest InteractiveOmni-4B LibriSpeech WER, at 4.01 versus 4.02 for BF16. These results demonstrate **that SPARCQuant preserves strong accuracy when quantization error is already limited.**

At W4A8, the lower weight-bit budget makes SPARCQuant's advantage more pronounced. Its Qwen3-VL-2B/8B average scores of 61.21/69.07 exceed the strongest baselines of 60.30/68.26 by 0.91/0.81 points, compared with gains of 0.29/0.15 at W8A8. On MiniCPM-o-2.6, SPARCQuant leads all five tasks; in particular, **LibriSpeech/FLEURS-English WER drops from the best baseline values of 23.86/31.87 to 15.81/22.05**, while TED-LIUM 3 WER improves from 58.91 to 47.60. On InteractiveOmni-4B, it leads FLEURS-English, TED-LIUM 3, and OmniBench and ties the best MMMU-val score.

At W4A6, the extremely low weight and activation bit budgets expose much larger differences in robustness, and SPARCQuant delivers its most substantial gains. It achieves the best PTQ result on **19 of the 20 model–task combinations**. On Qwen3-VL-2B/8B, its average scores reach 54.77/64.58, improving over the strongest baselines by **4.13/5.42 points**. Particularly large gains occur on Qwen3-VL-2B ScienceQA (**61.07 versus 45.08**, a 15.99-point gain) and Qwen3-VL-8B VizWiz (**64.73 versus 51.98**, a 12.75-point gain). Some baselines suffer near-total failure on individual tasks: SmoothQuant scores only 0.04/1.21 on Qwen3-VL-2B TextVQA/VizWiz, whereas SPARCQuant retains **64.24/57.01**. The same advantage extends to both omni-modal models, where SPARCQuant leads all five tasks at W4A6. On MiniCPM-o-2.6, **LibriSpeech/FLEURS-English WER falls from 56.52/81.01 for the strongest baselines to 22.49/29.70**, corresponding to relative error reductions of 60.2%/63.3%. SmoothQuant and MASQuant exceed 100% WER on FLEURS-English and TED-LIUM 3, indicating severe recognition degradation; SPARCQuant obtains 29.70/64.94, respectively. Its MiniCPM OmniBench accuracy remains **42.12, compared with just 4.60 for SmoothQuant** and 36.11 for the strongest baseline. On InteractiveOmni-4B, **TED-LIUM 3 WER decreases from the best baseline's 11.65 to 5.08, a 56.4% relative reduction**, while LibriSpeech/FLEURS-English WER improves from 7.62/8.62 to 6.35/6.29 and MMMU-val/OmniBench rise from 43.67/38.14 to 46.22/41.24.

4.3 Sensitivity-guided Sparse Modality Mismatch Analysis

This analysis tests whether cross-modal mismatch is concentrated in a few channels using $C_i = A_i|\log R_i - \log R_{b,i}|$, where R_i and $R_{b,i}$ are modality-specific robust radii and A_i is activation importance. Top-K mismatch-mass coverage and enrichment over uniform allocation are averaged over layer/module/non-base-modality combinations.

Table 3: Sensitivity-guided sparse mismatch concentration in W4A8. Top-32 Ch. is the mean selected-channel ratio for $K = 32$; coverage is the mismatch mass captured by Top-K channels.

Model	Pairs	Top-32 Ch.	Top-16 Cov.	Top-32 Cov.	Top-16 Enrich.	Top-32 Enrich.	Gini
MiniCPM-o-2.6	280	0.75%	8.0%	11.0%	24.7×	17.3×	0.5213
Qwen3-VL-2B	140	1.35%	13.4%	17.5%	22.2×	14.6×	0.5389
InteractiveOmni-4B	360	0.97%	9.8%	12.8%	22.2×	14.7×	0.5199

Table 3 shows that Top-32 channels account for at most 1.35% of input channels on average, but capture 11.0%–17.5% of mismatch mass (14.6×–17.3× enrichment), with Gini coefficients of 0.5199–0.5389. InteractiveOmni-4B captures 12.8% using just 0.97% of channels. Module concentration also varies: averaging modality-pair means, `q_proj` has the highest coverage for Qwen3-VL-2B (26.4%) and MiniCPM-o-2.6 (16.8%), while `gate_proj`/`up_proj` lead for InteractiveOmni-4B (19.0%). Lower coverage in output/down projections further supports sparse correction with module-adaptive budgets.

Table 4: Selector ablation for sparse repair channels. The table reports proxy reconstruction MAE for the sensitivity-guided selector and its relative reduction over alternatives at the same K. Lower is better.

Model	K	Sensitivity-score MAE	vs. `range_gap`	vs. activation	vs. random
MiniCPM-o-2.6	8	0.0887	2.5%	-0.3%	4.3%
MiniCPM-o-2.6	16	0.0812	3.8%	0.2%	6.0%
MiniCPM-o-2.6	32	0.0696	4.8%	0.8%	7.5%
Qwen3-VL-2B	8	0.2430	1.3%	0.9%	6.1%
Qwen3-VL-2B	16	0.1909	9.0%	3.5%	12.3%
Qwen3-VL-2B	32	0.1353	10.2%	2.4%	13.7%

Table 4 shows that the sensitivity-guided selector reduces proxy MAE by 4.3%–13.7% over random support and outperforms range-gap-only selection. It is competitive with activation-only selection on MiniCPM-o-2.6 and consistently better on Qwen3-VL-2B, validating the combined score.

4.4 Component Ablation

The component ablation evaluates SPARCQuant on MiniCPM-o-2.6 W4A6 using LibriSpeech `test-other` WER, average selected budget, and extra parameter ratio.

Table 5: Component ablation on MiniCPM-o-2.6 W4A6. Extra Params are relative to LLM Linear weights; ΔWER is relative to full SPARCQuant.

Variant	Intervention	Avg. K	Extra Params	WER↓	ΔWER
Full SPARCQuant	reference	11.47	0.53%	**22.49**	ref
Fixed K=16	global budget	16.00	0.74%	34.86	+12.37
Random support	random channels	8.72	0.40%	42.91	+20.42
Base-only	no sparse repair	0.00	0.02%	46.57	+24.08
Auto sparse only	no compensation	2.37	0.02%	49.07	+26.58
Fixed text base	no adaptive base	9.58	0.43%	49.51	+27.02

Table 6: CUDA deployment on one RTX 5090 with SDPA, a 319-token prompt, and 32-token generation. Alloc./Peak denote post-pack/peak CUDA memory; Save and Rel. wall are relative to paired BF16. GiB uses 2^{30} bytes.

