ProQuant: Progressive Quantization-aware Training for Edge MLLMs

Yufei Xue[1] **Yushi Huang**[1] **Jiawei Shao**[2,3*] **Pingcheng Dong**[1]
Yonghao Tan[1] **Shiyao Li**[4] **Kwang-Ting Cheng**[1] **Jun Zhang**[1*]
[1]The Hong Kong University of Science and Technology
[2]Northwestern Polytechnical University
[3]Institute of Artificial Intelligence (TeleAI), China Telecom
[4]Tsinghua University
jiawei.shao@nwpu.edu.cn, eejzhang@ust.hk

Abstract

Multi-modal large language models (MLLMs) with impressive perception and understanding capabilities will enable a range of downstream applications. However, their substantial computational and memory requirements hinder the real-world implementation, particularly for resource-constrained edge devices. Quantization has emerged as an effective solution for deploying large models. While post-training quantization (PTQ) successfully retains the performance at moderate bit-width (*e.g.*, W8A8), it suffers from severe performance degradation under low-bit quantization (*e.g.*, W4A8), especially for small-scale models that are more suitable for edge deployment. In this paper, we propose ProQuant, a novel weight-activation quantization-aware training (QAT) framework tailored for low-bit MLLMs. First, to align the limited finetuning dataset distribution with the pretrained dataset distribution, we propose a self-relabeling (SRL) scheme to resample the responses of the open-source multimodal dataset. Second, we are the *first* to take a decoupled view of multi-modal (MM) understanding and language modeling abilities. We propose a progressive block activation (PBA) mechanism to prioritize the MM understanding recovery. We also introduce a vision-sensitive factor $\vec{\tau}$ that effectively identifies the MM-critical layers. Extensive experiments across various benchmarks, covering models with 2B$\sim$8B parameters, show that ProQuant significantly outperforms existing methods. For example, our W4A8 Qwen3-VL-2B-Instruct improves average accuracy by impressively 6.6% and achieves performance comparable to full precision counterparts with only $\sim$1% performance drop. Code is available at https://github.com/yufeixue-ai/ProQuant/.

1 Introduction

Extending large language models (LLMs) beyond text to additional modalities [Bai et al., 2025a; Li et al., 2024a; Xiaomi, 2025; Zhu et al., 2025; Wu et al., 2024; Team, 2026; Li et al., 2026a] has achieved remarkable successes and enabled a wide range of applications, including chatbots [Bai et al., 2025a], autonomous driving [Tian et al., 2024], and AR/VR glasses [Chen et al., 2024]. However, the large memory footprint and substantial computational overhead [Huang et al., 2026; Lv et al., 2025] of multi-modal LLMs (MLLMs) present major obstacles to deployment in resource-constrained environments, especially on edge devices [An et al., 2026; Shao and Li, 2025; Yuan et al., 2025]. Quantization [Gholami et al., 2022] alleviates this issue by reducing the precision of both weights and activations, thereby accelerating memory access and computation. Post-training quantization

*Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

(PTQ) [Xiao et al., 2023; Ashkboos et al., 2024; Liu et al., 2024a; Huang et al., 2024a, 2025a; Gong et al., 2024, 2025] typically delivers near-lossless performance at moderate precision (e.g., W8A8), yet its accuracy degrades markedly when pushed to lower-bit settings (e.g., W4A4/W4A8). Additionally, small-scale models (2B/4B), which are more practical for edge deployment, typically suffer more pronounced performance degradation after quantization [Chen et al., 2025a].

In contrast, quantization-aware training (QAT) that incorporates quantization during training has emerged as a promising solution to recover the model performance under low-bit quantization settings [Liu et al., 2023; Du et al., 2024; Chen et al., 2025a; Huang et al., 2025b]. The QAT solutions for LLM primarily target the training data corpus [Liu et al., 2023], training efficiency [Chen et al., 2025b], or distillation paradigm [Du et al., 2024]. In the context of MLLM QAT, additional challenges arise. First, preserving the zero-shot generalization of compressed models hinges on high-quality fine-tuning data that are sufficiently broad and well-aligned with the original pre-training distribution. However, open-source datasets are often domain-narrow or distribution-shifted [Liu et al., 2023], and multi-modal (MM) data are far more difficult to collect than text-only data [Guo et al., 2025]. Second, quantization degrades both MM understanding and language modeling capabilities. A natural question is whether these two capabilities should be recovered jointly, or whether one should be prioritized during training.

Considering the challenges discussed above, we propose `ProQuant`, a novel QAT framework tailored to small-scale MLLMs, as illustrated in Figure 1. First, we propose a self-relabeling (SRL) scheme that re-annotates the training set to better align the fine-tuning data distribution with the model distribution. Second, we introduce a progressive block activation (PBA) mechanism that prioritizes recovering the MM understanding ability by first enabling quantization for vision-sensitive layers while keeping the remaining blocks in full precision during the forward pass. As training progresses, we gradually activate quantization for increasingly text-sensitive layers, thereby restoring language modeling capability. The core contributions are summarized as follows:

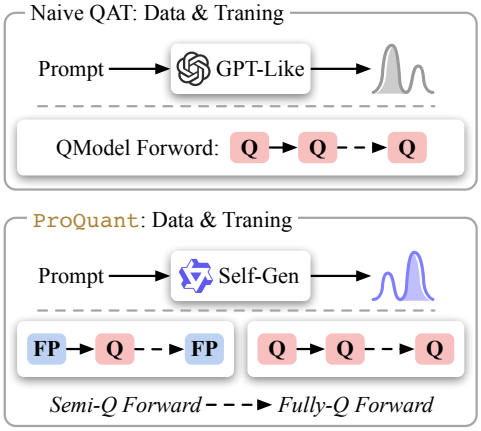


Figure 1: Naive QAT *v.s.* `ProQuant`.

- ▷ We propose `ProQuant`, a dedicated QAT framework for MLLMs. To the best of our knowledge, it is *the first* weight-activation QAT framework for MLLMs, enabling effective low-bit (e.g., W4A8/W4A4) quantization.
- ▷ To address the scarcity of high-quality MM data, we introduce the SRL scheme, which re-annotates the fine-tuning data to better match the model distribution, thereby reducing data mismatch and improving post-quantization generalization.
- ▷ We take a disentangled view of MM understanding and language modeling capabilities during quantization. Particularly, we propose the PBA mechanism to prioritize recovering MM understanding first, and then gradually restore language modeling ability, instead of jointly optimizing both from the outset.
- ▷ Extensive experiments show that `ProQuant` achieves state-of-the-art (SOTA) performance across various models and quantization settings. Notably, `ProQuant` attains near-lossless compression ($\sim$1% performance drop) under W4A8 quantization and substantially outperforms baselines under W4A4, where PTQ methods often collapse.

2 Related Works

2.1 Post-training quantization for (M)LLMs.

PTQ has become a widely adopted technique for compressing and accelerating large models. In the context of LLMs, SmoothQuant-style methods [Xiao et al., 2023; Lin et al., 2023; Shao et al., 2023] mitigate activation outliers in transformer blocks via smoothing-based transformations. More recently, incoherence-processing-based approaches [Ashkboos et al., 2024; Liu et al., 2024a; Hu et al., 2025] reshape weight distributions into more quantization-friendly ones, albeit at the cost of introducing online activation rotations. An orthogonal line of work, namely Hessian-informed techniques [Frantar

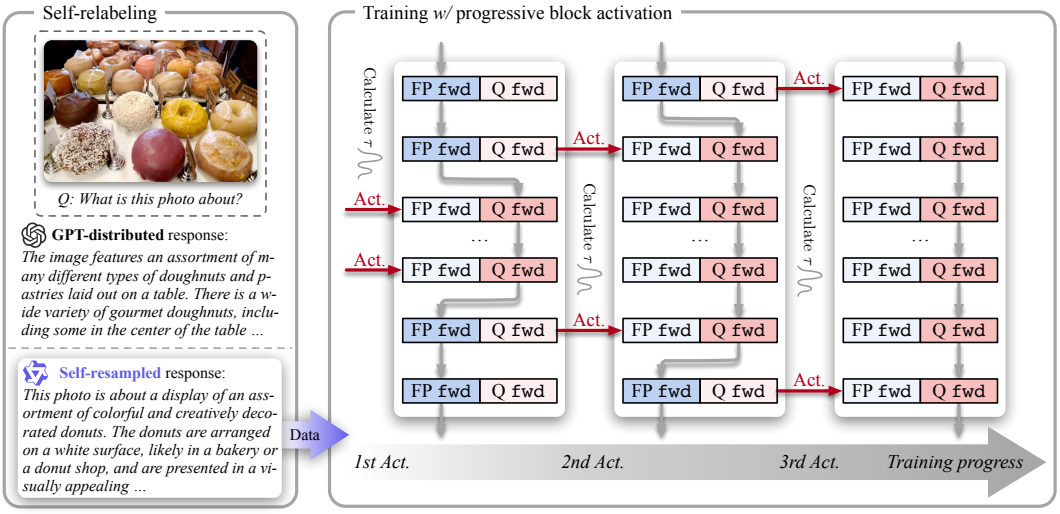


Figure 2: Overview of the proposed `ProQuant`. Left: examples of responses from `GPT-4` [Bubeck et al., 2023] and re-sampled via the SRL scheme (Section 3.1) using `Qwen3-VL-2B-Instruct` (see more details in Appendix B). Right: QAT with the PBA mechanism, where quantization is first activated for decoding layers with the larger vision-sensitivity factor τ. As training proceeds, quantization is gradually enabled for the remaining layers in descending order of τ (Section 3.2).

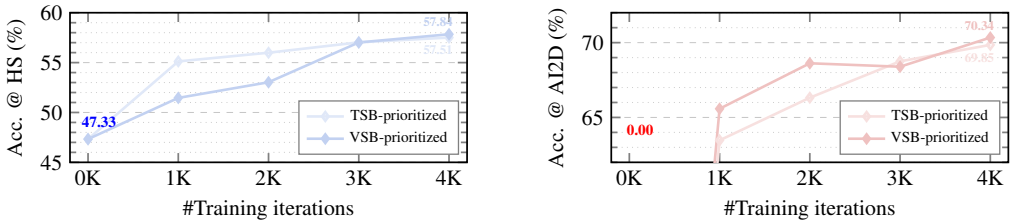
et al., 2022; Li et al., 2025a], performs iterative weight quantization with closed-form updates. However, PTQ for VLMs is considerably more challenging due to heterogeneous features and mismatched distributions between text and vision modalities. Recent efforts such as MBQ [Li et al., 2025b] and VLMQ [Xue et al., 2025] attribute performance degradation to modality discrepancy and apply asymmetric treatments to vision and text tokens during quantization. MQuant [Yu et al., 2025] further extends rotation-based quantization by introducing modality-specific static activation quantization and attention-invariant flexible switching to better handle outliers.

2.2 Quantization-aware training for (M)LLMs.

QAT can further improve quantized models' accuracy beyond PTQ, particularly under low-bit settings. LLM-QAT [Liu et al., 2023] pioneers the QAT paradigm for language transformers. EfficientQAT [Chen et al., 2025b] improves training efficiency via block-wise initialization of quantization parameters. BitDistiller [Du et al., 2024] and SPEED-Q [Guo et al., 2025] investigate knowledge distillation within the quantization context. Other learning-based works [Wang et al., 2024a; Xie et al., 2024] enhance quantized performance by exploiting cross-layer dependencies and refining the data corpus, respectively. However, except for LLM-QAT, the above methods primarily target weight-only quantization. To the best of our knowledge, no previous work has systematically explored weight-activation quantization for MLLMs. See Appendix A.1 for more details.

