MARK-VLA: MIXED-PRECISION ACTION-RISK-GUIDED QUANTIZATION WITH MIXED-WEIGHT KERNELS FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models couple visual perception, language reasoning, and action generation within a unified policy, but their large model size and computationally intensive inference impose substantial memory and latency overheads. We present MARK-VLA, a post-training quantization framework that enables efficient VLA inference through channel-wise mixed-precision quantization and tailored GPU kernel support for mixed-precision weights. Instead of pursuing extreme compression for individual modules, MARK-VLA applies W2/W4/W8 weight quantization with A8 activation quantization across the major perception-to-action stages to reduce the overall memory footprint. We propose a per-module precision allocation method that automatically determines channel-wise precision based on gradient-based action risk evaluated under a quantized reference policy. To account for the discrepancy between the risk estimated for individual quantization decisions and their combined effect, we adopt a progressive precision reduction scheme that gradually lowers the quantization precision. Finally, to translate these fine-grained precision assignments into practical acceleration, we develop fused mixed-weight GPU kernels together with trace-guided per-layer specialization to optimize the complete execution path of quantized Linear layers. Experiments on LIBERO across six VLA policies and three GPU platforms demonstrate up to $3.62\times$ compression and up to $1.69\times$ end-to-end speedups, while the largest decrease in average success rate is only 0.82 percentage points. These results demonstrate that action-aware precision allocation and deployment-oriented kernel co-design can jointly achieve compact and efficient VLA inference with minimal accuracy degradation.

1 INTRODUCTION

Vision-Language-Action (VLA) models integrate perception, language, and physical execution into a unified policy, representing a promising direction for general-purpose robotics. Pioneering works such as OpenVLA (Kim et al., 2025), the GR00T series (NVIDIA et al., 2025), and the π series (Black et al., 2025) have shown strong performance in bridging vision-language reasoning with robot control. Yet, deploying these large models on resource-constrained edge robotic systems poses significant challenges. The heavy vision-language backbones impose high memory and bandwidth demands, while processing multi-view images and multimodal token streams leads to considerable inference latency. Furthermore, some architectures adopt DiT or flow-matching based action experts that must be invoked repeatedly to produce each action chunk, adding substantial computational overhead to the closed-loop control pipeline. Thus, for practical deployment, it is essential to reduce both memory usage and latency while preserving the accuracy and reliability of the generated actions.

While several recent works have explored quantization for VLA models, existing methods often achieve low-bit compression only for selected modules of the policy. QuantVLA (Zhang et al., 2026) excludes the vision encoder and DiT attention projections, while QVLA (Xu et al., 2026) retains the projector and action head in floating point. Consequently, substantial parts of the policy remain uncompressed, and reducing the precision of the quantized modules does not translate proportionally into smaller whole-model storage. The resulting complete policy can therefore remain considerably

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

larger than its low-bit modules alone would suggest. Achieving compact whole-model deployment requires extending low-bit quantization across all major stages of the perception-to-action pipeline, rather than restricting it to selected modules.

Mainstream quantization methods, including those designed for VLAs, increasingly adopt fine-grained mixed-precision schemes, such as channel-wise quantization, to achieve higher compression ratios. To determine the precision of each channel, various proxy metrics have been proposed to estimate the impact of different quantization choices, and these estimates are then used by the allocation algorithm to assign the final precision to each channel. Designing such a proxy involves three key considerations: accuracy, evaluation efficiency, and scalability. Ideally, the proxy should reliably reflect the impact of quantization with limited calibration overhead, while remaining applicable across different network architectures without substantial loss of fidelity. Beyond proxy design itself, another challenge lies in precision allocation. Since the proxy typically estimates the impact of individual quantization decisions in isolation, the allocation algorithm must determine how to combine these local estimates into a globally effective mixed-precision configuration, as the combined effect of multiple precision reductions may differ from their individually estimated effects. Furthermore, model compression alone does not necessarily translate into faster inference. Activation preprocessing, quantization, and mixed-weight execution introduce additional overhead around low-bit matrix multiplication, which can offset the latency reduction achieved by faster GEMM. Existing VLA quantization methods have yet to jointly address these challenges within a unified framework that delivers end-to-end model compression and practical inference acceleration.