Model	Runtime	Alloc. GiB	Save	Peak GiB	Wall ms	Prefill ms	Decode ms/tok	Tokens/s	Rel. wall
2B	BF16	3.963	–	4.034	579.70	32.88	17.323	57.73	1.000×
	W8A8	2.684	32.28%	2.741	204.41	25.80	5.449	183.57	2.857×
	W4A8	2.044	48.43%	2.113	201.33	26.21	5.344	187.13	2.842×
	W4A6	2.042	48.47%	2.112	203.20	26.84	5.379	185.91	2.869×
8B	BF16	16.330	–	16.432	766.94	62.73	22.400	44.65	1.000×
	W8A8	9.913	39.30%	10.004	289.48	36.88	7.839	127.57	2.656×
	W4A8	6.711	58.90%	6.849	275.31	46.55	7.069	141.46	2.812×
	W4A6	6.709	58.91%	6.848	279.80	46.72	7.197	140.06	2.709×

Table 5 supports adaptive sparse repair on this ASR benchmark. Removing sparse repair or compensation more than doubles WER, while fixing text as the base performs worst. Random support adds 20.42 WER points, and fixed $K = 16$ uses more parameters yet adds 12.37 points, supporting sensitivity-guided selection and calibration-driven budgeting. Further base-modality and budget diagnostics appear in Appendices A.4–A.6.

4.5 End-to-End CUDA Deployment Performance

To characterize inference behavior after calibration, a CUDA runtime is developed to support W8A8, W4A8, and W4A6 for Qwen3-VL-2B and Qwen3-VL-8B. The runtime uses resident integer weights and fused decoder operations, while attention is executed with SDPA. Full-model CUDA memory and end-to-end autoregressive generation performance are reported against BF16.

Table 6 reports the deployment measurements. Relative to the paired BF16 baseline, the quantized CUDA runtime reduces full-model allocated memory by 32.3%–58.9% and achieves 2.656×–2.869× end-to-end speedup on the tested setup, while peak generation memory follows the same reduction trend. **These results demonstrate SPARCQuant's practicality and ease of deployment.**

5 Conclusion

This paper reveals that sensitivity to cross-modal smoothing mismatch is concentrated in a small subset of channels. Based on this finding, we propose SPARCQuant, a framework for sparse adaptive repair of critical channels in MLLMs. It combines adaptive base-modality selection, sensitivity-guided channel ranking, online budget search, and selected-channel ridge compensation to automatically determine which channels to repair and how many, while preserving a shared quantized weight. Experiments on vision-language and omni-modal models demonstrate improved robustness under low-bit quantization. Custom CUDA kernels deliver the expected inference performance, demonstrating SPARCQuant's practicality and ease of deployment.

AI USE STATEMENT

During the preparation of this manuscript, AI tools were used solely to assist with proofreading, language polishing, code writing, and searching for relevant literature. Their role was limited to supporting these tasks and did not replace the authors' independent judgment or intellectual contributions. The authors retain full responsibility for the accuracy, integrity, and originality of the manuscript, including the references, code, analyses, interpretations, and conclusions.

ETHICS STATEMENT

This work does not introduce a new dataset or involve new studies with human subjects. It uses public datasets and model checkpoints under their applicable licenses and terms. Quantized models retain the safety, privacy, fairness, and dual-use risks of their underlying MLLMs.

REPRODUCIBILITY STATEMENT

Section 4.1 specifies the evaluated models, calibration data, baselines, bit-widths, and benchmarks. Appendix A.1 summarizes the supplementary code and evaluation protocol, with exact configurations and reproduction commands provided in the code documentation, while Appendices A.3–A.6 provide algorithmic and diagnostic details needed to reproduce the reported results.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736, 2022. doi: 10.52202/068431-1723.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025a. URL https://arxiv.org/abs/2511.21631.

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

Shubhang Bhatnagar, Andy Xu, Kar-Han Tan, and Narendra Ahuja. LUQ: Layerwise ultra-low bit quantization for multimodal large language models. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=3eK6U6ZiSp.

Alexis Conneau, Min Ma, Simran Khanuja, Yu Zhang, Vera Axelrod, Siddharth Dalmia, Jason Riesa, Clara Rivera, and Ankur Bapna. FLEURS: Few-shot learning evaluation of universal representations of speech. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pp. 798–805, 2023. doi: 10.1109/slt54892.2023.10023141.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49250–49267, 2023. doi: 10.52202/075280-2142.

Tianyu Guo, Shanwei Zhao, Shiai Zhu, and Chenguang Ma. SPEED-Q: Staged processing with enhanced distillation towards efficient low-bit on-device VLM quantization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(26):21486–21494, 2026. doi: 10.1609/aaai.v40i26.39296.

Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. VizWiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3608–3617, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Gurari_VizWiz_Grand_Challenge_CVPR_2018_paper.html.

François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Estève. TED-LIUM 3: Twice as much data and corpus repartition for experiments on speaker adaptation. In *Speech and Computer (SPECOM 2018)*, pp. 198–208. Springer, 2018. doi: 10.1007/978-3-319-99579-3_21.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634.

Lulu Hu, Wenhu Xiao, Xin Chen, Xinhua Xu, Bowen Xu, Kun Li, and Yongliang Tao. MASQuant: Modality-aware smoothing quantization for multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8707–8716, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Hu_MASQuant_Modality-Aware_Smoothing_Quantization_for_Multimodal_Large_Language_Models_CVPR_2026_paper.html.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742. PMLR, 2023. URL https://proceedings.mlr.press/v202/li23q.html.

Shiyao Li, Yingchun Hu, Xuefei Ning, Xihui Liu, Ke Hong, Xiaotao Jia, Xiuhong Li, Yaqi Yan, Pei Ran, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. MBQ: Modality-balanced quantization for large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4167–4177, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Li_MBQ_Modality-Balanced_Quantization_for_Large_Vision-Language_Models_CVPR_2025_paper.html.