3 `ProQuant`

In this section, we present `ProQuant`, a dedicated QAT framework for MLLMs that improves both the fine-tuning data distribution and the training pipeline. Specifically, 1) to obtain high-quality fine-tuning data that are closely aligned with the pre-training distribution, we discard the original labels in open-source MM datasets and resample responses from the model to quantize. Then, 2) We adopt a disentangled view of MM understanding and language modeling in MLLMs, and prioritize recovering MM understanding first via the proposed PBA mechanism. The vision-sensitive modules in the LLM backbone are identified using our proposed factor $\vec{\tau}$.



(a) Language modeling capability.

(b) MM understanding capability.

Figure 3: Accuracy of `Qwen3-VL-2B-Instruct` W4A8 checkpoints on HellaSwag [Zellers et al., 2019] (left) and AI2D [Kembhavi et al., 2016] (right) benchmarks over training iterations.

3.1 Data Augmentation via Self-Relabeling

SRL scheme. To align the distribution of the limited fine-tuning data with the pretrained data distribution, we resample responses for open-source MM datasets, which we refer to as the SRL scheme. Specifically, we re-generate responses with the target model using default sampling parameters until reaching either the `[EOS]` token or the maximum generation length. In Figure 2 (left), We present responses of `GPT-4` [Bubeck et al., 2023] and `Qwen3-VL-2B-Instruct` triggered by prompts from the COCO caption dataset [Liu et al., 2024b]. The two sets of responses exhibit distinct styles, suggesting a distribution gap between the original labels and the target-model outputs. In particular, GPT-style [Bubeck et al., 2023] captions tend to be more literal and position-aware (e.g., "*in the center of the table*"), whereas the Qwen-series MLLMs [Bai et al., 2025a] produce captions that are more holistic, semantically enriched, and interpretive (e.g., "*creatively decorated donuts*"). Such a stylistic mismatch can introduce inconsistent supervision during training, and the resampled distribution better aligns with the pretrained data distribution of models to quantize.

Effectiveness of SRL. We further evaluate quantized models trained on datasets produced by different labelers in Section 4.3. The results show that self-generated labels consistently achieve the best performance, outperforming both a stronger closed-source labeler and a homogeneous larger labeler. Appendix D.4 further compares target-model likelihood, vocabulary coverage, response length, formatting, and semantic similarity on paired cases. These analyses support target-distribution alignment beyond a single stylistic factor and characterize the limits of cross-model label transfer.

3.2 QAT with Progressive Block Activation

Motivation. Quantization degrades both MM understanding and language modeling in MLLMs, making QAT substantially more challenging than for text-only LLMs [Liu et al., 2023; Chen et al., 2025b]. Naive QAT attempts to recover these two capabilities jointly, yet yields suboptimal results (see Section 4 for details). We therefore decouple MM understanding and language modeling during training via the proposed PBA mechanism. As illustrated in Figure 3, initial evaluations of the round-to-nearest baseline (0-th iteration) reveal a stark performance disparity: while the quantized backbone retains a non-trivial linguistic prior (47.33% accuracy on HellaSwag), its MM understanding collapses entirely (0% on AI2D). This phenomenon suggests that cross-modal alignment is significantly more vulnerable to quantization noise than the model's intrinsic generative inertia. Since MM understanding is the prerequisite anchor for grounded reasoning, we propose to prioritize the restoration of the MM understanding path before progressively refining the more text-sensitive modules.

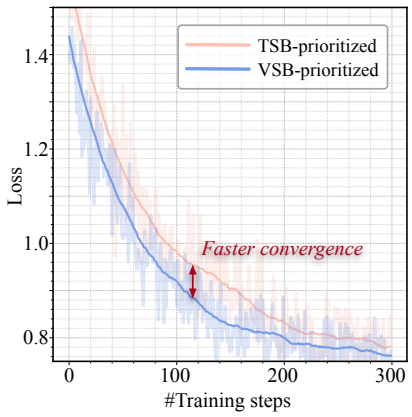


Figure 4: Loss trajectories of TSB/VSB-prioritized QAT over the first 300 steps.

Vision sensitivity guided PBA mechanism. We propose PBA to prioritize quantization recovery for the vision-sensitive blocks (VSBs) in the LLM backbone. To identify VSBs, we introduce a vision-sensitivity factor $\vec{\tau}$. Specifically, for the i-th block, we denote the output gradient as

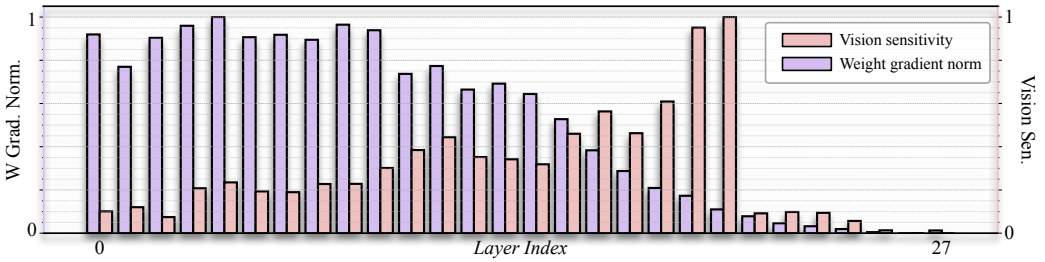


Figure 5: Normalized vision sensitivity $\vec{\tau}$ and weight gradient norm $\|\mathbf{G}_w\|_{\ell_1}$ across decoding layers.

$\mathbf{G}^i = \left[\mathbf{G}_v^i;\ \mathbf{G}_t^i\right] \in \mathbb{R}^{K\times N}$, where $\mathbf{G}_v^i \in \mathbb{R}^{K_v\times N}$ and $\mathbf{G}_t^i \in \mathbb{R}^{K_t\times N}$ correspond to the gradients of vision and text tokens, respectively. We define the vision-sensitivity factor τ^i as the ratio between the average token-wise gradient magnitude over vision tokens and that over all tokens, i.e.,

$$\tau^i = \frac{K}{K_v}\frac{\sum_{k=1}^{K_v}\left\|\mathbf{G}_v^i[k,:]\right\|_{\ell_1}}{\sum_{j=1}^{K}\left\|\mathbf{G}^i[j,:]\right\|_{\ell_1}}. \tag{1}$$

Figure 5 visualizes $\vec{\tau} = \left[\tau^0,\ \tau^1,\ \ldots,\ \tau^{L-1}\right]$ as red bars for `Qwen3-VL-2B-Instruct` with $L = 28$ decoding layers. We observe that VSBs are concentrated in the middle layers, whereas text-sensitive blocks (TSBs) are more prevalent in both shallow and deep layers. Accordingly, we schedule the quantization activation points such that VSBs are quantized first, and TSBs are gradually enabled later in training. This process is illustrated in Figure 2 (right).

Effectiveness of τ. We demonstrate the effectiveness of the proposed vision-sensitivity factor $\vec{\tau}$ from two perspectives: convergence behavior and checkpoint performance. We first compare the loss trajectories of the first 300 training steps under TSB-prioritized and VSB-prioritized schedules in Figure 4. The VSB-prioritized schedule converges faster, indicating that prioritizing multimodal understanding is beneficial for optimization. We further report checkpoint performance from 0K to 4K iterations in Figure 3. As shown in Figure 3a, the VSB-prioritized schedule restores language modeling more slowly at early stages than the TSB-prioritized one, but surpasses it by 4K iterations. Meanwhile, in Figure 3b, the VSB-prioritized schedule recovers MM understanding substantially faster and outperforms the TSB-prioritized schedule at nearly all checkpoints. Overall, $\vec{\tau}$-guided PBA effectively prioritizes the recovery of MM understanding over language modeling, leading to faster early-stage convergence and better final performance.

Effectiveness of PBA. We further validate the effectiveness of PBA by comparing the full training trajectories in Figure 6 against a naive QAT baseline. Under both W4A8 and W4A4 settings, PBA approaches the pre-trained loss level more rapidly during the warm-up stage. Once new block quantization is activated, the loss quickly descends below the naive baseline, despite tolerable spikes naturally introduced by the staged quantization. In the subsequent post-hoc recovery stage, PBA attains a better optimum as well. This advantage is further corroborated by the performance gains reported in Section 4.

4 Experiments

4.1 Implementation Details

Models and datasets. We conduct experiments on small-scale models, including `Qwen3-VL-2B-Instruct` [Bai et al., 2025a], `Qwen2.5-VL-3B-Instruct` [Bai et al., 2025b], and `Qwen2-VL-2B-Instruct` [Wang et al., 2024b]. To demonstrate the generalization of `ProQuant`, we further extend our evaluation to a large model, `Qwen3-VL-8B-Instruct`. The training data are adapted from `LLaVA-680K` [Liu et al., 2024b].

Baselines. We compare the proposed `ProQuant` with strong PTQ and QAT baselines. Specifically, we consider outlier-smoothing methods, including MBQ [Li et al., 2025b] and SmoothQuant [Xiao et al., 2023], the rotation-assisted approach MQuant [Yu et al., 2025], as well as the QAT solution LLM-QAT [Liu et al., 2023]. PTQ baselines use their official implementations and paper configurations. We

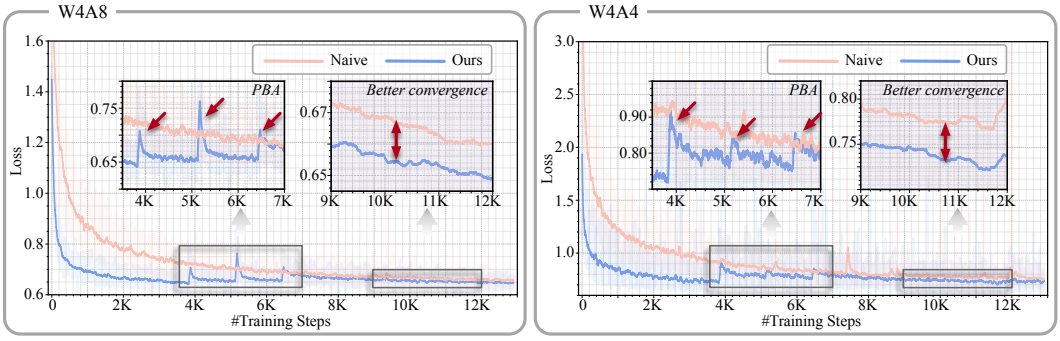


Figure 6: Training loss comparison of `Qwen3-VL-2B-Instruct` between naive QAT and `ProQuant` under W4A8 and W4A4. PBA exhibits faster warm-up convergence toward the pre-trained level, rapid recovery during PBA, and better final convergence.

Table 1: Weight-activation quantization results for Qwen VL series. The headers represent model architectures, bitwidth, quantization methods, AI2D, ChartQA, DocVQA, MMMU, ScienceQA, SeedBench, TextVQA, average accuracy, and MME scores, respectively. The superscript † indicates `down_proj` activations are quantized to 8-bit under W4A4 settings.