In this paper, we present MARK-VLA, a post-training quantization framework that integrates a calibration-efficient proxy, full model mixed-precision quantization, and GPU kernel optimization. MARK-VLA provides an automatic and scalable workflow that can directly transform a full-precision VLA model into a deployable quantized model with end-to-end inference acceleration. Our main contributions are summarized as follows:

- **Precise and Calibration-Efficient Action Risk.** We propose a quantized-reference action-risk estimator that computes the full action gradient through the complete VLA and diffusion denoising process, while requiring only a small calibration set. This provides more faithful and scalable sensitivity estimation with substantially lower calibration overhead.
- **Per-Module Progressive Precision Allocation.** We introduce a per-module progressive allocation strategy that avoids unreliable risk comparisons across heterogeneous VLA modules and mitigates overly aggressive low-bit assignments. By progressively reducing precision from W8 to W4 and then W2, MARK-VLA achieves more balanced compression under a given memory budget.
- **Mixed-Weight GPU Kernels and Trace-Guided Specialization.** We develop fused activation-quantization and mixed-weight GEMM kernels together with trace-guided specialization that maps each quantized layer to a target-GPU-specific execution configuration, translating fine-grained mixed-precision compression into practical end-to-end acceleration.

2 Related Work

2.1 Post-Training Quantization for VLA Models.

QuantVLA (Zhang et al., 2026) combines selective W4A8 quantization with scale corrections. Its reported LLM+DiT memory excludes the vision encoder and therefore does not represent complete-policy storage. Its DuQuant-based preprocessing employs general orthogonal rotations that introduce nontrivial online transformation overhead. MARK-VLA instead uses structured Hadamard rotations, whose fast Hadamard transform provides a lower-overhead implementation of the required online rotation.

QVLA (Xu et al., 2026) introduces action-guided channel-wise precision allocation, but its analytical proxy models quantization error as isotropic noise and subsequently refines selected sensitivities through additional full forward evaluations. Its deployment configuration further depends on heuristic tuning of the precision mixture, with different proportions of multiple bit-widths considered even under the same nominal bit budget. MARK-VLA likewise requires empirical selection of

deployment budgets, but restricts this tuning to module-wise storage budgets. Once these budgets are specified, channel assignments and the resulting precision proportions are determined algorithmically from candidate-specific action risk, without targeted full-forward sensitivity refinement or manual specification of the precision mixture.

ActQuant (Akbari et al., 2026) performs action-guided mixed-precision weight quantization only on the vision encoder and language backbone, while retaining the projector and action head in floating point. Its weight-only formulation also leaves activations outside the low-bit quantization scheme. Consequently, its aggressive low-bit compression remains confined to the backbone weights rather than the complete perception-to-action pipeline. MARK-VLA instead extends weight–activation quantization across the major stages of the pipeline, including cross-modal and action-generation linear operators.

3 METHODOLOGY

3.1 PRELIMINARY

Symmetric quantization. Quantization is a technique widely used for efficient approximate computation, mapping the high-precision parameters of a network to low-precision integer representations to enable fast deployment. Among various quantization schemes, symmetric quantization is particularly favored due to its simplicity in subsequent computations. Given a real-valued scalar r and a scale $s > 0$, the signed symmetric b-bit quantizer is:

$$\mathcal{Q}_b(r; s) = \text{clip}\left(\text{round}\left(\frac{r}{s}\right), -q_b^{\max}, q_b^{\max}\right), \quad q_b^{\max} = 2^{b-1} - 1 \tag{1}$$

For a group of values $\mathcal{G}$, the scale is calculated as follows:

$$s_b(\mathcal{G}) = \frac{\max(\max_{r\in\mathcal{G}} |r|, \varepsilon)}{q_b^{\max}} \tag{2}$$

Smooth scaling and orthogonal rotation. The effectiveness of symmetric quantization is highly dependent on the underlying distribution. Activation outliers, which commonly exist in LLM-related models, necessitate high precision to fully cover the data range. Channel-wise smoothing (Xiao et al., 2023) and orthogonal rotation (Ashkboos et al., 2024) are thus widely adopted in LLM quantization to enable aggressive bit-width reduction. Let $\mathbf{s} \in \mathbb{R}^K_{>0}$ denote a positive channel-wise smoothing vector, where K is the input width of the linear layer. The smoothing transformation is

$$Y = \left(X \,\text{diag}(\mathbf{s})^{-1}\right)\left(W \,\text{diag}(\mathbf{s})\right)^\top = \hat{X}\hat{W}^\top. \tag{3}$$

To further suppress the outliers, an orthogonal rotation can be introduced. By applying a fixed, normalized blockwise Hadamard matrix $R \in \mathbb{R}^{K\times K}$, which satisfies

$$RR^\top = R^\top R = I_K. \tag{4}$$

the linear operation can be reformulated as:

$$Y = (\hat{X}R)(\hat{W}R)^\top = \bar{X}\bar{W}^\top \tag{5}$$

Subsequently, quantization is applied to $\bar{X}$ and $\bar{W}$, whose values are now more evenly distributed, thereby enabling more aggressive bit-width reduction.