Yizhi Li, Yinghao Ma, Ge Zhang, Ruibin Yuan, Kang Zhu, Hangyu Guo, Yiming Liang, Jiaheng Liu, Zekun Wang, Jian Yang, Siwei Wu, Xingwei Qu, Jinjie Shi, Xinyue Zhang, Zhenzhu Yang, Yidan Wen, Yanghai Wang, Shihao Li, Zhaoxiang Zhang, Zachary Liu, Emmanouil Benetos, Wenhao Huang, and Chenghua Lin. OmniBench: Towards the future of universal omni-language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. doi: 10.52202/085713-1036.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pp. 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48.

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, 2023. doi: 10.52202/075280-1516.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: on the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024. doi: 10.1007/s11432-024-4235-6.

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 *Advances in Neural Information Processing Systems*, volume 35, pp. 2507–2521, 2022. doi: 10.52202/068431-0182.

OpenBMB. MiniCPM-o 2.6: A GPT-4o level MLLM for vision, speech, and multimodal live streaming on your phone. Hugging Face model card, 2025. URL https://huggingface.co/openbmb/MiniCPM-o-2_6. Accessed September 25, 2026.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. LibriSpeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015. doi: 10.1109/icassp.2015.7178964.

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.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Singh_Towards_VQA_Models_That_Can_Read_CVPR_2019_paper.html.

Wenwen Tong, Hewei Guo, Dongchuan Ran, Jiangnan Chen, Jiefan Lu, Kaibin Wang, Keqiang Li, Xiaoxu Zhu, Jiakui Li, Kehan Li, Xueheng Li, Lumin Li, Chenxu Guo, Jiasheng Zhou, Jiandong Chen, Xianye Wu, Jiahao Wang, Silei Wu, Lei Chen, Hanming Deng, Yuxuan Song, Dinghao Zhou, Guiping Zhong, Ken Zheng, Shiyin Kang, and Lewei Lu. InteractiveOmni: A unified omni-modal model for audio-visual multi-turn dialogue. *arXiv preprint arXiv:2510.13747*, 2025. URL https://arxiv.org/abs/2510.13747.

Zekun Wang, MingHua Ma, Zexin Wang, Rongchuan Mu, Liping Shan, Ming Liu, and Bing Qin. EffiVLM-BENCH: A comprehensive benchmark for evaluating training-free acceleration in large vision-language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25546–25572. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1242.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier Suppression: Pushing the limit of low-bit Transformer language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17402–17414, 2022. doi: 10.52202/068431-1265.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier Suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1648–1665. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.102.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

JingJing Xie, Yuxin Zhang, Mingbao Lin, Liujuan Cao, and Rongrong Ji. Advancing multimodal large language models with quantization-aware scale learning for efficient adaptation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 10582–10591, 2024. doi: 10.1145/3664647.3680838.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, Bin Zhang, Xiong Wang, Yunfei Chu, and Junyang Lin. Qwen2.5-Omni technical report. *arXiv preprint arXiv:2503.20215*, 2025. URL https://arxiv.org/abs/2503.20215.

Yufei Xue, Yushi Huang, Jiawei Shao, and Jun Zhang. VLMQ: Efficient post-training quantization for large vision-language models via Hessian augmentation. *arXiv preprint arXiv:2508.03351v1*, 2025. URL https://arxiv.org/abs/2508.03351v1.

JiangYong Yu, Sifan Zhou, Dawei Yang, Shuoyu Li, Shuo Wang, Xing Hu, Chen Xu, Zukang Xu, Changyong Shu, and Zhihang Yuan. MQuant: Unleashing the inference potential of multimodal large language models via static quantization. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 1783–1792, 2025. doi: 10.1145/3746027.3755433.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9556–9567, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yue_MMMU_A_Massive_Multi-discipline_Multimodal_Understanding_and_Reasoning_Benchmark_for_CVPR_2024_paper.html.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/50623630a2372839c078474efa6c0cb8-Abstract-Conference.html.

A APPENDIX

A.1 REPRODUCIBILITY AND ASSETS

The anonymous supplementary code provides the core SPARCQuant quantization and inference pipeline, model adapters, and C++/CUDA operators. It implements adaptive base-modality selection, sensitivity-guided channel selection, online budget search, scale optimization, and ridge compensation. Quantization and reference inference support Qwen3-VL, MiniCPM-o-2.6, and InteractiveOmni-4B under W8A8, W4A8, and W4A6. The CUDA runtime includes INT8 and packed INT4 operators, sparse compensation, and fused decoder operations. The package also provides configuration recipes, input examples, numerical checks, dependency specifications, and instructions for installation, calibration, checkpoint loading, and inference. Pretrained models and calibration data are supplied by the user.

A.2 CUDA RUNTIME AND MEASUREMENT

The CUDA runtime supports W8A8, W4A8, and W4A6 using one resident quantized weight per decoder projection. W4 weights are packed for storage and consumed directly during decoding; prefill uses temporary unpacking. A6 activations use six-bit quantization bounds in an INT8 container. Smoothing, quantization, and sparse compensation execute on the GPU, while attention uses SDPA.

Deployment measurements follow the protocol in Section 4.1. Each quantized configuration and its BF16 baseline use the same input and generation length. Timing excludes preprocessing and includes CUDA synchronization; results are averaged after warm-up. We report end-to-end latency, prefill and decode times, and full-model allocated and peak GPU memory. Memory savings include the effects of unquantized components and temporary workspaces. Exact runtime settings and validation checks are included in the code.

A.3 ADDITIONAL DETAILS OF SPARCQUANT

A.3.1 CALIBRATION STATISTICS

For each linear module, SPARCQuant collects calibration activations separately for each modality at the module input. For modality m and input channel i, a two-sided quantile interval is used to estimate a robust activation radius:

$$R_{m,i} = \max\left(\frac{\mathrm{Quantile}_q(X_{m,:,i}) - \mathrm{Quantile}_{1-q}(X_{m,:,i})}{2}, \epsilon_R\right), \qquad q = 0.995. \tag{7}$$

The activation-magnitude importance is computed as

$$A_{m,i} = \frac{\frac{1}{T_m}\sum_{t=1}^{T_m}|X_{m,t,i}|}{\frac{1}{d_{\mathrm{in}}}\sum_{j=1}^{d_{\mathrm{in}}}\frac{1}{T_m}\sum_{t=1}^{T_m}|X_{m,t,j}| + \epsilon_A}. \tag{8}$$

The weight channel scale used for smoothing initialization is

$$P_i = \max_o |W_{i,o}|. \tag{9}$$

These statistics are lightweight and require no gradient computation.