Arch	#BW	Method	AI2D	ChQ	DocV	M3U	SciQ	Seed	TxtV	Avg. (↑)	MME
Qwen2-VL-2B	BF16	-	70.04	72.16	89.40	39.89	74.23	69.98	79.38	70.73	1870.19
	W4A8	SmoothQ	62.66	53.68	76.59	34.00	66.47	63.20	69.17	60.82	1736.61
		MBQ	65.12	62.08	79.76	35.22	68.99	64.64	71.91	63.96	1810.68
		MQuant	67.36	**70.44**	87.39	38.33	–	–	77.72	–	–
		LLM-QAT	66.94	68.20	87.14	35.56	69.37	68.27	77.94	67.63	**1931.70**
		`ProQuant`	**69.30**	70.32	**88.10**	**39.00**	**72.79**	**69.19**	**78.60**	**69.61**	1830.28
	W4A4	MBQ	2.17	0.00	0.00	23.67	1.53	8.38	0.00	5.11	55.41
		MQuant	11.30	0.36	0.16	12.56	–	–	0.18	–	–
		LLM-QAT†	29.95	39.68	66.74	27.56	43.10	40.68	69.94	45.38	1532.13
		`ProQuant`	**40.97**	**47.32**	**72.27**	**30.11**	**56.99**	**61.07**	**72.40**	**54.45**	**1611.94**
Qwen2.5-VL-3B	BF16	-	78.63	83.32	92.50	47.00	83.33	70.94	78.61	76.33	2149.00
	W4A8	SmoothQ	71.83	74.36	79.65	40.44	73.92	66.90	63.91	67.29	2063.23
		MBQ	73.87	79.24	88.44	40.44	77.18	69.27	74.00	71.78	2049.63
		LLM-QAT	54.86	72.76	62.58	33.00	53.24	49.07	71.35	56.69	1939.38
		`ProQuant`	**77.27**	**82.68**	**91.26**	**44.11**	**81.61**	**70.80**	**78.21**	**75.13**	**2083.30**
	W4A4	MBQ	0.94	0.00	0.51	24.33	0.87	5.35	0.02	4.57	11.24
		LLM-QAT†	31.22	44.96	**66.40**	27.56	45.84	24.99	64.75	43.67	1090.88
		`ProQuant`	**48.48**	**51.00**	64.53	**30.67**	**61.64**	**61.56**	**67.65**	**55.08**	**1520.12**
Qwen3-VL-2B	BF16	-	73.25	79.56	92.70	38.78	79.58	72.26	79.26	73.63	2029.18
	W4A8	SmoothQ	48.64	61.08	80.84	33.00	40.18	64.91	64.82	56.21	1834.91
		MBQ	50.62	67.44	84.20	32.78	50.51	65.75	69.40	60.10	1758.88
		LLM-QAT	60.52	65.88	86.11	38.89	69.02	67.45	73.71	65.94	**1962.77**
		`ProQuant`	**71.34**	**75.44**	**91.10**	**41.22**	**78.73**	**71.90**	**78.08**	**72.54**	1962.04
	W4A4	MBQ	5.18	0.00	0.10	25.56	6.65	7.03	0.08	6.37	0.00
		LLM-QAT†	19.95	28.68	25.39	26.33	30.91	15.94	43.82	27.29	764.33
		`ProQuant`	**54.86**	**54.12**	**73.23**	**33.11**	**64.84**	**67.47**	**71.18**	**59.83**	**1652.63**

adapt LLM-QAT to multimodal inputs using the same training data, budget, and optimization settings as `ProQuant`; the marked W4A4 cases retain 8-bit `down_proj` activations. Further implementation details are provided in Appendix C.

Evaluation. To evaluate quantized models, we report results on a diverse suite of benchmarks built on LMMs-Eval [Zhang et al., 2024]: MME [Fu et al., 2023] for comprehensive multi-modal evaluation; AI2D [Kembhavi et al., 2016] and TextVQA for text recognition; ChartQA [Masry et al., 2022] for chart parsing; DocVQA [Mathew et al., 2021] for document understanding; MMMU [Yue et al., 2024] and ScienceQA [Lu et al., 2022] for visual reasoning; and SeedBench [Li et al., 2023a] for spatial–temporal image/video understanding.

Table 2: W8A8 quantization results on three model backbones. Values in red denote the corresponding full-precision (BF16) performance.

Method	AI2D	ChQ	DocV	M3U	SciQ	Seed	TxtV	Avg. (↑)	MME
`Qwen2-VL-2B-Instruct`								70.73	1870.19
MBQ	**69.88**	**72.28**	86.89	**39.33**	**74.09**	68.82	77.13	69.77	**1884.59**
`ProQuant`	69.66	72.08	**88.91**	38.89	73.21	**69.88**	**79.34**	**70.28**	1864.33
`Qwen2.5-VL-3B-Instruct`								76.33	2149.00
MBQ	78.27	83.00	**92.69**	46.78	**82.55**	**71.12**	**78.60**	76.14	**2156.88**
`ProQuant`	**78.30**	**83.52**	92.25	**48.11**	82.34	71.08	78.42	**76.29**	2102.85
`Qwen3-VL-2B-Instruct`								73.63	2029.18
MBQ	**75.00**	**79.96**	**92.80**	38.99	**79.56**	71.98	**79.35**	**73.95**	**2009.60**
`ProQuant`	72.25	78.64	92.35	**39.22**	76.75	**72.28**	78.65	72.88	1996.02

4.2 Performance Analysis

4.2.1 Performance under W4A8 & W4A4 quantization.

We report the performance of quantized models on a diverse set of MM benchmarks in Table 1. Overall, `ProQuant` consistently outperforms both PTQ and QAT baselines across all model backbones and quantization settings, with the advantage becoming especially significant in the more aggressive W4A4 regime.

Under W4A8 quantization, `ProQuant` preserves the performance of the full-precision models remarkably well. Across all three backbones, the average accuracy of `ProQuant` remains within approximately one point of the BF16 counterparts (i.e., around $\sim 1\%$ absolute degradation), indicating near-lossless quantization in this setting. In contrast, existing PTQ and QAT baselines exhibit noticeably larger drops on multiple benchmarks, even when their performance is competitive on a subset of metrics.

The performance gap becomes much larger under W4A4 quantization. In this regime, PTQ baselines (e.g., MBQ and MQuant) largely collapse, with near-zero or extremely low scores on multiple benchmarks, indicating that they fail to maintain usable multi-modal reasoning and perception abilities at 4-bit activation quantization. The LLM-QAT baseline is substantially more robust than PTQ, but still suffers severe degradation; in particular, it can only maintain reasonable performance when `down_proj` activations are kept at 8-bit. Even with this relaxation, its performance remains far below `ProQuant` in most cases.

4.2.2 Performance under W8A8 quantization.

Under the relatively mild W8A8 setting, both PTQ approaches and `ProQuant` achieve performance comparable to BF16 across all three backbones, indicating that strong PTQ methods are already sufficient in this regime and that the extra training cost of QAT offers limited cost-effectiveness. In contrast, the advantage of `ProQuant` becomes much more pronounced in lower-bit settings (e.g., W4A8/W4A4) as demonstrated in Table 1, making it more suitable for aggressive low-bit quantization where performance preservation is substantially more challenging.

4.2.3 Compatibility on weight-only quantization.

Results in Table 3 show that `ProQuant` generalizes well beyond W/A quantization and remains effective in weight-only (group-wise) quantization settings. Specifically, under both W4g128 and the more aggressive W3g128 configurations, `ProQuant` consistently outperforms the PTQ baseline on all benchmarks and yields clear gains in both average accuracy and MME. The advantage is particularly significant at W3g128, where MBQ suffers severe degradation while `ProQuant` still preserves strong MM performance, demonstrating the robustness and generalizability of our method to low-bit weight-only quantization.

4.2.4 Scale to large models.

Results on `Qwen3-VL-8B-Instruct` in Table 4 show that `ProQuant` generalizes well beyond the small-sized models targeted in our main experiments. It remains highly effective on this large

Table 3: Weight-only quantization results of MBQ and ProQuant under W4g128 and W3g128 settings.

#BW	Method	AI2D	ChQ	DocV	M3U	SciQ	Seed	TxtV	Avg. (↑)	MME
W4g128	MBQ	64.35	74.88	91.23	36.67	68.57	70.22	76.94	68.98	1955.42
	ProQuant	**71.76**	**76.44**	**91.26**	**38.67**	**77.08**	**72.11**	**77.97**	**72.18**	**1958.53**
W3g128	MBQ	32.25	61.00	76.21	28.00	46.59	62.91	65.48	53.21	1350.49
	ProQuant	**65.90**	**71.92**	**88.29**	**35.67**	**68.45**	**70.23**	**75.60**	**68.01**	**1922.82**

Table 4: W/A quantization results on Qwen3-VL-8B-Instruct.

#BW	Method	AI2D	ChQ	DocV	M3U	SciQ	Seed	TxtV	Avg. (↑)	MME
FP	-	83.71	84.80	95.53	50.22	92.88	75.31	82.48	80.70	2381.20
W4A8	MBQ	77.62	75.40	92.71	45.78	87.31	72.55	78.00	75.62	2162.47
	ProQuant	**82.19**	**84.12**	**95.20**	**49.33**	**91.44**	**75.16**	**81.18**	**79.80**	**2325.49**
W4A4	MBQ	2.20	0.00	0.03	26.22	3.56	8.30	0.00	5.76	0.00
	ProQuant	**62.44**	**57.36**	**88.22**	**40.89**	**76.77**	**67.53**	**75.32**	**66.93**	**1877.45**

backbone, consistently outperforming the PTQ baseline under both W4A8 and W4A4 quantization. In particular, ProQuant preserves near-full-precision performance at W4A8 and maintains strong MM capability even under the aggressive W4A4 setting, while MBQ collapses. These results demonstrate the strong scalability and generalization ability of ProQuant across model sizes.

4.2.5 Mixed-precision quantization.

As pointed out in QAT scaling law [Chen et al., 2025a], persistent outliers in down_proj activations are a major bottleneck for low-bit QAT. Consistent with this observation, our mixed-precision results in Table 5 show that keeping down_proj activations at 8-bit (W4A4†) consistently improves performance over full W4A4 quantization.

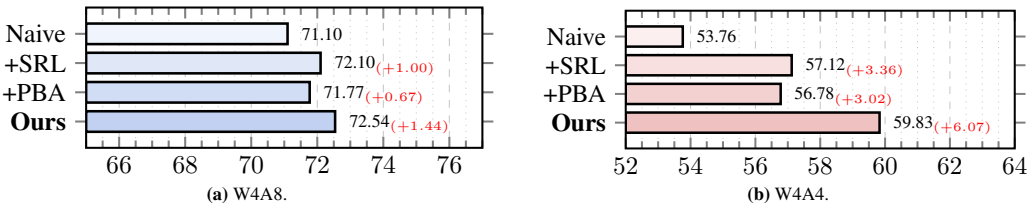
However, the gain varies substantially across backbones (*e.g.*, +1.44, +9.39, and +3.21 Avg. points), suggesting that the benefit of mixed-precision design is relatively model-dependent. Therefore, mixed-precision quantization likely requires a more fine-grained, backbone-aware treatment. Here we provide only preliminary evidence and leave a systematic analysis to future work.