3.2 ACTION-RISK-GUIDED QUANTIZATION

As illustrated in Fig. 1, MARK-VLA employs a channel-wise mixed-precision quantization scheme, with activations uniformly quantized to 8 bits and weights quantized per output channel, with bit-width $p_i \in \{8, 4, 2\}$, where p_i abbreviates $p_{\ell,n}$ for output channel n of layer ℓ.

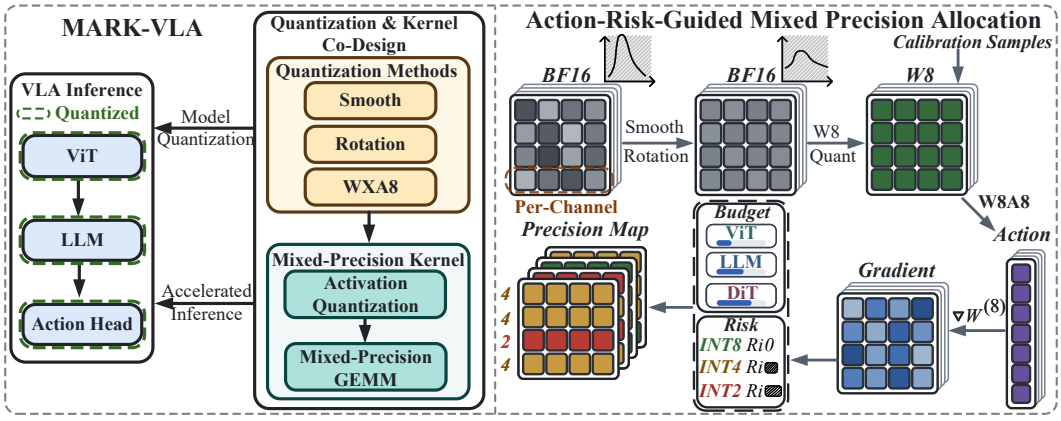


Figure 1: Overview of MARK-VLA and action-risk-guided mixed-precision allocation.

3.2.1 Action-risk estimation

Gradient-based sensitivity is commonly used to assess quantization impact and guide mixed-precision allocation. For VLA models, this sensitivity can be measured at the action level as action risk. Prior work such as QVLA uses action risk for precision evaluation, but requires a large calibration set for generalizable estimation and therefore approximates the gradient by neglecting cross-element terms to reduce the cost.

To better balance accuracy, efficiency, and scalability, MARK-VLA revisits action-risk estimation from three aspects. First, we use a W8A8 quantized reference policy instead of the full-precision model, so that the estimated risk reflects activation quantization and deployment transformations such as smoothing and rotation. Second, MARK-VLA directly computes the full action gradient through standard backpropagation; for diffusion-based action generation, gradients are further propagated through the iterative denoising process to capture the accumulated impact across diffusion steps. Third, we use only a small calibration set, motivated by our observation that relative channel sensitivities remain largely consistent across independently sampled subsets. Experiments confirm that this provides reliable precision guidance with substantially fewer calibration samples.

In summary, MARK-VLA evaluates the first-order action perturbation induced by each candidate precision reduction and defines its risk as

$$q_{i,c}^{(b)}(x) = g_{i,c}(x)(\bar{w}_i^{(b)} - \bar{w}_i^{(8)})^\top, \quad \mathcal{R}_i^{(b)} = \frac{\sum_{x\in\mathcal{D}_{\text{cal}}}\sum_{c=1}^{D}(q_{i,c}^{(b)}(x))^2}{|\mathcal{D}_{\text{cal}}|D}. \tag{6}$$

Here, $g_{i,c}(x) = \nabla_{\bar{w}_i^{(8)}} a_c(x)$ denotes the row-vector gradient of action coordinate c with respect to channel i's dequantized W8 weights, evaluated in the W8A8 reference policy. The difference $\bar{w}_i^{(b)} - \bar{w}_i^{(8)}$ is the actual weight residual induced by the candidate precision reduction. More specifically, for a diffusion-based action module, an action chunk with H timesteps and d components per timestep is flattened into $D = Hd$ action coordinates. Accordingly, $\mathcal{R}_i^{(b)}$ measures the mean squared first-order action perturbation over calibration samples and action coordinates.

3.2.2 Per-Module Progressive Precision Allocation

For mixed-precision allocation based on first-order gradient risk, a common strategy is to progressively reduce channel precision in ascending order of estimated risk. However, while the gradient reflects the impact of quantization on the action output, its absolute magnitude can also depend on factors such as network depth and the data processed by each module. Empirically, we observe that the risk distribution of the DiT module is noticeably shifted relative to that of the LLM module, suggesting that risk values across heterogeneous VLA modules may lie on different scales and therefore may not be directly comparable.