A.3.2 COARSE-TO-FINE ONLINE BUDGET SEARCH

For each non-base modality, SPARCQuant searches the sparse budget from a small candidate set. It first evaluates coarse candidates and then refines the search around the best coarse budget. The layer-wise candidate set is adjusted by an online scheduler. Layers are divided into early, middle, and late regions according to depth: early layers use smaller candidate budgets and stronger complexity penalties, middle layers use moderate budgets, and late layers allow larger budgets because accumulated quantization errors may require stronger correction.

SPARCQuant also uses a late-tail trigger. After entering the late region, the first few late layers serve as anchors. If a subsequent late layer has both a significantly larger reconstruction loss and a high near-maximum budget ratio, the scheduler switches the remaining late layers to a more conservative tail budget set. This avoids repeatedly choosing saturated budgets in difficult tail layers.

A.3.3 RIDGE COMPENSATION

For an automatically selected channel set $\mathcal{I}_m$, SPARCQuant fits an affine compensation module. Define

$$Z_m = \left[\widetilde{X}_{m,:,\mathcal{I}_m} \quad \mathbf{1}\right], \qquad E_m = Y_m^{\mathrm{fp}} - \left(\widetilde{X}_m \widehat{W}^0 + b\right).$$

The compensation parameter is obtained by ridge regression:

$$\Theta_m = \left(Z_m^\top Z_m + \lambda_r \Lambda\right)^{-1} Z_m^\top E_m. \tag{10}$$

The first K_m rows of Θ_m form B_m, and the last row forms the affine compensation vector c_m. Because the correction uses only selected input channels, the effective residual weight is row-sparse and has rank at most K_m.

A.4 MINICPM-O-2.6 BASE-MODALITY DISTRIBUTION

To further diagnose the fixed-text failure mode in Table 5, the saved `scsq_base_modality_id` entries are parsed from a MiniCPM-o-2.6 W4A6 SPARCQuant checkpoint. This diagnostic does not rerun calibration or benchmark evaluation; it only summarizes the base modality selected by SPARCQuant for each decoder layer and linear module. MiniCPM-o-2.6 has 28 tracked decoder layers and seven tracked linear projections per layer, yielding 196 layer/module decisions.

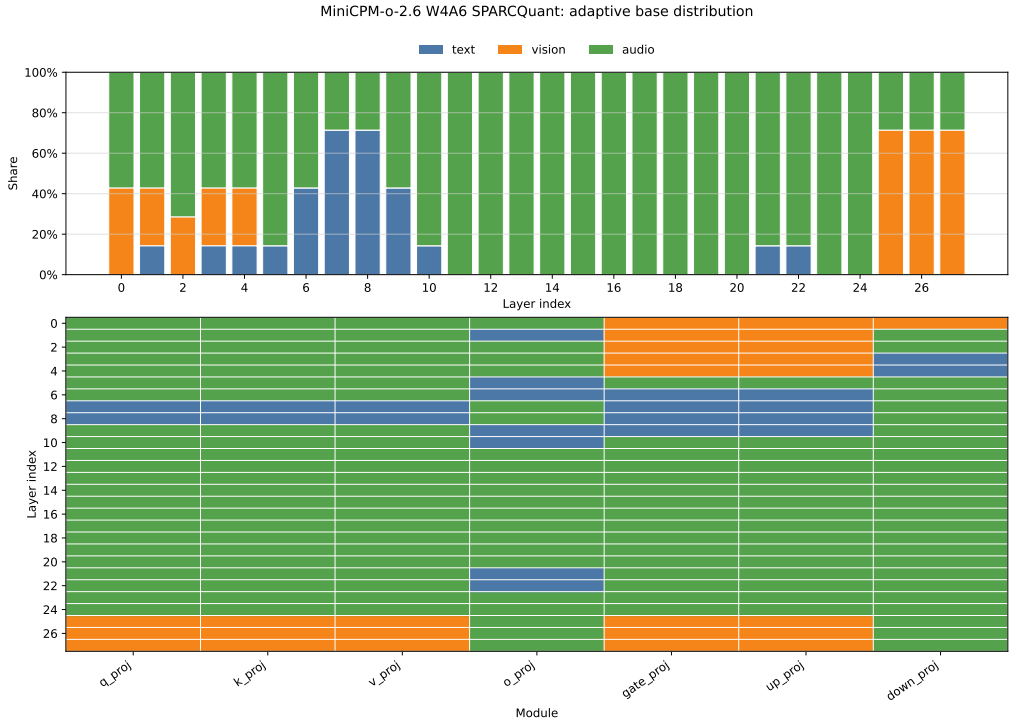


Figure 4: Adaptive base-modality choices in the MiniCPM-o-2.6 W4A6 SPARCQuant checkpoint. The layer-wise stacked bar shows how many of the seven tracked projections select text, vision, or audio as the base modality at each layer. The module-wise heatmap shows the selected base for every layer/projection pair.

The selected bases are strongly non-text. Text is chosen in only 23 of 196 decisions, while vision and audio are chosen in 26 and 147 decisions, respectively. Thus, a fixed text base would override 173 of 196 adaptive choices, or 88.27% of all layer/module decisions. The distribution is not uniform across depth: layers 11–20 and 23–24 select audio for all seven tracked projections, while the text-heavy region is limited to layers 7–8, where five of seven projections select text. Vision is most visible near the tail, peaking at five of seven projections in layer 25.

Table 7: Module-wise adaptive base-modality distribution on MiniCPM-o-2.6 W4A6. Counts are over 28 decoder layers per module. The overall row aggregates all 196 layer/module decisions.

Module	Text	Vision	Audio	Dominant base
`q_proj`	2	3	23	audio
`k_proj`	2	3	23	audio
`v_proj`	2	3	23	audio
`o_proj`	7	0	21	audio
`gate_proj`	4	8	16	audio
`up_proj`	4	8	16	audio
`down_proj`	2	1	25	audio
Overall	23	26	147	audio
Overall ratio	11.73%	13.27%	75.00%	audio

This analysis explains why the fixed-text ablation in Table 5 is especially harmful on MiniCPM-o-2.6 W4A6. In the learned SPARCQuant checkpoint, text is not a neutral reference modality for most modules; forcing text as the base removes the calibration-selected audio and vision bases for the majority of the model. The result supports the layer- and module-adaptive base selection in Eq. (3): the shared smoothing pattern should be chosen from calibration statistics rather than fixed by a text-only prior.