Table 5: Mixed-precision quantization results on three backbones. The superscript † denotes the W4A4† setting, where down_proj activations are kept at 8-bit.

#BW	Qwen2-VL-2B		Qwen2.5-VL-3B		Qwen3-VL-2B	
	Avg.	Δ	Avg.	Δ	Avg.	Δ
W4A4	54.45	–	55.08	–	59.83	–
W4A4†	55.89	+1.44	64.47	+9.39	63.04	+3.21

4.3 Ablation Study

4.3.1 Step-by-step ablation analysis.



(a) W4A8. **(b)** W4A4.

Figure 7: The step-by-step ablation studies on SRL and PBA.

We conduct step-by-step ablation studies on Qwen3-VL-2B-Instruct under W4A8 and W4A4 settings in Figure 7. Starting from the naive QAT baseline, the proposed SRL scheme improves the average accuracy by +1.00 and +3.36 under W4A8 and W4A4, respectively, indicating that refined supervision alone already provides meaningful gains. Applying PBA to the same baseline yields gains of +0.67 and +3.02, confirming that the staged quantization strategy also contributes independently by better preserving visual-sensitive layers. Combining both components further increases the average accuracy by +1.44 and +6.07, respectively, achieving the best performance in both settings and demonstrating that SRL and PBA are complementary.

Table 7: Ablation studies on SRL labeler, PBA scheduler, and block granularity.

(a) PBA scheduler.

Scheduler	Avg. (↑)
Cosine	71.34
Linear	71.00
Log. (Ours)	**72.54**

(b) Block granularity

Blk. Granu.	Avg. (↑)
$0.5 \times \theta_{\text{DecLayer}}$	72.15
$2 \times \theta_{\text{DecLayer}}$	70.28
$1 \times \theta_{\text{DecLayer}}$	**72.54**

(c) SRL labeler.

Labeler	W4A8	W4A4
GPT-4	71.77	56.78
Self (Ours)	**72.54**	**59.83**
Homo. 8B‡	72.35	59.04

4.3.2 Ablation on block activation metric.

Table 6 compares different metrics and priors for block quantization activation priority. A random selection yields an average accuracy of 71.88, serving as a baseline. We evaluate the performance by taking the weight gradients norm $\|\mathbf{G}_w\|_{\ell_1}$ as an activation metric. As plotted in Figure 5, early layers have larger $\|\mathbf{G}_w\|_{\ell_1}$ than that of deep layers. Therefore, $\|\mathbf{G}_w\|_{\ell_1}$ (Higher) ($\|\mathbf{G}_w\|_{\ell_1}$ (Lower)) in Table 6 activates layers from early to deep layers (from deep to shallow layers) approximately. With our vision-sensitivity factor τ, the lower-prior setting (TSB-prioritized) is slightly weaker, whereas prioritizing higher-τ blocks (VSB-prioritized) achieves the best result. Overall, these results validate that τ is a more effective criterion for PBA, and that allocating higher priority to the most VSBs is crucial for maximizing performance.

Table 6: Ablation studies on block activation metric

Metric	Prior	Avg. (↑)
Random	-	71.88
$\|\mathbf{G}_w\|_{\ell_1}$	Lower	72.10
$\|\mathbf{G}_w\|_{\ell_1}$	Higher	72.19
$\vec{\tau}$	Lower	72.03
$\vec{\tau}$ (Ours)	Higher	**72.54**

4.3.3 Ablation on PBA scheduler.

As shown in Table 7a, we test the PBA scheduler across 3 different annealing functions, which are *Cosine*, *Linear*, and *Logarithm*, respectively. Taking `Qwen3-VL-2B-Instruct` with 28 decoding layers as an example, we employ the *Logarithm* functionalized scheduler, which activates 50%, 75%, 88%, and 100% layers as training progresses. Training iterations within each stage are empirically set to $\sim$10% of total iterations, with the remaining budgets assigned to warmup and post-hoc recovery. Our ablation study demonstrates that *Logarithm* scheduler, which activates more blocks at early stages, yields better performance than *Cosine* and *Linear* functions.

4.3.4 Ablation on block granularity.

We explore how the block splitting strategy affects performance. Table 7b shows that treating each decoding layer as a single activation unit ($1 \times \theta_{\text{DecLayer}}$) yields the best accuracy. $0.5 \times \theta_{\text{DecLayer}}$ treats the `attn` and `ffn` modules as separate units, which can trigger asynchronous quantization activation within the same layer and thus hurts performance. $2 \times \theta_{\text{DecLayer}}$ leads to severe degradation, suggesting that overly coarse granularity has detrimental side effects. One reason is that coarse units may mix blocks with substantially different vision sensitivity, forcing them to share the same activation schedule. As a result, quantization may be enabled too early for less robust components or too late for truly vision-critical ones, weakening the intended MM-understanding-first recovery. This is consistent with Figure 5, where consecutive layers can exhibit large variations in visual sensitivity.

4.3.5 Ablation on SRL labeler.

We conduct experiments on `Qwen3-VL-2B-Instruct` with dataset labeled by a closed-source stronger model `GPT-4` [Liu et al., 2024b], self (`Qwen3-VL-2B-Instruct`) and homogeneous larger model (Homo. 8B‡, `Qwen3-VL-8B-Instruct`). Results in Table 7c show that both self and homogeneous larger models outperform the distribution-mismatched `GPT-4` labeler by a large margin. Furthermore, self-annotated datasets are better aligned with the target distribution and yield the best results. The extended labeler comparison and distribution diagnostics in Appendix D.4 show that self-labels outperform same-family and cross-family alternatives. Per-model relabeling remains preferable when its generation cost is acceptable.

4.4 Hardware Efficiency on Edge Accelerator

To assess edge deployment, we evaluate a silicon-proven TSMC 22 nm ASIC using measured on-chip latency and average power, off-chip DRAM simulation, and an end-to-end cycle-accurate accelerator model. The boundary covers the accelerator and DRAM accesses. Across three backbones, `ProQuant` achieves up to $3.64\times$ prefill speedup and $5.64\times$ energy reduction. Appendix E.2 documents the setup and measurement methodology; Appendices E.3 and E.4 report results across precisions and token configurations.

5 Conclusion

In this paper, we present `ProQuant`, a weight-activation QAT framework tailored for low-bit quantization of small-scale MLLMs. To mitigate data mismatch under limited fine-tuning budgets, we introduced an SRL scheme that re-annotates open-source MM datasets using the target model, yielding supervision that more closely aligns with the model's distribution. To address the coupled degradation of MM understanding and language modeling, we proposed a PBA mechanism that prioritizes recovering MM understanding first, guided by a vision-sensitivity factor $\vec{\tau}$ for identifying MM-critical layers. Extensive evaluations on diverse MM benchmarks and models demonstrate that `ProQuant` achieves near-lossless accuracy under W4A8 and remains robust where PTQ methods collapse at W4A4. We expect `ProQuant` to facilitate hardware-friendly deployment of MLLMs on resource-constrained edge platforms.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their constructive feedback. This work received no external funding or third-party support. The authors declare no competing interests.

References

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.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

LLM-Core-Team Xiaomi. Mimo-vl technical report, 2025. URL `https://arxiv.org/abs/2506.03569`.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models, 2025. URL `https://arxiv.org/abs/2504.10479`.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun,

Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, Xin Xie, Yuxiang You, Kai Dong, Xingkai Yu, Haowei Zhang, Liang Zhao, Yisong Wang, and Chong Ruan. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding, 2024. URL https://arxiv.org/abs/2412.10302.

Ovis-Embedding Team. Ovis-embedding: Pushing the frontiers of universal omni-modal embeddings. *arXiv preprint arXiv:2609.25165*, 2026.

Yanping Li, Wei Zhou, Yawen Liu, Yibo Wang, Ke Zhu, Guangda Huzhang, Qing-Guo Chen, Zhao Xu, Jun Zhang, and Wei Wei. Pace: Progressive angular-to-norm contrastive embedding. *arXiv preprint arXiv:2609.15152*, 2026a.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18407–18418, 2024.

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, 2026. URL https://arxiv.org/abs/2511.15690.

Chengtao Lv, Bilang Zhang, Yang Yong, Ruihao Gong, Yushi Huang, Shiqiao Gu, Jiajun Wu, Yumeng Shi, Jinyang Guo, and Wenya Wang. Llmc+: Benchmarking vision-language model compression with a plug-and-play toolkit, 2025. URL https://arxiv.org/abs/2508.09981.

Hongjun An, Wenhan Hu, Sida Huang, Siqi Huang, Ruanjun Li, Yuanzhi Liang, Jiawei Shao, Yiliang Song, Zihan Wang, Cheng Yuan, et al. Ai flow: Perspectives, scenarios, and approaches. *Vicinagearth*, 3(1):1, 2026.

Jiawei Shao and Xuelong Li. Ai flow at the network edge. *IEEE Network*, 2025.

Cheng Yuan, Zhening Liu, Jiashu Lv, Jiawei Shao, Yufei Jiang, Jun Zhang, and Xuelong Li. Task-oriented feature compression for multimodal understanding via device-edge co-inference. *IEEE Transactions on Mobile Computing*, 2025.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, pages 291–326. Chapman and Hall/CRC, 2022.

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 *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024a.

Yushi Huang, Ruihao Gong, Jing Liu, Tianlong Chen, and Xianglong Liu. Tfmq-dm: Temporal feature maintenance quantization for diffusion models, 2024a. URL https://arxiv.org/abs/2311.16503.

Yushi Huang, Ruihao Gong, Xianglong Liu, Jing Liu, Yuhang Li, Jiwen Lu, and Dacheng Tao. Temporal feature matters: A framework for diffusion model quantization, 2025a. URL https://arxiv.org/abs/2407.19547.

Ruihao Gong, Yang Yong, Shiqiao Gu, Yushi Huang, Chentao Lv, Yunchen Zhang, Xianglong Liu, and Dacheng Tao. Llmc: Benchmarking large language model quantization with a versatile compression toolkit, 2024. URL https://arxiv.org/abs/2405.06001.

Ruihao Gong, Xianglong Liu, Yuhang Li, Yunqiang Fan, Xiuying Wei, and Jinyang Guo. Pushing the limit of post-training quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Mengzhao Chen, Chaoyi Zhang, Jing Liu, Yutao Zeng, Zeyue Xue, Zhiheng Liu, Yunshui Li, Jin Ma, Jie Huang, Xun Zhou, et al. Scaling law for quantization-aware training. *arXiv preprint arXiv:2505.14302*, 2025a.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. Bitdistiller: Unleashing the potential of sub-4-bit llms via self-distillation. *arXiv preprint arXiv:2402.10631*, 2024.

Yushi Huang, Ruihao Gong, Jing Liu, Yifu Ding, Chengtao Lv, Haotong Qin, and Jun Zhang. Qvgen: Pushing the limit of quantized video generative models, 2025b. URL https://arxiv.org/abs/2505.11497.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10081–10100, 2025b.

Tianyu Guo, Shanwei Zhao, Shiai Zhu, and Chenguang Ma. Speed-q: Staged processing with enhanced distillation towards efficient low-bit on-device vlm quantization. *arXiv preprint arXiv:2511.08914*, 2025.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

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. *arXiv preprint arXiv:2308.13137*, 2023.

Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Zhihang Yuan, Jiangyong Yu, Chen Xu, Zhe Jiang, and Sifan Zhou. OstQuant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. *arXiv preprint arXiv:2501.13987*, 2025.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Yuhang Li, Ruokai Yin, Donghyun Lee, Shiting Xiao, and Priyadarshini Panda. Gptaq: Efficient finetuning-free quantization for asymmetric calibration. *arXiv preprint arXiv:2504.02692*, 2025a.

Shiyao Li, Yingchun Hu, Xuefei Ning, Xihui Liu, Ke Hong, Xiaotao Jia, Xiuhong Li, Yaqi Yan, Pei Ran, Guohao Dai, et al. Mbq: Modality-balanced quantization for large vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4167–4177, 2025b.

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:2508.03351*, 2025.

JiangYong Yu, Sifan Zhou, Dawei Yang, Shuo Wang, Shuoyu Li, Xing Hu, Chen Xu, Zukang Xu, Changyong Shu, and Zhihang Yuan. Mquant: Unleashing the inference potential of multimodal large language models via full static quantization. *arXiv preprint arXiv:2502.00425*, 2025.

Changyuan Wang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Q-vlm: Post-training quantization for large vision-language models. *arXiv preprint arXiv:2410.08119*, 2024a.

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*, pages 10582–10591, 2024.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800, 2019.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pages 235–251. Springer, 2016.

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.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Lmms-eval: Reality check on the evaluation of large multimodal models, 2024. URL `https://arxiv.org/abs/2407.12772`.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, et al. Mme: a comprehensive evaluation benchmark for multimodal large language models. corr abs/2306.13394 (2023), 2023.

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

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209, 2021.

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 CVPR*, 2024.

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 *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023a.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms. *arXiv preprint arXiv:2402.04291*, 2024b.

Mart Van Baalen, Andrey Kuzmin, Suparna S Nair, Yuwei Ren, Eric Mahurin, Chirag Patel, Sundar Subramanian, Sanghyuk Lee, Markus Nagel, Joseph Soriaga, et al. Fp8 versus int8 for efficient deep learning inference. *arXiv preprint arXiv:2303.17951*, 2023.

Jinhao Li, Jiaming Xu, Shan Huang, Yonghua Chen, Wen Li, Jun Liu, Yaoxiu Lian, Jiayi Pan, Li Ding, Hao Zhou, et al. Large language model inference acceleration: A comprehensive hardware perspective. *arXiv preprint arXiv:2410.04466*, 2024b.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024.

Sangjin Kim, Jungjun Oh, Byeongcheol Kim, Yuseon Choi, Gwangtae Park, and Hoi-Jun Yoo. 31.2 revolver: Low-bit genai accelerator for distilled-model and cot with phase-aware-quantization and rotation-based integer-scaled group quantization. In *2026 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 69, pages 534–536, 2026. doi: 10.1109/ISSCC49663.2026.1140901.

Bo Liu, Zihan Zou, Xinming Yan, Xilong Kang, Xinyang Chen, Bo Hu, Jiaming Lin, Haoran Du, Jun Yang, Xin Si, and Hao Cai. A 51.6uj/token subspace-rotation-based dual-quantized large-language-model accelerator with fused scale-activation int datapath and rearranged bit-slice lut computation. In *2026 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 69, pages 536–538, 2026. doi: 10.1109/ISSCC49663.2026.11409212.

Pingcheng Dong, Yonghao Tan, Xuejiao Liu, Peng Luo, Yu Liu, Di Pang, Songchen Ma, Xijie Huang, Shih-Yang Liu, Dong Zhang, Zhichao Lu, Luhong Liang, Chi-Ying Tsui, Fengbin Tu, Liang Zhao, and Kwang-Ting Cheng. 31.1 a 14.08-to-135.69token/s reram-on-logic stacked outlier-free large-language-model accelerator with block-clustered weight-compression and adaptive parallel-speculative-decoding. In *2026 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 69, pages 532–534, 2026. doi: 10.1109/ISSCC49663.2026.11409211.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Sijia Li, Yuchen Huang, Zifan Liu, Yanping Li, Jingjing Fu, Li Zhao, Jiang Bian, Ling Zhang, Jun Zhang, and Rui Wang. Gear: Granularity-adaptive advantage reweighting for llm agents via self-distillation. *arXiv preprint arXiv:2605.11853*, 2026b.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023b.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385. IEEE, 2024.

An Vo, Khai-Nguyen Nguyen, Mohammad Reza Taesiri, Vy Tuong Dang, Anh Totti Nguyen, and Daeyoung Kim. Vision language models are biased. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2505.23941.

Yanping Li, Zhening Liu, Zijian Li, Zehong Lin, and Jun Zhang. Token-level data selection for safe llm fine-tuning. *arXiv preprint arXiv:2603.01185*, 2026c.

Appendix Overview

We provide additional details in the appendix, organized as follows

- ▷ Section A introduces MLLM quantization preliminaries.
- ▷ Section B provides additional details of `ProQuant`, including the overall training procedure and extended SRL examples.
- ▷ Section C reports implementation details, covering baseline configurations, SRL data generation settings, and the reported training configuration for QAT.
- ▷ Section D supplements the main results with additional ablations, cross-family evaluation, and SRL distribution and robustness analyses.
- ▷ Section E presents the hardware architecture, measurement setup, and BF16/W8A8/W4A8/W4A4 efficiency comparisons.
- ▷ Section F discusses limitations and broader impacts.

A Preliminary

A.1 Background

Existing QAT solutions primarily focus on low-bit weight-only quantization [Chen et al., 2025b; Du et al., 2024; Huang et al., 2024b; Guo et al., 2025]. However, weight-only inference typically relies on floating-point cores to process activations, incurring substantial power and area overhead on edge AI chips [Van Baalen et al., 2023; Li et al., 2024b]. This practical constraint motivates us to pursue high-performance weight-activation quantization, and our work targets weight-activation QAT for low-bit edge MLLMs.

SmoothQuant-style methods [Xiao et al., 2023] achieve strong accuracy at W8A8, but pushing to lower-bit settings often depends on rotation-based quantization [Ashkboos et al., 2024; Liu et al., 2024a; Hu et al., 2025; Sun et al., 2024]. Although the overhead can be modest on GPUs [Ashkboos et al., 2024; Sun et al., 2024], such online rotations typically require additional hardware support on edge chips, increasing chip area and compute cost [Kim et al., 2026; Liu et al., 2026; Dong et al., 2026]. In contrast, `ProQuant` introduces no extra runtime operators during inference beyond standard quantized linear layers.

A.2 Quantization

Quantization reduces the precision of weights (and/or activations) to save memory and accelerate computation. For a weight tensor $\mathbf{W}$, standard uniform b-bit quantization and de-quantization are formulated as

$$\mathbf{W}_{\text{int}} = \text{clip}\left(\left\lfloor \frac{\mathbf{W}}{s} \right\rceil + z,\ q_{\min},\ q_{\max}\right), \quad \hat{\mathbf{W}} = (\mathbf{W}_{\text{int}} - z) \times s, \tag{A1}$$

where $\mathbf{W}_{\text{int}}$ and $\hat{\mathbf{W}}$ denote the quantized integer weights and the de-quantized full-precision weights, respectively. Here, s and z are the scale and zero-point, and $\text{clip}(\cdot,\cdot,\cdot)$ bounds integer values to $[q_{\min}, q_{\max}]$. In this paper, we adopt static *per-channel* quantization for weights and dynamic *per-token* quantization for activations, following common practice in the community [Liu et al., 2023; Xiao et al., 2023]. We use signed symmetric quantization with $z = 0$ throughout; $q_{\min}$ and $q_{\max}$ denote the signed integer bounds of the quantizer.

A.3 Training-based Quantization

Training-based quantization introduces learnable parameters [Xie et al., 2024; Liu et al., 2023, 2024a] or functions [Nagel et al., 2020; Li et al., 2021] to recover quantized model accuracy in a block-wise [Li et al., 2021; Shao et al., 2023] or network-wise manner [Liu et al., 2023; Du et al., 2024]. In this paper, `ProQuant` performs network-level QAT by learning the quantization scales s together with the weight parameters $\mathbf{W}$. To make the rounding operator $\lfloor\cdot\rceil$ differentiable, we adopt the straight-through estimator (STE) [Bengio et al., 2013]. Following prior works [Liu et al., 2023; Du et al., 2024; Li et al., 2026b], we employ self-distillation, where the full-precision model serves as the teacher to guide the quantized student. We also incorporate supervision from the training response

labels, as in [Guo et al., 2025]. Specifically, our objective is a weighted sum of forward KL, reverse KL, and cross-entropy losses:

$$\mathcal{L}_{\text{QAT}} = \alpha\mathcal{L}_{\text{KL}}(P_s||P_t) + (1-\alpha)\mathcal{L}_{\text{KL}}(P_t||P_s) + \beta\mathcal{L}_{\text{CE}}(Y, P_s), \tag{A2}$$

where α is estimated at the start of training as the average token probability over a randomly sampled mini-batch [Guo et al., 2025; Du et al., 2024]. We empirically set β to match the scale of the other loss terms. Here, P_t and P_s are teacher and student token-probability distributions obtained by applying softmax to their logits. For `ProQuant`, $Y = Y_{\text{SRL}}$ denotes the self-relabeled response targets; labeler ablations replace these targets with the corresponding label set. More discussion is provided in Appendix D.2.

B `ProQuant`

B.1 Training Procedures

Algorithm 1 outlines the training procedure of `ProQuant`. Starting from the re-labeled dataset $\mathcal{D}_{\text{SRL}}$, the full-precision teacher $\mathcal{M}_{\text{FP}}$, and the quantized student $\mathcal{M}_{\text{Q}}$, the model is optimized over S steps with a predefined activation schedule $\mathbb{T}$ and block budget $\mathbb{B}$. At scheduled steps, `ProQuant` performs teacher-student forward propagation, computes the QAT loss, updates the quantized parameters, and progressively activates more quantized blocks based on the estimated sensitivity. At all other steps, the student model is updated with standard QAT.