As a result, MARK-VLA performs precision allocation independently within the vision encoder, language model, and action module, each under its own storage budget. Within each module, all target channels are initialized at W8 and progressively demoted according to their action risk. For each eligible transition from b_s to b_t, we define the incremental action risk per saved storage bit as

$$\mu_i^{b_s \to b_t} = \frac{\mathcal{R}_i^{(b_t)} - \mathcal{R}_i^{(b_s)}}{C_i(b_s) - C_i(b_t)}, \tag{7}$$

where $C_i(b)$ denotes the storage cost of channel i at precision b. A smaller $\mu_i^{b_s \to b_t}$ indicates a more favorable precision reduction with less additional action risk per unit of storage saving.

Allocation then proceeds progressively within each module. MARK-VLA first ranks all $W8 \to W4$ transitions in ascending order of $\mu_i^{8 \to 4}$ and applies them until the module budget is satisfied. Only if the budget remains unmet after all target channels have reached W4 does the allocator activate the $W4 \to W2$ stage. This progressive design is motivated by the fact that channel risks are estimated independently and do not explicitly capture interactions among multiple quantization decisions. By avoiding aggressive low-bit assignments to a small subset of channels at an early stage, the progression distributes compression more gradually and reduces the risk of over-compression.

3.3 Trace-Compiled GPU Execution via Two-Stage Kernel Search

In practical deployment, both the mixed-precision configuration produced by the preceding allocation stage and the available resources of the target GPU can vary, making efficient execution highly dependent on the resulting activation shapes, weight-precision composition, and hardware characteristics. MARK-VLA therefore constructs a compositional portfolio of fused activation-quantization and mixed-weight GEMM kernels, and uses inference traces to compile a static execution plan for each quantized layer. Based on the observed workload characteristics and target-GPU resources, each layer is mapped to a specialized kernel configuration that is fixed before deployment.

3.3.1 Activation quantization kernel candidates

In the quantized execution flow, activations must be requantized on the fly, since operations between successive GEMMs are typically executed in BF16 on GPUs. To reduce this repeated conversion overhead, MARK-VLA adopts fused kernels that combine activation smoothing, rotation, and quantization, with two variants for different execution scenarios (Fig. 2(a)). **No-Store** exploits the compact Hadamard structure by assigning each activation row to a single CTA, keeping intermediate data in shared memory and avoiding global-memory traffic; however, wide activations with few rows require sequential channel tiling, limiting parallelism. **Block-Staged** instead partitions each row across multiple CTAs to expose more parallelism, at the cost of a global workspace for token-wise quantization and additional memory traffic. Efficient execution therefore depends on matching the activation shape and exposed parallelism to the compute and memory resources of the target GPU.

3.3.2 Mixed-precision GEMM kernel candidates

The GEMM kernel consumes on-the-fly quantized activations and prepacked mixed-precision weights. Since the use of INT8 activations constrains the underlying GEMM to INT8 execution, low-bit weights stored compactly in DRAM must be unpacked to INT8 on the fly, preserving memory and DRAM-traffic savings while keeping decoding overhead low (Lin et al., 2025). MARK-VLA therefore provides three kernel families that trade off storage compactness, unpacking cost, and hardware utilization across different precision layouts and matrix shapes (Fig. 2(b)).

True-Bit Grouped stores W2/W4/W8 channels separately and processes them independently, providing the most compact storage for irregular precision mixtures but incurring higher decoding overhead and typically relying on CUDA Cores. *Uniform-W4 GEMM* targets regular all-W4 tiles, where efficient sign extension and regular packing enable effective Tensor-Core execution. *Promoted Main/Side* targets mixed-precision tiles by promoting W2 weights to W4 for efficient decoding and Tensor-Core mapping while keeping W8 channels separate, trading a small amount of compression efficiency for better hardware utilization.