A.5 Budget Search Components

This diagnostic examines the budget-search components on Qwen3-VL-2B W4A6 across five vision-language tasks. The saved checkpoints share the same 128-sample calibration set and are reevaluated with a common inference protocol: batch size four (16 for ScienceQA) and fixed cuDNN convolution selection. The ablations isolate the complexity penalty and coarse-to-fine refinement used to select module-wise supports from calibration reconstruction, as described in Appendix A.3.2.

Table 8: Core budget-search components on Qwen3-VL-2B W4A6. The ablations remove the complexity penalty or coarse-to-fine refinement. Metrics are percentages. Comp. counts compensation matrices and affine biases in millions; Avg. is the unweighted five-task mean. Best scores among the displayed variants are bold.

Variant	Mean K	Comp. (M)	ScienceQA↑	TextVQA↑	VizWiz↑	MMMU↑	OCRBench↑	Avg.↑
Auto w/o penalty	29.24	17.47	45.77	63.48	46.59	32.22	55.70	48.75
Auto w/o coarse-to-fine	20.88	12.12	48.95	63.93	**57.37**	31.33	57.40	51.80
Full auto-budget	20.80	11.94	**60.62**	**64.42**	51.47	**32.78**	**57.50**	**53.36**

Table 8 focuses on the complexity penalty and coarse-to-fine refinement. **The budget penalty reduces mean K from 29.24 to 20.80 and improves average accuracy from 48.75 to 53.36.** Refinement further improves ScienceQA by 11.67 points and the five-task average by 1.56 points over coarse search alone. These results support regularized reconstruction and refinement as the central search components. Appendix A.6 visualizes the resulting depth- and module-dependent allocation; the complete diagnostic results accompany the supplementary code.

A.6 Online Budget Allocation Heatmaps

The budget heatmaps visualize how the online auto-budget scheduler allocates sparse smoothing capacity across depth and modules. This diagnostic does not rerun calibration or evaluation. Instead, it reads the saved W4A6 `full_scsquant` checkpoints and computes each heatmap cell as the number of active entries in the corresponding `scsq_m_scale_mask`. Each value is therefore the selected residual budget $K_{l,n,m}$ for layer l, linear module n, and non-base modality m.

Layers are split using the same default scheduler ratios as SPARCQuant: the early region ends at $\lfloor 0.36L \rfloor$, and the late region starts at $\lceil 0.71L \rceil$. Table 9 summarizes the stage-wise mean selected

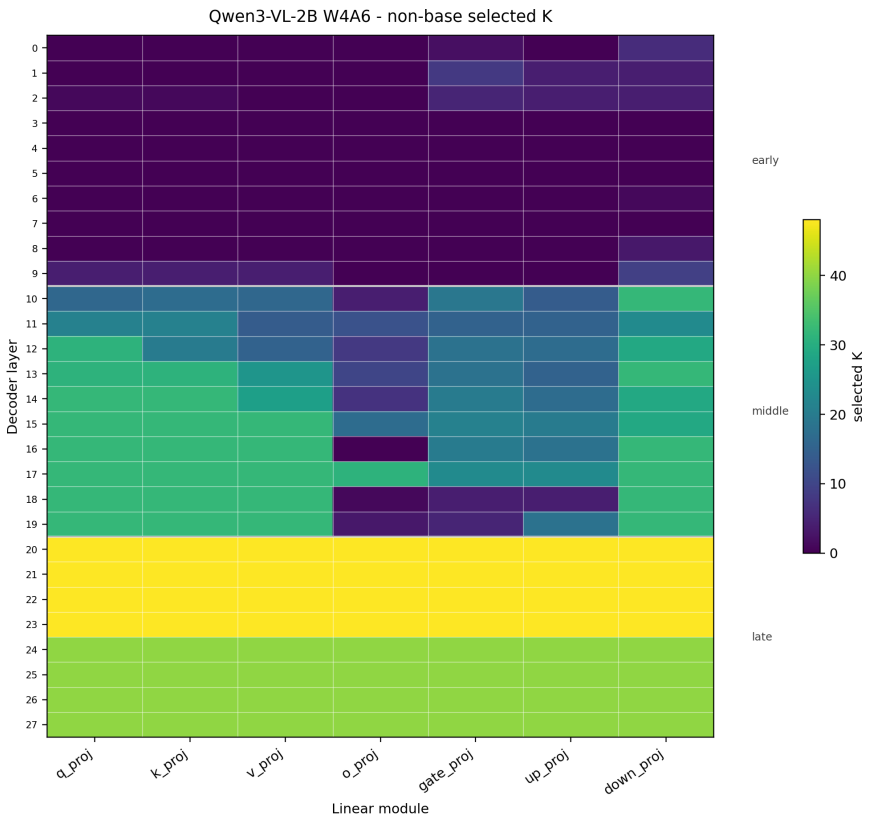


Figure 5: Online budget allocation on Qwen3-VL-2B W4A6. Each cell is the selected K for the non-base modality of a layer/module pair; budgets grow from near-zero in early layers to moderate and large values in middle and late layers.

budget. For Qwen3-VL-2B, the non-base budget increases from 0.91 in early layers to 22.11 in middle layers and 44.00 in late layers. The stricter vision-only view shows the same trend, with nearly zero early vision residual budget and large late-layer repair. MiniCPM-o-2.6 exhibits a similar depth pattern for vision, while audio residual budgets are sparse because audio is often selected as the base modality, as shown in Appendix A.4.

Table 9: Stage-wise selected budget K from the W4A6 SPARCQuant budget heatmaps. Non-missing cells exclude cases where the target modality is already the base modality and therefore has no residual repair branch.

Heatmap view	Early mean K	Middle mean K	Late mean K	Non-missing cells
Qwen3-VL non-base all	0.91	22.11	44.00	70 / 70 / 56
Qwen3-VL vision non-base	0.02	21.33	45.09	47 / 18 / 44
MiniCPM-o vision non-base	0.15	2.29	33.22	59 / 70 / 50
MiniCPM-o audio non-base	0.62	0.00	27.12	32 / 2 / 8

These heatmaps provide a direct sanity check for the auto-budget mechanism. The scheduler does not assign a uniform global support size: it leaves many early modules on the shared base path, allocates moderate budgets where reconstruction difficulty grows, and allows larger supports in late layers. The within-layer variation across `q_proj`, `k_proj`, `v_proj`, `o_proj`, and MLP projections further supports module-wise budget search rather than a manually fixed Top-K for the whole model.