Algorithm 1 `ProQuant trainer`

Preparation: re-labeled finetuning datasets $\mathcal{D}_{\text{SRL}} = (X,\ Y_{\text{SRL}})$, full precision teacher model $\mathcal{M}_{\text{FP}}$, quantized student model $\mathcal{M}_{\text{Q}}$ with L layers, and training steps S.
PBA toolkit: activation timing $\mathbb{T} = [S\times 0\%,\ S\times 30\%,\ S\times 40\%,\ S\times 50\%]$, and quantized block budget $\mathbb{B} = [L\times 50\%,\ L\times 75\%, L\times 88\%, L\times 100\%]$.

for $s = 0, 1, 2, \ldots, S$ **do**
  if $s \in \mathbb{T}$ **then**
    $\texttt{logits}_{\text{FP}} \leftarrow \texttt{forward}\,(\mathcal{M}_{\text{FP}}, \mathcal{D}^s_{\text{SRL}})$; ▷ compute teacher logits
    $\texttt{logits}_{\text{Q}} \leftarrow \texttt{forward}\,(\mathcal{M}_{\text{Q}}, \mathcal{D}^s_{\text{SRL}})$; ▷ compute student logits
    $\mathcal{L}_{\text{QAT}} \leftarrow \texttt{loss}(\texttt{logits}_{\text{FP}},\ \texttt{logits}_{\text{Q}},\ Y_{\text{SRL}})$; ▷ compute loss in Eq. A2
    $\vec{\tau},\ \mathbf{G} \leftarrow \texttt{backward}\,(\mathcal{M}_{\text{Q}}, \mathcal{L}_{\text{QAT}})$; ▷ cache $\vec{\tau}$ in Eq. 1
    $\widetilde{\mathbf{W}},\ \widetilde{\mathbf{S}} \leftarrow \texttt{optimize}(\mathbf{W},\ \mathbf{S};\ \mathbf{G})$; ▷ update weight and scale
    $\mathcal{M}_{\text{Q}} \leftarrow \texttt{block_quant_act}\left(\mathcal{M}_{\text{Q}};\ \texttt{argsort}_{\downarrow}\left(\vec{\tau}\right),\ \mathbb{B}\right)$; ▷ PBA mechanism
  else
    $\texttt{standard_qat_update}\,(\mathcal{M}_{\text{Q}};\ \mathcal{M}_{\text{FP}},\ \mathcal{D}_{\text{SRL}})$;
  end if
end for
Return $\mathcal{M}_{\text{Q}}$

B.2 Examples of SRL

We present one full example from the COCO caption dataset [Liu et al., 2024b] in Figure A1. It is evident that the two responses exhibit distinct stylistic characteristics, suggesting a potential distribution gap between them. Specifically, GPT-style responses tend to be more literal and spatially grounded, emphasizing object presence and layout with explicit positional phrases such as in the center of the table, closer to the top left corner, and along the top edge. In contrast, Qwen-style responses are more holistic, semantically enriched, and interpretive, focusing more on scene-level summarization, attributes, and visual appeal, as reflected in expressions such as colorful and creatively decorated donuts, different flavors, toppings, and textures, and a delightful and tempting selection of sweet treats. This discrepancy indicates that, even when describing the same image, different caption sources may encode substantially different linguistic preferences and descriptive priors. Such stylistic inconsistency can further introduce additional challenges for model adaptation and alignment, especially when the training objective implicitly assumes a unified target distribution.

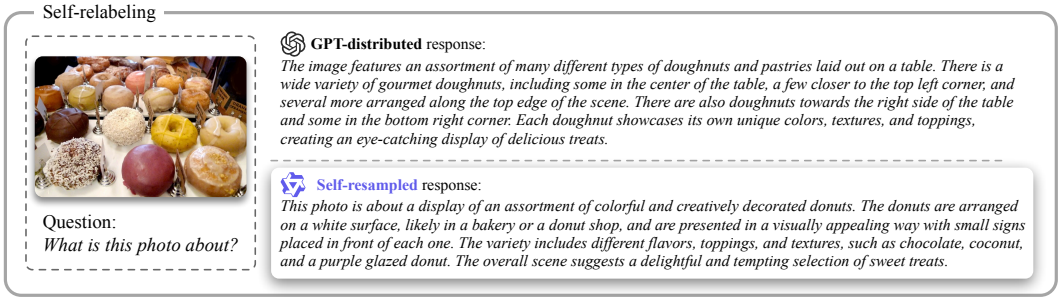


Figure A1: Extended example of resampled datasets in Section 3.

C Implementation Details

C.1 Baselines

SmoothQuant [Xiao et al., 2023]. SmoothQuant alleviates activation outliers via an equivalent per-channel scaling transformation. It uses dynamic per-token activation quantization together with static per-channel weight quantization.

MBQ [Li et al., 2025b]. MBQ extends SmoothQuant by introducing modality-balanced scaling factors derived from gradient information. Its quantized inference configuration is kept the same as that of SmoothQuant.

MQuant [Yu et al., 2025]. MQuant addresses the distributional discrepancy between visual and textual tokens, as well as the extreme outliers introduced by Hadamard-based transformations. It adopts a fully static quantization scheme for efficient deployment.

LLM-QAT [Liu et al., 2023]. LLM-QAT pioneers QAT framework for language transformers, with a primary focus on data-free distillation. Since its original data-free design is not directly applicable to MLLMs, we reproduce LLM-QAT using the same training data, budget, and optimization settings as `ProQuant`. For the W4A4 backbones marked with † in Table 1, we retain 8-bit `down_proj` activations for LLM-QAT to avoid severe degradation. The PTQ baselines are reproduced with their official implementations and the configurations reported in their respective papers.

C.2 Data Generation

We use the vLLM engine [Kwon et al., 2023] to generate the dataset on a single 8×H200 node. Following the practice of LLM-QAT [Liu et al., 2023], we generate SRL responses via stochastic sampling rather than greedy decoding, as sampling produces more diverse outputs and yields better performance across tasks. We follow the default configuration files for sampling hyperparameters. For `Qwen3-VL-2B-Instruct`, we set `top_p` $= 0.8$, `top_k` $= 20$, `temperature` $= 0.7$, and `repetition_penalty` $= 1.0$ according to the released `generation_config.json`. Additionally, we set `max_token_len` $= 512$ to avoid excessively long responses. Generating SRL labels for `LLaVA-680K` [Liu et al., 2024b] takes ∼2.5 GPU days for 2B-scale models.

C.3 Training Configuration

We implement `ProQuant` in `PyTorch` and train quantized models with `DeepSpeed`. Each experiment is conducted on a single 8×H200 node. Unless otherwise specified, we fine-tune quantized models for 5 epochs in BF16, enabling gradient checkpoints and using gradient accumulation with 2 steps. We set the per-device batch size to 32 and the maximum sequence length to 8192 tokens. We use the `AdamW` optimizer with a constant learning rate, zero weight decay, and a maximum gradient norm of 1.0. During QAT, we freeze the vision transformer and train the remaining modules together with the quantization scales, using module-specific learning rates. We optimize a hybrid objective that combines forward KL, reverse KL, and supervised fine-tuning losses, as defined in Eq. A2. The QAT takes ∼2 GPU days for 2B-scale models.

D More Experiments

D.1 Ablation on SmoothQuant initialization.

`ProQuant` can be applied on top of SmoothQuant [Xiao et al., 2023] initialization. As shown in Table A1, combining SmoothQuant with W4A4 quantization further improves accuracy. In contrast, SmoothQuant can slightly degrade performance under W4A8 settings. These results suggest that SmoothQuant serves as an effective initialization for extreme low-bit (W4A4) QAT, consistent with observations in LLM-QAT [Liu et al., 2023].

Table A1: `ProQuant` *w/* or *w/o* SmoothQuant.

#BW	SQ Init	Avg. (↑)	MME
W4A8	✓	71.41	1956.58
	✗	**72.54**	**1962.04**
W4A4	✗	50.87	1459.14
	✓	**59.83**	**1652.63**

D.2 Ablation Study on Loss Functions

We present performance under different supervision objectives for W4A8 `Qwen3-VL-2B-Instruct` QAT in Table A2. Using a single KL term yields limited gains, where forward KL achieves 71.76% on average accuracy while reversed KL performs worse, suggesting that either direction alone provides insufficient guidance for stable recovery. Combining both terms via CAKLD [Du et al., 2024] improves the average accuracy to 71.94%, indicating that balancing the two divergences is beneficial. The balance factor α is estimated at the start of training as described above. Building on CAKLD, our multi-objective formulation further incorporates a cross-entropy term following [Guo et al., 2025], achieving the best performance. Overall, the results validate that jointly leveraging bidirectional distillation and label supervision provides the most effective training signal for low-bit QAT.

Table A2: QAT performance under different supervisions.

Supervision	Formulation	Avg. (↑)
KL Div.	$\mathcal{L}_{\text{KL}} = \mathcal{D}(P_t \| P_s)$	71.76
Reversed KL	$\mathcal{L}_{\text{ReKL}} = \mathcal{D}(P_s \| P_t)$	71.41
CAKLD Du et al. [2024]	$\mathcal{L}_{\text{CAKLD}} = \alpha\mathcal{L}_{\text{ReKL}} + (1-\alpha)\mathcal{L}_{\text{KL}}$	71.94
Multi-obj. Guo et al. [2025]	$\mathcal{L}_{\text{QAT}} = \mathcal{L}_{\text{CAKLD}} + \beta\mathcal{L}_{\text{CE}}$	**72.54**

D.3 Generalization Beyond the Qwen-VL Family

We evaluate InternVL3.5-1B and LLaVA-OV-0.5B at W4A8 using the same seven-benchmark average as in the main experiments (Table A3). `ProQuant` achieves 63.93 and 52.29, respectively, outperforming MBQ and LLM-QAT on both models. These experiments extend the evaluated architectures beyond Qwen-VL.

Table A3: Average accuracy over the seven benchmarks used in the main paper on two additional model families. Quantized results use W4A8.

Method	InternVL3.5-1B	LLaVA-OV-0.5B
BF16	69.92	57.95
MBQ	56.77	48.97
LLM-QAT	60.02	50.05
ProQuant (ours)	63.93	52.29

Table A4 isolates the contributions of SRL and PBA on InternVL3.5-1B. Each component improves naive QAT, and their combination performs best.

Table A4: Component ablation on InternVL3.5-1B at W4A8.

Setting	InternVL3.5-1B Avg.
Naive QAT	61.74
+ SRL	62.98 (+1.24)
+ PBA	63.01 (+1.27)
ProQuant	63.93 (+2.19)

D.4 Understanding SRL: Distribution, Style, and Label Transfer

We compare five label sets on the same randomly sampled 10K cases, pairing responses by prompt. The target is Qwen3-VL-2B and its self-labels serve as the reference. GPT and LV-0.5B/LV-7B are cross-family labelers; Qw-Homo-8B is a larger same-family labeler. The diagnostics in Table A5 are defined as follows.

1. **NLL (nats/token).** We compute $-|y|^{-1} \sum_t \log p_{\text{target}}(y_t \mid y_{<t}, x)$. Lower values indicate higher likelihood under the target model.
2. **Vocabulary coverage.** The fraction of the reference's top-5K most frequent vocabulary items covered by assistant tokens in the label set.
3. **Normalized length distance.** The Wasserstein-1 distance between response token-count distributions, divided by the reference mean length.
4. **Format distance.** The ℓ_1 distance between mean 13-dimensional binary format-feature vectors, including headings, bold text, bullets, lists, code blocks, LaTeX, and tables.
5. **Semantic distance.** The mean paired $1 - \cos$ similarity between responses encoded with the frozen `gte-large-en-v1.5` sentence encoder [Li et al., 2023b].

Table A5: Label-distribution diagnostics and QAT accuracy on Qwen3-VL-2B. Qw-Self-2B is the reference; LV labelers are cross-family and Qw-Homo-8B is a larger same-family labeler.

Metric	GPT	LV-0.5B	LV-7B	Qw-Homo-8B	Qw-Self-2B
NLL nats/token (↓)	1.459	1.504	1.450	0.906	0.474
Vocab cov. top-5k (↑)	0.973	0.987	0.951	1.000	1
Length W1, normalized (↓)	0.641	0.780	0.929	0.459	0
Format L1 (↓)	2.831	2.861	3.268	1.213	0
1 - paired cosine (↓)	0.178	0.172	0.265	0.067	$\sim$0
W4A8 Avg. Acc. (↑)	71.77	70.71	71.29	72.35	72.54
W4A4 Avg. Acc. (↑)	56.78	56.31	57.34	59.04	59.83

Response style is a component of the output distribution, rather than an alternative to distributional alignment. Vocabulary, length, and format characterize surface form, while embedding similarity characterizes semantics; these diagnostics are complementary views and are not an additive decomposition of NLL. Matching surface form alone does not explain the full gain: the same-family 8B labeler reaches reference vocabulary coverage (1.000), but its NLL remains 0.906 versus 0.474 for self-labels and its QAT accuracy remains lower.