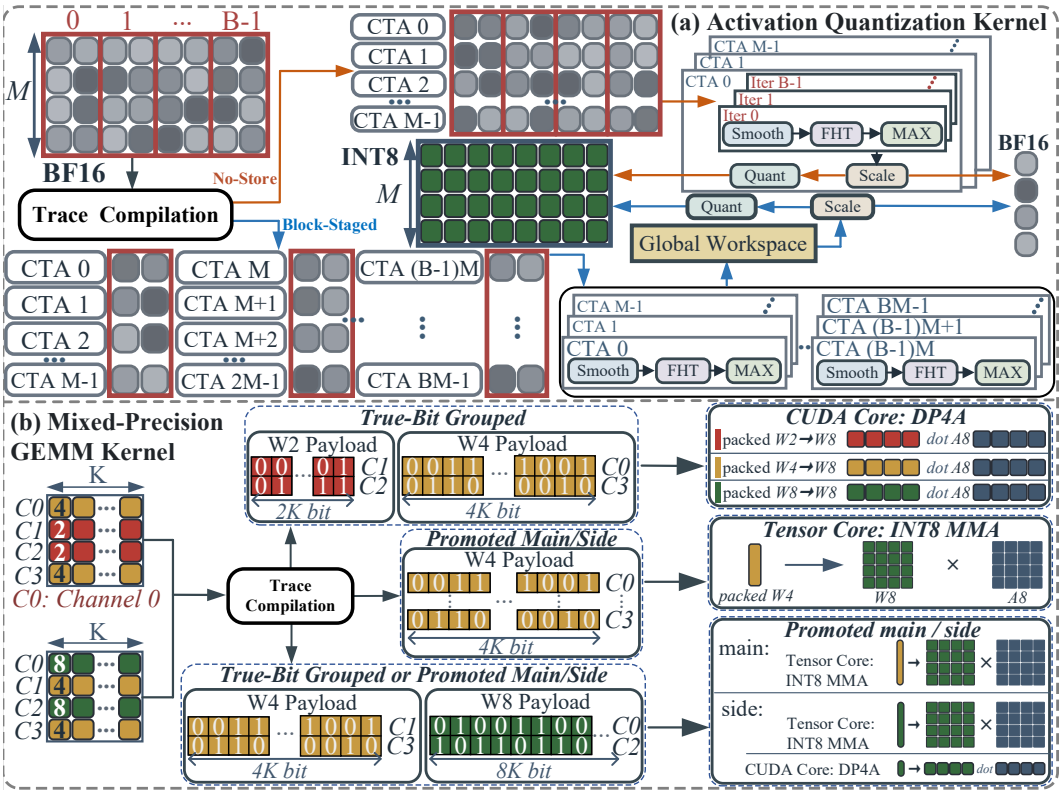


Figure 2: (a) Activation quantization kernel and (b) mixed-precision GEMM kernel candidates.

3.3.3 Two-stage trace compilation for per-layer specialization

To select and specialize kernels for each quantized layer, MARK-VLA first traces inference to collect observed input shapes and invocation frequencies, then performs a two-stage search over activation-quantization and mixed-weight GEMM kernels. In Stage 1, candidate kernels are benchmarked separately for each shape, and pairwise latency is estimated and aggregated according to invocation frequency. In Stage 2, the top-k pairs are benchmarked end-to-end through CUDA Graph replay using the deployment execution path. The lowest-latency configuration is then statically bound to each layer, forming the trace-compiled execution plan used during inference.

4 Experiments

4.1 Experimental Setup

We evaluate MARK-VLA on six VLA configurations covering different vision-language backbones and action-generation architectures across the four LIBERO suites (Liu et al., 2023). Among them, OpenVLA-OFT and GR00T N1.5 use suite-specific checkpoints, while each Qwen-based StarVLA policy (Ye et al., 2026) uses a single checkpoint jointly trained across all four suites.

For quantization, calibration samples are drawn from the offline LIBERO demonstration datasets. For each suite, we sample two tasks, two episodes per task, and one timestep per episode, yielding four samples. Each suite-specific OpenVLA-OFT and GR00T N1.5 checkpoint uses these four samples, whereas each Qwen-based checkpoint uses 16 samples across the four suites to derive a shared quantization configuration.

Evaluation is conducted across platforms with different hardware and software configurations. For experiments performed on multiple platforms, we report the mean success rate across environments. Controlled comparisons and ablation studies are conducted within the same environment using identical settings for all compared variants.

4.2 MAIN RESULTS

Table 1: Accuracy, Compression and Speedup of MARK-VLA on LIBERO.

Model	Method	LIBERO					Weights	Weight	Speedup		
		Spatial	Object	Goal	Long	Avg.	(GiB)	saving	4090	6000	A800
GR00T N1.5	BF16	90.80	97.60	93.67	85.80	91.97	7.07	1.00×	1.00×	1.00×	1.00×
	QuantVLA	93.60	88.80	86.80	81.20	87.60	5.81	1.22×	—	—	—
	MARK-VLA	90.53	96.93	92.67	85.13	91.32	3.16	2.24×	1.54×(1.51×)	1.34×(1.31×)	1.37×(1.40×)
OpenVLA-OFT	BF16	96.80	97.60	96.40	93.67	96.12	14.84	1.00×	1.00×	1.00×	1.00×
	QVLA*	96.20	97.60	96.40	93.80	96.00	4.81	3.09×	—	—	—
	MARK-VLA	96.33	97.67	97.80	93.47	96.32	4.10	3.62×	1.63×(1.53×)	1.19×(1.12×)	1.02×(1.09×)
Qwen2.5-VL-GR00T	BF16	97.40	99.00	96.40	92.07	96.22	7.86	1.00×	1.00×	1.00×	1.00×
	MARK-VLA	95.87	98.53	96.93	92.33	95.92	2.25	3.49×	1.69×(1.69×)	1.41×(1.50×)	1.49×(1.49×)
Qwen2.5-VL-OFT	BF16	95.87	98.00	97.27	89.07	95.05	7.65	1.00×	1.00×	1.00×	1.00×
	MARK-VLA	95.80	98.47	97.53	91.00	95.70	2.24	3.41×	1.66×(1.82×)	1.38×(1.33×)	1.32×(1.31×)
Qwen3-VL-PI	BF16	96.07	98.93	97.93	95.40	97.08	15.24	1.00×	1.00×	1.00×	1.00×
	MARK-VLA	95.20	99.07	96.73	94.07	96.27	4.78	3.19×	1.63×(1.61×)	1.36×(1.42×)	1.22×(1.49×)
Qwen3-VL-OFT	BF16	96.73	99.53	98.33	95.20	97.45	9.11	1.00×	1.00×	1.00×	1.00×
	MARK-VLA	95.93	99.93	97.93	93.07	96.72	2.70	3.37×	1.57×(1.39×)	1.16×(1.13×)	1.18×(1.22×)