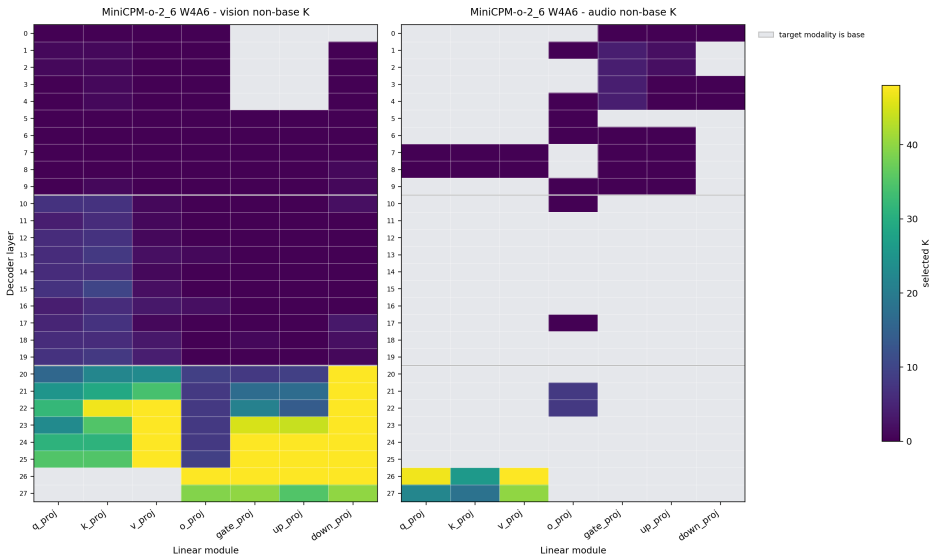


Figure 6: Online budget allocation on MiniCPM-o-2.6 W4A6 for vision and audio residual branches. Gray cells denote cases where the target is the base modality. Vision receives substantial late-layer repair; audio repair is concentrated in the final layers because audio is usually the base.

A.7 Component Ablation Across Speech-Recognition Corpora

Table 10 extends the component ablation to FLEURS-English and LibriSpeech `test-other` using MiniCPM-o-2.6 at W4A6. All variants follow a common evaluation protocol to assess the contribution of sparse channel selection and compensation across speech-recognition corpora. These results are reported separately from the earlier LibriSpeech ablation in Table 5.

Table 10: Component ablation on MiniCPM-o-2.6 W4A6. FLEURS-English and LibriSpeech `test-other` report WER (%); lower is better. All variants use complete test splits and a common inference protocol. Best scores are bold.

Variant	FLEURS↓	LibriSpeech↓
Full SPARCQuant	**29.70**	**24.77**
Fixed K=16	73.40	37.56
Random support	54.47	40.46
Base-only	62.15	49.73
Auto sparse only (no B, c)	54.70	54.04
Auto + B only (no c)	32.17	28.23
Fixed text base	64.11	50.27

The FLEURS comparison supports sensitivity-guided sparse repair. Full SPARCQuant achieves WER 29.70, compared with 54.47 for random support, 54.70 without compensation, and 62.15 for base-only quantization.

On LibriSpeech, full SPARCQuant obtains WER 24.77; the corresponding values are 40.46 for random support, 54.04 without compensation, and 49.73 for base-only quantization. The lowest WER in this comparison is 24.77, obtained by full SPARCQuant.

Paired utterance bootstrapping gives a 95% interval of [14.40, 34.99] WER points for the FLEURS improvement over random support. The incremental FLEURS gain over matrix-only compensation is 2.46 points, with an interval of $[-7.12, 11.51]$. These exploratory intervals quantify test-sample uncertainty without multiple-comparison adjustment.

A.8 Channel-Selection Generalization to Held-Out Inputs

To test whether sparse repair extends beyond the inputs used for fitting, the selector diagnostic in Table 4 is repeated with disjoint source media. For each of three random splits, 64 calibration sequences determine statistics, base modalities, channel supports, and ridge coefficients; 32 separate sequences are used only to evaluate reconstruction MAE. The comparison covers seven decoder layers and five projection types on Qwen3-VL-2B and MiniCPM-o-2.6 under W4A6 and W4A8, with identical K for all selectors. Table 11 reports means and sample standard deviations across the three splits.

Table 11: Channel-selection generalization to disjoint source media. Entries are mean $\pm$ sample standard deviation over three splits. MAE is evaluated only on held-out inputs; percentage columns give its relative reduction against each selector at the same K. Positive reductions favor sensitivity guidance.

Model	Bits	K	Score MAE	vs. range gap (%)	vs. activation (%)	vs. random (%)
Qwen3-VL-2B	W4A6	8	0.2872 ± 0.0033	3.22 ± 0.49	2.38 ± 0.88	5.43 ± 0.87
		16	0.2633 ± 0.0029	5.22 ± 1.39	2.11 ± 0.64	8.82 ± 2.06
		32	0.2478 ± 0.0018	4.95 ± 2.07	2.02 ± 0.50	11.79 ± 1.65
	W4A8	8	0.2521 ± 0.0032	3.35 ± 0.53	1.46 ± 0.89	4.69 ± 1.02
		16	0.2256 ± 0.0028	5.70 ± 1.27	1.43 ± 0.62	8.46 ± 2.38
		32	0.2045 ± 0.0012	5.64 ± 2.15	1.59 ± 0.60	11.99 ± 1.55
MiniCPM-o-2.6	W4A6	8	0.1146 ± 0.0018	2.00 ± 0.33	0.07 ± 0.14	3.74 ± 0.20
		16	0.1137 ± 0.0020	2.68 ± 0.74	0.39 ± 0.07	4.75 ± 0.28
		32	0.1148 ± 0.0025	3.33 ± 1.07	0.39 ± 0.28	5.79 ± 0.41
	W4A8	8	0.0912 ± 0.0011	2.38 ± 0.27	-0.33 ± 0.19	4.20 ± 0.05
		16	0.0899 ± 0.0013	3.15 ± 0.45	0.02 ± 0.09	5.21 ± 0.10
		32	0.0899 ± 0.0016	4.21 ± 0.80	-0.08 ± 0.22	6.40 ± 0.14

Sensitivity guidance retains an advantage over random and range-gap selection on held-out inputs. Every reported mean favors the combined score over these two alternatives. At $K = 32$, the reduction relative to random support is **11.79%/11.99% on Qwen3-VL-2B** and **5.79%/6.40% on MiniCPM-o-2.6** for W4A6/W4A8, respectively. The corresponding reductions over range-gap selection are 4.95%/5.64% and 3.33%/4.21%. The benefit therefore persists when both channel selection and compensation fitting are separated from evaluation inputs.