At W4A4, the NLL ranking agrees exactly with the inverse accuracy ranking. At W4A8, GPT versus LV-7B is an exception (their NLL differs by 0.009). Moreover, LV-0.5B is closer to the reference than LV-7B on all four auxiliary diagnostics, but has higher NLL and lower QAT accuracy at both precisions. These results support target-distribution proximity as an explanation of SRL's gain beyond a single stylistic property.

Label transferability. Self-labels give the best W4A4 accuracy (59.83), followed by same-family 8B labels (59.04), and then cross-family labels (57.34, 56.78, and 56.31). Same-family reuse is therefore possible with an accuracy penalty, while cross-family reuse does not preserve the full benefit of SRL. We recommend per-target-model relabeling where its generation cost is acceptable.

D.5 Hallucination and Bias Evaluation

We evaluate Qwen2-VL-2B on HallusionBench [Guan et al., 2024], using DeepSeek-V4-Flash as the judge, and on VLMsAreBiased [Vo et al., 2026]. Table A6 compares `ProQuant` with and without SRL at W4A8 and W4A4. SRL improves the reported scores at both precisions, while the quantized models remain below the BF16 target. Thus, these evaluations do not show a deterioration from SRL on the two tested benchmarks. They do not rule out inherited hallucinations, biases, or failure modes [Li et al., 2026c] outside these evaluations.

Table A6: Hallucination and bias evaluation on Qwen2-VL-2B. HallusionBench uses DeepSeek-V4-Flash as the judge. The table reports the scores for each setting.

Setting	HallusionBench	VLMsAreBiased
Full (BF16, target model)	51.21	17.71
ProQuant W4A8	50.37	17.24
ProQuant w/o SRL W4A8	50.16	15.56
ProQuant W4A4	43.84	14.58
ProQuant w/o SRL W4A4	41.42	13.00

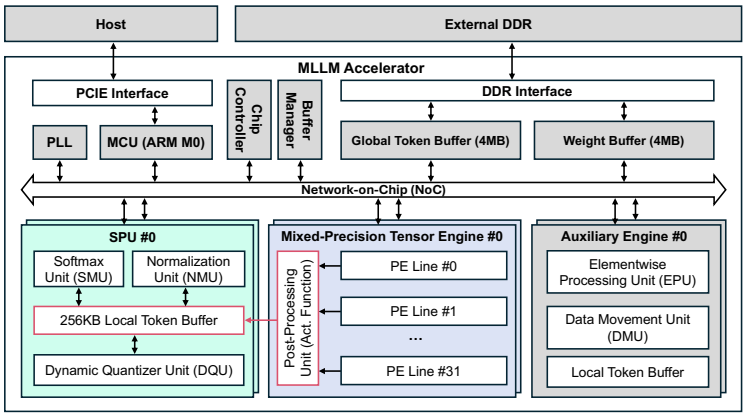


Figure A2: System-level architecture of our benchmark edge MLLM accelerator.

E Hardware Efficiency on Edge Accelerator

E.1 Hardware Architecture

We evaluate an edge MLLM accelerator using measured test-chip behavior and a cycle-accurate accelerator model, as detailed in Section E.2. Figure A2 shows its architecture. The accelerator is equipped with two Mixed-Precision Tensor Engines (MPTEs), each of which comprises 1K MACs organized into 32 PE lines under an output-stationary dataflow, delivering 2.46 TOPS at INT8 precision. The MPTEs are able to support BF16, INT8 and INT4 precision for both weights and activations. Besides, two Special Processing Units (SPUs) support BF16 softmax, normalization, and dynamic quantization. Furthermore, output feature map tiles from the MPTEs can be bypassed directly to the SPU's local token buffer to facilitate layer fusion, effectively overlapping the dynamic quantization latency. Two Auxiliary Engines handle element-wise operations, such as RoPE, and various data movement tasks like reshaping and transposing. The accelerator also features a 4 MB global token buffer and a 4 MB weight buffer, connected via a DDR4 interface that delivers 38.4 GB/s of external memory bandwidth. The compute engines and on-chip buffers are interconnected via a network-on-chip (NoC).

E.2 System Setup and Measurement Methodology

Hardware and memory. The test chip is a silicon-proven ASIC fabricated in TSMC 22 nm CMOS. The FPGA is the interface/control board, rather than the inference accelerator. The system includes 8 GB off-chip DDR4, a 4 MB global token buffer for intermediate features, and a 4 MB weight buffer for the current weight tile. The off-chip capacity accommodates the evaluated model weights when quantized to 8 bits or below. These are provisioned capacities, not measured peak memory usage.

Connections and workflow. The DC supply powers the system board; the PC monitors it and the oscilloscope observes its state signals. The voltage tuning circuit supplies the test chip. The FPGA sends data to the test-chip board and receives results. A customized MLIR-based compiler produces test data and instructions. The FPGA forwards them to the chip, and latency is monitored via the PC waveform.

Measured and simulated components. On-chip energy is obtained from measured latency and average power,

$$E_{\text{chip}} = t_{\text{chip}}\,\overline{P}_{\text{chip}}. \tag{A3}$$

Off-chip DRAM access latency and energy are simulated using DRAMPower. The end-to-end cycle-accurate accelerator model combines the on-chip and off-chip contributions. The reported boundary includes the accelerator and DRAM accesses, not the wall-plug consumption of the PC, FPGA control board, oscilloscope, and other test equipment.

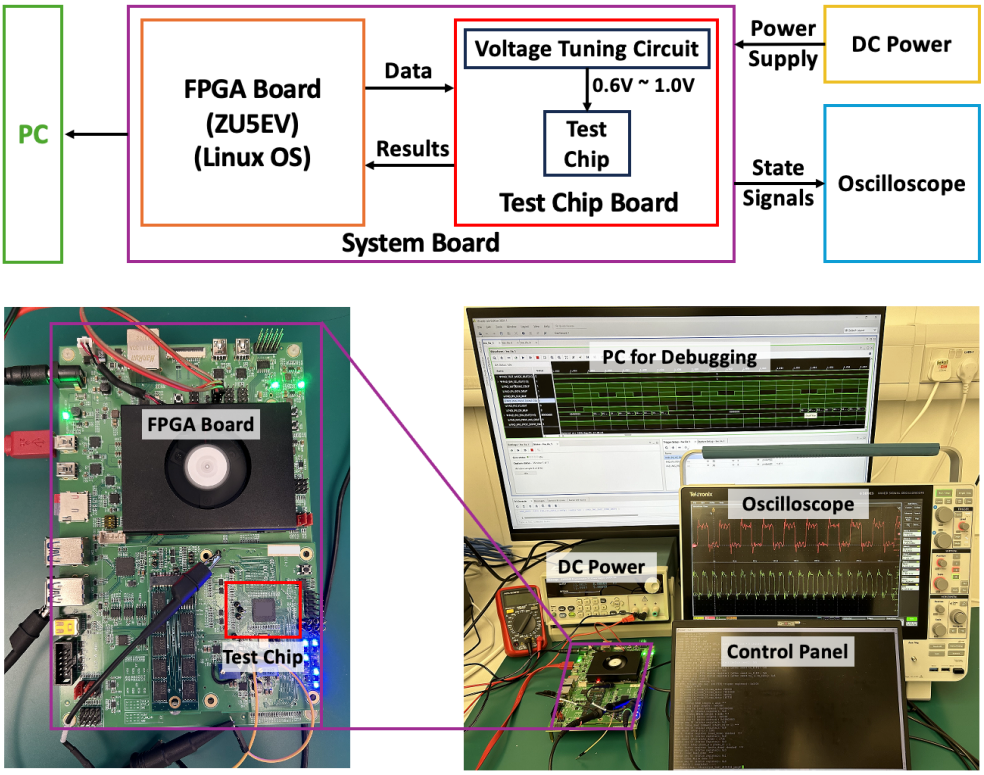


Figure A3: Test environment. A Linux-based ZU5EV FPGA board supplies data and instructions to the test-chip board and receives results. The PC supports debugging and control, the DC supply powers the system board, and the oscilloscope observes state signals. The voltage tuning circuit supports 0.6–1.0 V; this range is not a statement of the operating voltage used in every experiment.

Reported metrics. Throughput is calculated as $1000/t$, where t is the reported latency in milliseconds. For the same token configuration, speedup is $t_{\mathrm{BF16}}/t_{\mathrm{quant}}$ and the energy-reduction ratio is $E_{\mathrm{BF16}}/E_{\mathrm{quant}}$. The tables distinguish prefill and decode and preserve the original latency/energy units.

E.3 Edge Evaluation Results

Tables A7, A8, and A9 present comprehensive latency and energy measurements for `Qwen2-VL-2B-Instruct`, `Qwen2.5-VL-3B-Instruct` and `Qwen3-VL-2B-Instruct` on the edge MLLM accelerator. Across all three models, the prefill stage shows W4A8 achieving 1.48∼1.89× speedup and 1.94∼3.13× energy savings, while W4A4 further attains 2.11∼3.64× speedup and 2.53∼5.64× energy reduction. The decode stage exhibits consistent 3.74∼3.98× speedup and 3.76∼3.99× energy savings for both W4A8 and W4A4 due to memory-bound issue. Notably, the speedup ratio diminishes with longer visual token length since the ViT FLOPs proportion increases. As the ViT backbone typically remains in BF16 precision to preserve visual features, it becomes a bottleneck on edge accelerators where BF16 compute resources are insufficient due to area constraints. These results validate that `ProQuant` generalizes effectively across model scales and delivers substantial efficiency gains for edge deployment.

E.4 W8A8 Deployment and Precision Trade-offs

Tables A10–A12 add W8A8 deployment results for all three backbones. Table A13 summarizes the precision-dependent gains alongside the existing W4A8/W4A4 measurements. W8A8 gives about 1.00–1.05× prefill speedup and 1.42–1.79× energy reduction. Decode speedup is 1.95–2.00×, with 1.99–2.00× energy reduction.

Table A7: Performance comparison of BF16, W4A8, and W4A4 for `Qwen2-VL-2B-Instruct` in both prefill and decode stages. "#V Tok." and "#T Tok." denote the numbers of vision tokens and text tokens, respectively; "Lat." and "Eng." denote latency (ms) and energy (mJ), respectively; "Spd." and "Sav." denote the speedup and energy saving relative to the BF16 baseline, respectively.