Table 1 reports the success rates on the four LIBERO suites and their arithmetic mean, the complete-policy weight size and reduction relative to BF16, and the end-to-end policy speedups on different GPU platforms. The QuantVLA results are based on our reproduction, whereas the QVLA results marked with an asterisk are taken directly from the original paper. For existing methods that leave certain modules unquantized, weight size is computed over the complete parameter set, including both quantized weights and parameters retained in floating point. MARK-VLA quantizes all major modules and achieves up to $3.62\times$ weight reduction, while the largest decrease in average success rate is only 0.82 percentage points.

We further compare our result with the cuBLASLt uniform-W8A8 results are shown in parentheses, with unavailable results marked by a dash. MARK-VLA achieves up to $1.69\times$ end-to-end speedup over BF16, while matching or outperforming the highly optimized cuBLASLt-based uniform-W8A8 implementation in most evaluated settings. This demonstrates that fine-grained mixed-weight quantization can provide substantially greater weight compression while maintaining competitive end-to-end inference efficiency.

MARK-VLA and cuBLASLt achieve smaller speedups on the RTX 6000 Ada and A800 because dense BF16 execution benefits more from the greater compute and memory resources of these GPUs, whereas quantized inference additionally involves low-arithmetic-intensity operations such as activation quantization, weight unpacking, and dequantization that scale less effectively with these resources. The small and irregular GEMM shapes in batch-1 VLA inference further limit the available parallelism.

4.3 QUANTIZATION PERFORMANCE AND EFFICIENCY

We further compare MARK-VLA with the state-of-the-art QVLA method, which also adopts action risk for precision evaluation but estimates it using an approximate gradient over a large calibration set on FP reference policy. QVLA performs global greedy precision allocation by progressively reducing channel precision according to action risk over a broader set of candidate bit-widths. We evaluate both methods on the Qwen-based policy using an NVIDIA RTX 4090, as this setting provides a single checkpoint shared across all LIBERO suites and therefore enables a more general comparison. Since neither the original QVLA paper nor its released implementation directly supports this setting, we adapt the public code accordingly.

Table 2 reports a controlled comparison between MARK-VLA and QVLA. To isolate the effects of risk estimation and precision allocation, we restrict both methods to the same W2/W4/W8 precision candidates and additionally incorporate smoothing and rotation into QVLA, although these transformations are not included in its original implementation. Without these enhancements, QVLA exhibits further performance degradation under quantization. For each model, we tune QVLA's global target average bit-width to match the average LIBERO success rate of MARK-VLA as closely as possible, and then compare the resulting complete-policy weight sizes. At comparable success rates, MARK-VLA achieves $1.34\times$–$1.74\times$ higher compression than QVLA.

Table 2: Quantization performance and efficiency

Model	Quantization performance					Risk-estimation efficiency				
	QVLA Avg	MARK-VLA Avg	QVLA Weights (GiB)	MARK-VLA Weights (GiB)	Weight saving	QVLA 1 GPU (s)	MARK-VLA 1 GPU (s)	MARK-VLA 8 GPUs (s)	Speedup 1 GPU	Speedup 8 GPUs
Qwen2.5-VL-OFT	94.40	94.75	3.80	2.24	1.69×	991.82	161.30	31.82	6.15×	31.17×
Qwen3-VL-OFT	96.00	95.75	4.43	2.70	1.64×	741.60	197.49	37.15	3.76×	19.96×
Qwen2.5-VL-GR00T	94.40	95.40	3.92	2.25	1.74×	1615.64	254.68	38.18	6.34×	42.32×
Qwen3-VL-PI	94.30	94.60	6.42	4.78	1.34×	2755.64	415.71	74.32	6.63×	37.08×

The efficiency columns of Table 2 compare the wall-clock cost of risk estimation between MARK-VLA and the released QVLA implementation adapted to the four Qwen-based VLA models. On a single GPU, MARK-VLA achieves a $3.76\times$–$6.63\times$ speedup over QVLA across the four models. We further develop a parallel implementation of risk estimation, which reduces the calibration time to only 1–2 minutes when using eight GPUs.