The comparison with activation-only selection is more nuanced. Qwen3-VL-2B improves across both bit-widths and all three budgets, whereas MiniCPM-o-2.6 remains within 0.40% of that selector and slightly trails it in some W4A8 settings. These results support the generalization of sensitivity-guided sparse repair, with model-dependent gains from combining range discrepancy and activation importance. They measure local reconstruction quality; the downstream task trade-offs are assessed separately in Appendices A.5 and A.7.

A.9 Mechanism Diagnostics on Disjoint Inputs

We present four mechanism diagnostics on Qwen3-VL-2B and MiniCPM-o-2.6 under W8A8, W4A8, and W4A6. Each of three random splits uses 64 fitting sequences and 32 held-out sequences with disjoint source media. The probes cover eight decoder layers and all seven linear projections per layer. Statistics, base choices, supports, budgets, and ridge coefficients use fitting data only. These are isolated linear-module probes on fixed BF16 trajectories, without scale training or accumulated quantization error; they complement the checkpoint and task evaluations above. Compared with Appendix A.8, this suite expands the layer/projection coverage and reports BF16 runtime NMSE rather than reconstruction MAE. Detailed configurations and all per-seed results accompany the code.

For reference output Y and reconstructed output $\widehat{Y}$, we measure $\mathrm{NMSE} = \|\widehat{Y} - Y\|_F^2 / \|Y\|_F^2$. The reference linear operation uses FP32 arithmetic on BF16 activations and weights; evaluated repair branches use BF16 inference arithmetic. We average branch-level metrics within each split and report

means ± sample standard deviations across the three splits. Repair, localization, and search metrics cover non-base branches; calibration metrics include every modality. Random-support replicates are averaged within each split before computing these statistics.

A.9.1 Localization of Quantization Error

To test channel importance independently of the mismatch score itself, we keep the quantized input and shared weight fixed and replace only selected channels' linear contributions with their floating-point counterparts. Table 12 reports the resulting fraction of squared output error removed, computed per branch before averaging. This oracle intervention identifies error-bearing channels; it is not a deployable correction and does not refit the quantizer.

Table 12: Output-error localization at $K = 32$. Entries are the mean percentage of squared error removed by replacing the selected quantized channel contributions with their floating-point contributions (↑). This is an oracle intervention, not an inference method. Random averages three supports; Bottom selects the lowest sensitivity scores.

Model	Bits	Score	Activation	Range gap	Random	Bottom
Qwen3-VL-2B	W8A8	5.78 ± 0.06	6.51 ± 0.03	2.91 ± 0.04	1.43 ± 0.02	1.11 ± 0.01
	W4A8	16.89 ± 0.03	19.67 ± 0.02	5.59 ± 0.24	1.48 ± 0.03	1.01 ± 0.05
	W4A6	13.90 ± 0.07	16.30 ± 0.03	4.70 ± 0.17	1.48 ± 0.03	1.03 ± 0.04
MiniCPM-o-2.6	W8A8	4.90 ± 0.05	5.60 ± 0.01	1.80 ± 0.02	0.79 ± 0.00	0.63 ± 0.02
	W4A8	15.09 ± 0.11	17.88 ± 0.08	4.94 ± 0.11	0.80 ± 0.03	0.63 ± 0.02
	W4A6	11.40 ± 0.12	13.77 ± 0.05	3.36 ± 0.08	0.82 ± 0.02	0.63 ± 0.04

At $K = 32$, the selected channels occupy only 1.41% and 0.79% of the input dimensions on average for Qwen3-VL-2B and MiniCPM-o-2.6, respectively. Under W4A6, sensitivity selection removes **13.90% and 11.40% of the output error**, compared with 1.48% and 0.82% for random supports. The advantage over random and range-gap selection holds in all six configurations. Activation-only selection removes more error in this oracle test, so the result supports error concentration on a small support rather than universal superiority of the combined score for channel-wise error attribution.

A.9.2 Repair Quality per Selected Channel

Figure 7 evaluates actual sparse smoothing and fitted compensation on held-out inputs. All selectors use the same K, fitting inputs, and shared quantized weight. Curves normalize each split's mean runtime NMSE by its uncompensated shared-base value, making the error–support trade-off explicit.

At $K = 32$ under W4A6, sensitivity-guided repair reduces runtime NMSE relative to the shared base by **47.17% on Qwen3-VL-2B and 14.92% on MiniCPM-o-2.6**; the corresponding reductions relative to random support are 23.48% and 2.62%. Compensation is essential when sparse smoothing changes the input scale while retaining the shared weight: removing compensation raises mean NMSE from 0.02407 to 0.07048 on Qwen3-VL-2B and from 0.04199 to 0.07377 on MiniCPM-o-2.6. The combined score gives lower mean NMSE than activation-only selection on Qwen3-VL-2B, while activation-only selection is better on MiniCPM-o-2.6. Larger supports are not uniformly beneficial: MiniCPM-o-2.6 W8A8 at $K = 32$ exceeds the shared-base error by 17.36%. Thus, the measured benefit depends on both the support and the quantization regime.