Stage	#V Tok.	#T Tok.	BF16		W4A8				W4A4			
			Lat.	Eng.	Lat.	Spd.	Eng.	Sav.	Lat.	Spd.	Eng.	Sav.
Prefill	256	256	877.02	2400.87	530.80	1.65×	1020.90	2.35×	350.51	2.50×	779.96	3.08×
		1024	1977.79	4817.34	1144.89	1.73×	1888.29	2.55×	632.23	3.13×	1164.54	4.14×
	1024	256	2559.89	6045.12	1726.99	1.48×	3116.07	1.94×	1214.34	2.11×	2392.32	2.53×
		1024	3766.04	8654.54	2441.82	1.54×	4069.73	2.13×	1522.48	2.47×	2831.59	3.06×
Decode	256	256	124.55	493.04	31.79	3.92×	124.33	3.97×	31.40	3.97×	123.95	3.98×
		1024	125.59	496.69	32.83	3.83×	126.66	3.92×	31.85	3.94×	125.71	3.95×
	1024	256	125.59	496.69	32.83	3.83×	126.66	3.92×	31.85	3.94×	125.71	3.95×
		1024	126.63	500.34	33.87	3.74×	128.99	3.88×	32.30	3.92×	127.46	3.93×

Table A8: Performance comparison of BF16, W4A8, and W4A4 for `Qwen2.5-VL-3B-Instruct` in both prefill and decode stages.

Stage	#V Tok.	#T Tok.	BF16		W4A8				W4A4			
			Lat.	Eng.	Lat.	Spd.	Eng.	Sav.	Lat.	Spd.	Eng.	Sav.
Prefill	256	256	1663.57	4802.11	882.53	1.89×	1580.31	3.04×	507.40	3.28×	1030.43	4.66×
		1024	3959.49	10486.84	2132.48	1.86×	3347.68	3.13×	1088.50	3.64×	1859.23	5.64×
	1024	256	4439.11	11419.90	2612.11	1.70×	4280.74	2.67×	1568.12	2.83×	2792.29	4.09×
		1024	6972.05	17529.33	4034.06	1.73×	6226.26	2.82×	2193.81	3.18×	3719.84	4.71×
Decode	256	256	248.69	984.06	63.25	3.93×	247.63	3.97×	62.45	3.98×	246.85	3.99×
		1024	250.47	989.19	65.03	3.85×	251.06	3.94×	63.03	3.97×	249.11	3.97×
	1024	256	250.47	989.19	65.03	3.85×	251.06	3.94×	63.03	3.97×	249.11	3.97×
		1024	252.25	994.31	66.81	3.78×	254.48	3.91×	63.61	3.97×	251.37	3.96×

Moving from W8A8 to W4A8 halves weight traffic and primarily improves memory-bound decode. Moving from W4A8 to W4A4 additionally lowers activation precision and benefits prefill. Relative to W8A8, the supported W4A8 and W4A4 configurations offer approximately $2\times$ and $4\times$ peak compute throughput, respectively; these hardware peak ratios are not end-to-end measured speedups. Off-chip data movement can be energy-intensive, motivating the reduction in weight traffic.

F Limitations and Broader Impacts

Limitations. Our experiments focus on compact MLLMs, with evaluated models up to 8B parameters. Extending `ProQuant` to larger models and a broader range of multimodal architectures remains an important direction for future work.

Broader impacts. By reducing inference memory requirements, latency, and energy consumption in the evaluated deployment setting, `ProQuant` can facilitate multimodal intelligence on resource-constrained devices. More efficient local inference may broaden access to multimodal applications and reduce reliance on remote computing infrastructure.

Table A9: Performance comparison of BF16, W4A8, and W4A4 for `Qwen3-VL-2B-Instruct` in both prefill and decode stages.

Stage	#V Tok.	#T Tok.	BF16		W4A8				W4A4			
			Lat.	Eng.	Lat.	Spd.	Eng.	Sav.	Lat.	Spd.	Eng.	Sav.
Prefill	256	256	869.62	2227.72	498.60	1.74×	860.56	2.59×	300.42	2.89×	569.67	3.91×
		1024	2090.36	4832.36	1190.86	1.76×	1791.66	2.70×	612.83	3.41×	1038.96	4.65×
	1024	256	2444.41	5570.82	1544.91	1.58×	2530.13	2.20×	966.88	2.53×	1777.43	3.13×
		1024	3798.92	8333.40	2370.94	1.60×	3559.28	2.34×	1313.97	2.89×	2231.03	3.74×
Decode	256	256	140.46	556.20	35.89	3.91×	141.56	3.93×	35.89	3.91×	141.56	3.93×
		1024	144.00	570.20	37.66	3.82×	148.57	3.84×	37.66	3.82×	148.57	3.84×
	1024	256	144.00	570.20	37.66	3.82×	148.57	3.84×	37.66	3.82×	148.57	3.84×
		1024	147.53	584.20	39.44	3.74×	155.57	3.76×	39.44	3.74×	155.57	3.76×

Table A10: BF16 / W8A8 deployment results for Qwen3-VL-2B. Throughput is in s^{-1}, latency in ms, and energy in mJ. Ratios are relative to BF16 under the same token configuration.

Stage	V tokens	T tokens	Throughput	Latency	Energy	Speedup	Energy ratio
Prefill	256	256	1.15 / 1.17	869.62 / 857.16	2227.72 / 1375.00	1.01x	1.62x
Prefill	256	1024	0.48 / 0.48	2090.36 / 2077.89	4832.36 / 2994.89	1.01x	1.61x
Prefill	1024	256	0.41 / 0.41	2444.41 / 2431.95	5570.82 / 3733.36	1.01x	1.49x
Prefill	1024	1024	0.26 / 0.26	3798.92 / 3786.46	8333.40 / 5437.21	1.00x	1.53x
Decode	256	256	7.12 / 14.21	140.46 / 70.35	556.20 / 278.22	2.00x	2.00x
Decode	256	1024	6.94 / 13.86	144.00 / 72.13	570.20 / 285.23	2.00x	2.00x
Decode	1024	256	6.94 / 13.86	144.00 / 72.13	570.20 / 285.23	2.00x	2.00x
Decode	1024	1024	6.78 / 13.53	147.53 / 73.90	584.20 / 292.23	2.00x	2.00x

Table A11: BF16 / W8A8 deployment results for Qwen2-VL-2B. Throughput is in s^{-1}, latency in ms, and energy in mJ. Ratios are relative to BF16 under the same token configuration.

Stage	V tokens	T tokens	Throughput	Latency	Energy	Speedup	Energy ratio
Prefill	256	256	1.14 / 1.16	877.02 / 863.03	2400.87 / 1519.00	1.02×	1.58×
Prefill	256	1024	0.51 / 0.51	1977.79 / 1968.44	4817.34 / 3018.88	1.00×	1.60×
Prefill	1024	256	0.39 / 0.39	2559.89 / 2550.54	6045.12 / 4246.66	1.00×	1.42×
Prefill	1024	1024	0.27 / 0.27	3766.04 / 3756.69	8654.54 / 5838.09	1.00×	1.48×
Decode	256	256	8.03 / 15.95	124.55 / 62.71	493.04 / 246.94	1.99×	2.00×
Decode	256	1024	7.96 / 15.69	125.59 / 63.75	496.69 / 249.27	1.97×	1.99×
Decode	1024	256	7.96 / 15.69	125.59 / 63.75	496.69 / 249.27	1.97×	1.99×
Decode	1024	1024	7.90 / 15.43	126.63 / 64.79	500.34 / 251.60	1.95×	1.99×

Table A12: BF16 / W8A8 deployment results for Qwen2.5-VL-3B. Throughput is in s^{-1}, latency in ms, and energy in mJ. Ratios are relative to BF16 under the same token configuration.

Stage	V tokens	T tokens	Throughput	Latency	Energy	Speedup	Energy ratio
Prefill	256	256	0.60 / 0.63	1663.57 / 1582.40	4802.11 / 2690.07	1.05×	1.79×
Prefill	256	1024	0.25 / 0.26	3959.49 / 3872.80	10486.84 / 5856.24	1.02×	1.79×
Prefill	1024	256	0.23 / 0.23	4439.11 / 4352.42	11419.90 / 6789.30	1.02×	1.68×
Prefill	1024	1024	0.14 / 0.15	6972.05 / 6814.82	17529.33 / 10164.20	1.02×	1.72×
Decode	256	256	4.02 / 8.00	248.69 / 125.07	984.06 / 492.73	1.99×	2.00×
Decode	256	1024	3.99 / 7.88	250.47 / 126.85	989.19 / 496.16	1.97×	1.99×
Decode	1024	256	3.99 / 7.88	250.47 / 126.85	989.19 / 496.16	1.97×	1.99×
Decode	1024	1024	3.96 / 7.77	252.25 / 128.63	994.31 / 499.58	1.96×	1.99×

Table A13: Latency speedup and energy-reduction ranges relative to BF16 across the four token configurations.

Model	Stage	Precision	Latency speedup ↑	Energy reduction ↑
Qwen3-VL-2B	Prefill	W8A8	1.00–1.01x	1.49–1.62x
Qwen3-VL-2B	Prefill	W4A8	1.58–1.76x	2.20–2.70x
Qwen3-VL-2B	Prefill	W4A4	2.53–3.41x	3.13–4.65x
Qwen3-VL-2B	Decode	W8A8	2.00x	2.00x
Qwen3-VL-2B	Decode	W4A8	3.74–3.91x	3.76–3.93x
Qwen3-VL-2B	Decode	W4A4	3.74–3.91x	3.76–3.93x
Qwen2-VL-2B	Prefill	W8A8	1.00–1.02×	1.42–1.60×
Qwen2-VL-2B	Prefill	W4A8	1.48–1.73×	1.94–2.55×
Qwen2-VL-2B	Prefill	W4A4	2.11–3.13×	2.53–4.14×
Qwen2-VL-2B	Decode	W8A8	1.95–1.99×	1.99–2.00×
Qwen2-VL-2B	Decode	W4A8	3.74–3.92×	3.88–3.97×
Qwen2-VL-2B	Decode	W4A4	3.92–3.97×	3.93–3.98×
Qwen2.5-VL-3B	Prefill	W8A8	1.02–1.05×	1.68–1.79×
Qwen2.5-VL-3B	Prefill	W4A8	1.70–1.89×	2.67–3.13×
Qwen2.5-VL-3B	Prefill	W4A4	2.83–3.64×	4.09–5.64×
Qwen2.5-VL-3B	Decode	W8A8	1.96–1.99×	1.99–2.00×
Qwen2.5-VL-3B	Decode	W4A8	3.78–3.93×	3.91–3.97×
Qwen2.5-VL-3B	Decode	W4A4	3.97–3.98×	3.96–3.99×

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions and scope are concretely claimed in the abstract and introduction. Our key contributions are summarized in Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation of our approach in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not present new mathematical theorems, or formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Our method is detailed in the Section 3 and illustrated in Figure 2. Full experimental settings are disclosed in the Section 4 and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Yes, we provide open access to all data in Appendix C and core code necessary to faithfully reproduce the main experimental results presented in this paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Full experimental settings are disclosed in Section 4 and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our experimental results are based on multiple independent runs in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sufficient details of the compute resources required to reproduce our experiments are provided Section 4 and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have thoroughly reviewed the NeurIPS Code of Ethics and confirm that all research presented in this paper strictly adheres to its guidelines, encompassing data privacy, responsible conduct, and transparent reporting of results and limitations.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive societal impacts, which is highlighted in Section 5.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work is not related to high-risk data or models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets used are publicly available, appropriately credited, and their licenses and terms of use are respected. Citations are provided in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not introduce or release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research does not involve crowdsourcing experiments or new studies with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve new studies with human subjects that would require Institutional Review Board (IRB) approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used as an important, original, or non-standard component of the core research methods presented in this paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.