4.4 Ablation of Risk Estimation and Precision Allocation

We ablate the action-risk estimator and precision-allocation strategy on GR00T N1.5 and Qwen2.5-OFT. All variants use the same calibration data, quantization configuration, and storage budgets as in the main experiments. For risk estimation, *Diagonal* removes the within-channel cross-element terms from our full action-risk formulation, mimicking the gradient approximation used in QVLA. For precision allocation, *Global* ranks risks across all modules and performs allocation under a whole-model storage budget equal to the sum of the original per-module budgets. *Direct* removes the proposed progressive W8→W4→W2 constraint, allowing individual channels to be reduced directly to W2 before all channels in the module have reached W4.

Table 3: Ablation of risk estimation and precision allocation.

Model	Risk	Per-Module Progressive	Per-Module Direct	Global Progressive	Global Direct
GR00T N1.5	Full	90.60	77.15	75.30	0.00
	Diagonal	90.20	76.05	74.65	0.05
Qwen2.5-OFT	Full	94.75	83.60	94.65	79.10
	Diagonal	94.65	65.15	94.70	78.40

As shown in Table 3, the full-gradient risk estimator achieves higher success rates than the diagonal approximation in most cases across the tested allocation settings. Under the same risk formulation, progressive precision reduction consistently outperforms direct reduction, regardless of whether allocation is performed per module or globally. Similarly, when using the same precision-reduction strategy, per-module allocation exhibits more stable performance across models than global allocation. These results validate the effectiveness of the proposed full-gradient risk estimation, per-module allocation, and progressive W8→W4→W2 precision reduction strategy.

4.5 Calibration Efficiency and Stability

As discussed in Sec. 3.2.1, the use of a small calibration set is motivated by the observation that relative channel sensitivities remain stable across independently sampled subsets. To verify this, we plot the W2 and W4 action-risk distributions of the vision, language, and action modules of Qwen2.5-VL-GR00T across five calibration seeds, each using 16 samples. As shown in Fig. 3, channels are first sorted in ascending order of risk under Seed 7, and the risks estimated from the other four seeds are then plotted using the same channel order. Despite variations in absolute risk values, the overall patterns remain highly consistent across seeds, with no obvious outliers or major trend reversals. Since our allocation depends primarily on relative rather than absolute risk, these results provide empirical support for using a small calibration set for risk estimation.

We further evaluate the sufficiency of a small calibration set by varying the number of calibration samples. Table 4 reports the success rates of models compressed to the same target sizes as in the default setting, with action risks estimated using different calibration-set sizes. The results show that the default calibration set already provides reliable precision guidance across all six VLA configurations. Increasing the calibration data by $5\times$ to cover all LIBERO tasks does not yield consistent

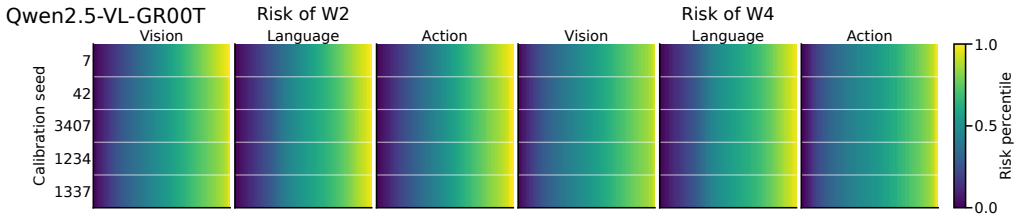


Figure 3: **Action-risk stability across calibration seeds.** Relative W2/W4 channel risks of Qwen2.5-VL-GR00T across five calibration seeds.

accuracy gains, while halving the calibration set causes little performance degradation. We use the default setting in our main experiments to provide broader task and episode coverage while keeping the calibration overhead negligible. Overall, these results support the sufficiency of small calibration sets for action-risk estimation.

Table 4: Calibration efficiency.