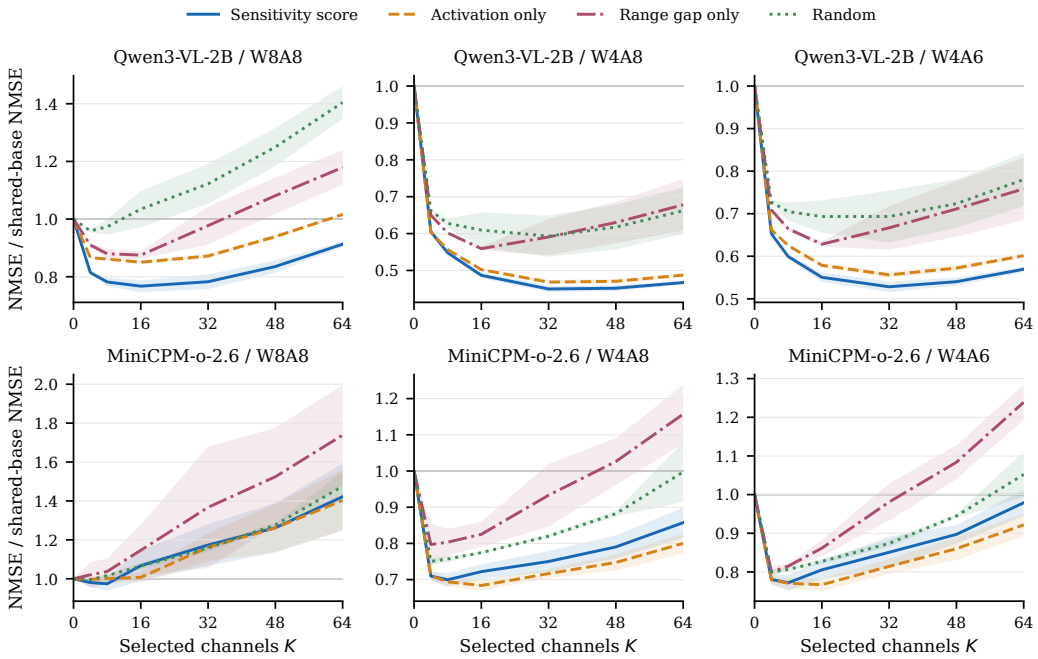


Figure 7: Held-out repair error versus support size on the common comparison grid. Lower is better; the horizontal line is the shared-base error at $K = 0$. Curves are means across three splits, with one sample standard deviation shaded. Random averages three supports per split. These local error and support measurements do not represent end-to-end task accuracy or latency.

A.9.3 Cost and Fidelity of Coarse-to-Fine Search

We compare the production coarse-to-fine routine with exhaustive search over every integer K within the same depth-dependent bounds, using the same penalty and fitting objective. This controlled probe retains the early/middle/late schedule but does not simulate online late-tail feedback from accumulated layer error. Table 13 separates search cost, fitted-objective regret, and held-out runtime error.

Table 13: Coarse-to-fine versus exhaustive integer-budget search under identical stage bounds and penalties. Saved is the reduction in actual candidate evaluations (including repeated coarse evaluations); exhaustive search averages 30 evaluations per branch. Speedup measures isolated search wall time. Match is exact K agreement. Regret is the relative increase in the fitted objective; ΔNMSE is the relative change in held-out runtime NMSE (negative is better). Percentages are in percentage units.

Model	Bits	Saved (%)	Speedup (×)	Match (%)	Regret (%)	ΔNMSE (%)
Qwen3-VL-2B	W8A8	56.75 ± 0.14	1.99 ± 0.01	89.29 ± 1.79	0.136 ± 0.072	0.47 ± 0.77
	W4A8	56.67 ± 0.24	1.97 ± 0.02	99.40 ± 1.03	0.000 ± 0.000	-0.00 ± 0.00
	W4A6	56.51 ± 0.14	1.99 ± 0.02	98.21 ± 1.79	0.026 ± 0.043	0.92 ± 1.73
MiniCPM-o-2.6	W8A8	58.02 ± 0.25	1.83 ± 0.02	98.51 ± 0.52	0.022 ± 0.003	-0.03 ± 0.03
	W4A8	54.44 ± 0.14	1.75 ± 0.01	90.48 ± 5.15	0.022 ± 0.013	0.01 ± 0.08
	W4A6	53.93 ± 0.12	1.75 ± 0.01	88.10 ± 2.73	0.068 ± 0.009	-0.03 ± 0.15

Coarse-to-fine search reduces candidate evaluations by 53.93–58.02% and achieves a 1.75–1.99× search-time speedup. Exact budget agreement ranges from 88.10% to 99.40%; the mean relative fitted-objective regret is at most 0.14% across the six configurations. Mean held-out NMSE changes range from -0.03% to $+0.92\%$. These results support the efficiency and fidelity of the

search approximation within its candidate bounds; the timing excludes the rest of calibration and model inference.

A.9.4 Sensitivity to Calibration Size

We recompute statistics, bases, supports, and compensation using the first 16, 32, or 64 sequences of each fitting split while keeping its held-out inputs fixed. Table 14 measures agreement with the 64-sequence reference as well as reconstruction quality. Support overlap excludes pairs of empty base-modality supports, which would otherwise inflate stability.

Table 14: Calibration-size sensitivity with the same held-out inputs and $K = 32$ on non-base branches. Base agreement and support Jaccard (%) use the 64-sequence setting as reference; its self-comparison is omitted. Jaccard excludes branches that are base modalities in both settings and assigns zero when only one support is empty. NMSE ($\times 10^3$, $\downarrow$) includes all modalities. Choices are bit-independent because statistics use BF16 trajectories.

Model	N_{fit}	Base (%)	Jaccard (%)	W8A8	W4A8	W4A6
Qwen3-VL-2B	16	97.02 ± 2.73	73.59 ± 4.43	0.566 ± 0.003	20.344 ± 0.146	26.358 ± 0.148
	32	97.02 ± 3.72	78.84 ± 6.83	0.560 ± 0.026	20.288 ± 0.771	26.339 ± 0.834
	64	–	–	0.566 ± 0.006	20.358 ± 0.335	26.433 ± 0.369
MiniCPM-o-2.6	16	91.67 ± 3.72	64.37 ± 3.01	1.345 ± 0.079	29.050 ± 0.580	41.101 ± 0.529
	32	95.24 ± 1.03	75.78 ± 1.92	1.275 ± 0.066	27.773 ± 0.474	40.145 ± 1.065
	64	–	–	1.338 ± 0.115	27.948 ± 0.818	40.896 ± 1.488

With 32 sequences, base choices agree with the 64-sequence reference in 97.02% of Qwen3-VL-2B modules and 95.24% of MiniCPM-o-2.6 modules. The corresponding support Jaccard values are 78.84% and 75.78%. Mean W4A6 NMSE is 0.02634 versus 0.02643 on Qwen3-VL-2B and 0.04015 versus 0.04090 on MiniCPM-o-2.6 for 32 versus 64 sequences. The base choices are relatively stable, while exact supports remain more sensitive to calibration composition; increasing the fitting set does not monotonically reduce held-out error.