Model	Samples	LIBERO				
		Spatial	Object	Goal	Long	Avg.
GR00T-N1.5	2	86.60	97.00	89.60	83.00	89.05
	4	89.20	97.20	90.80	85.20	90.60
	20	85.00	97.00	94.00	87.00	90.75
OpenVLA-OFT	2	92.80	97.00	96.80	92.80	94.85
	4	93.80	96.40	97.60	92.80	95.15
	20	93.20	97.20	96.80	92.00	94.80
Qwen2.5-VL-GR00T	8	91.20	99.60	97.60	93.80	95.55
	16	93.40	98.80	97.40	92.00	95.40
	80	93.80	99.20	96.80	92.20	95.50
Qwen2.5-VL-OFT	8	91.20	99.00	97.40	90.20	94.45
	16	90.80	99.40	97.80	91.00	94.75
	80	91.80	98.80	96.80	86.20	93.40
Qwen3-VL-PI	8	90.60	99.00	90.40	93.80	93.45
	16	90.80	98.40	96.00	93.20	94.60
	80	92.00	99.60	96.00	94.00	95.40
Qwen3-VL-OFT	8	92.40	99.80	98.20	94.20	96.15
	16	91.60	99.80	97.60	94.00	95.75
	80	91.00	100.00	98.00	94.40	95.85

Table 5: Two-stage search efficiency.

Model	GPU	Full / Two-stage	
		Search speedup	Latency ratio
GR00T-N1.5	4090	6.96×	0.975×
	6000	4.85×	0.997×
	A800	5.01×	0.995×
OpenVLA-OFT	4090	2.91×	1.003×
	6000	2.12×	1.001×
	A800	3.09×	0.996×
Qwen2.5-VL-GR00T	4090	2.59×	0.994×
	6000	1.53×	0.963×
	A800	2.50×	1.023×
Qwen2.5-VL-OFT	4090	2.97×	0.993×
	6000	1.59×	1.088×
	A800	2.51×	0.971×
Qwen3-VL-PI	4090	2.93×	0.984×
	6000	2.10×	0.990×
	A800	3.09×	0.992×
Qwen3-VL-OFT	4090	2.70×	1.026×
	6000	1.76×	0.981×
	A800	2.89×	0.980×

4.6 Efficiency of Two-Stage Trace Compilation

We compare Two-stage search with Full search across six VLA models and three GPUs, reporting search speedup (Full/Two-stage search time) and end-to-end inference latency ratio (Full/Two-stage latency). As shown in Table 5, Two-stage search achieves $1.53\times$–$6.96\times$ search speedup, while the execution plans selected by Two-stage search achieve nearly identical end-to-end inference latency to Full search, demonstrating substantially reduced search cost with negligible loss in deployment efficiency.

5 Conclusion

In this paper, we presented MARK-VLA, a deployment-oriented PTQ framework that co-designs channel-wise mixed-precision quantization with executable GPU kernels for heterogeneous VLA models. MARK-VLA quantizes activations to INT8 and assigns W2/W4/W8 precision to weight channels through module-wise action-risk estimation, while trace-compiled kernel selection maps the resulting irregular layouts to target GPU workloads. Experiments on LIBERO show that MARK-VLA reduces static weight storage by up to $3.62\times$ while maintaining average success rates within 0.82 percentage points of the corresponding BF16 baselines. The specialized mixed-weight execution further provides up to $1.69\times$ end-to-end speedup. Overall, these results demonstrate that jointly considering action-level sensitivity, precision allocation, and executable low-bit kernels can translate VLA compression into practical storage and latency benefits on general-purpose GPUs.

AI USE STATEMENT

We used AI tools for language editing to improve the clarity and readability of the manuscript, as well as for software development assistance, including code generation, modification, and debugging. All AI-assisted content and code were reviewed and verified by the authors.

REFERENCES

Arash Akbari, Arman Akbari, Masih Eskandar, Qitao Tan, Yixiao Chen, Jingwu Luo, Bertha Pangaribuan, Liyun Zhang, Jennifer Dy, Geng Yuan, Xue Lin, Gaowen Liu, Stratis Ioannidis, and Yanzhi Wang. ActQuant: Sub-4-bit action-guided quantization for Vision-Language-Action models, 2026. URL https://arxiv.org/abs/2605.24011.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. *Proceedings of Machine Learning and Systems*, 7, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

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*, pp. 38087–38099. PMLR, 2023.

Yuhao Xu, Yantai Yang, Zhenyang Fan, Yufan Liu, Yuming Li, Bing Li, and Zhipeng Zhang. QVLA: Not all channels are equal in vision-language-action model's quantization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TpL2nXanru.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. StarVLA-α: Reducing complexity in vision-language-action systems. In *European Conference on Computer Vision (ECCV)*, 2026.

Jingxuan Zhang, Yunta Hsieh, Zhongwei Wan, Haokun Lin, Xin Wang, Ziqi Wang, Yingtie Lei, and Mi Zhang. QuantVLA: Scale-calibrated post-training quantization for Vision-Language-Action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 39539–39549, 2026.