BEYOND ACTION DEVIATION: EXECUTION-LEVEL QUANTIZATION SENSITIVITY FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training quantization (PTQ) offers a practical way to reduce the deployment cost of vision-language-action (VLA) models. When uniform low precision cannot preserve control fidelity, mixed-precision PTQ relies on sensitivity estimation to identify which layers should retain higher precision. Existing sensitivity criteria mainly measure internal numerical distortion or action-level deviation, overlooking how quantization perturbations propagate through robot execution and influence task performance. Consequently, mixed-precision quantization guided by these criteria can lead to substantial performance degradation, particularly when applied to the sensitive action generation modules. We identify the need for an intermediate perspective between action deviations and task outcomes that captures robot-dependent execution information. To this end, we introduce Execution-Level Quantization Sensitivity (**EQS**), a layer-wise sensitivity criterion for mixed-precision quantization of VLA action generators. It models how quantization-altered action commands translate into horizon-level motion discrepancy and evaluates the discrepancy's execution significance under the current robot configuration. Together, these components provide an execution-aware sensitivity signal for layer prioritization. Experiments across simulation benchmarks and real-robot manipulation tasks show that EQS exceeds the best-performing baseline by **0.44**$\sim$**8.06** percentage points in average task success, with gains reaching **33.82** percentage points over individual baselines.

1 INTRODUCTION

Vision-Language-Action (VLA) models have emerged as a powerful paradigm for general-purpose robotic manipulation, unifying visual perception, language-conditioned reasoning, and action generation within a single policy (Kim et al., 2024; Black et al., 2024). Their increasing model scale, however, introduces substantial memory and computational costs for deployment. Post-training quantization (PTQ) offers an attractive way to reduce these costs without retraining (Liu et al., 2021). VLA's action generation module is a particularly consequential quantization target because it directly produces the action commands used for robotic control. Recent studies expose low-bit fragility in this part of the policy: action generation modules are often protected (Akbari et al., 2026), selectively quantized (Zhang et al., 2026), or require specialized quantization and calibration (Wang et al., 2026). When a uniform low-precision configuration does not meet the required control fidelity, mixed-precision PTQ provides a natural alternative by retaining higher precision for sensitive layers while quantizing less critical ones more aggressively (Xu et al., 2026a). This makes reliable layer-wise quantization sensitivity estimation a central requirement.

As illustrated in Fig. 1(a), quantization effects can be assessed at different levels. Conventional PTQ methods primarily focus on the *layer level*, characterizing quantization effects through internal numerical fidelity, such as layer-wise reconstruction (Frantar et al., 2022) or activation statistics (Xiao et al., 2023). For VLA policies, however, layer-level fidelity cannot fully capture the consequences of quantization: numerical perturbations propagate beyond the model and can affect subsequent robot execution. Quantization sensitivity therefore needs to be **informative about downstream effects** on robot execution and task outcomes while remaining practical to estimate during calibration.

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

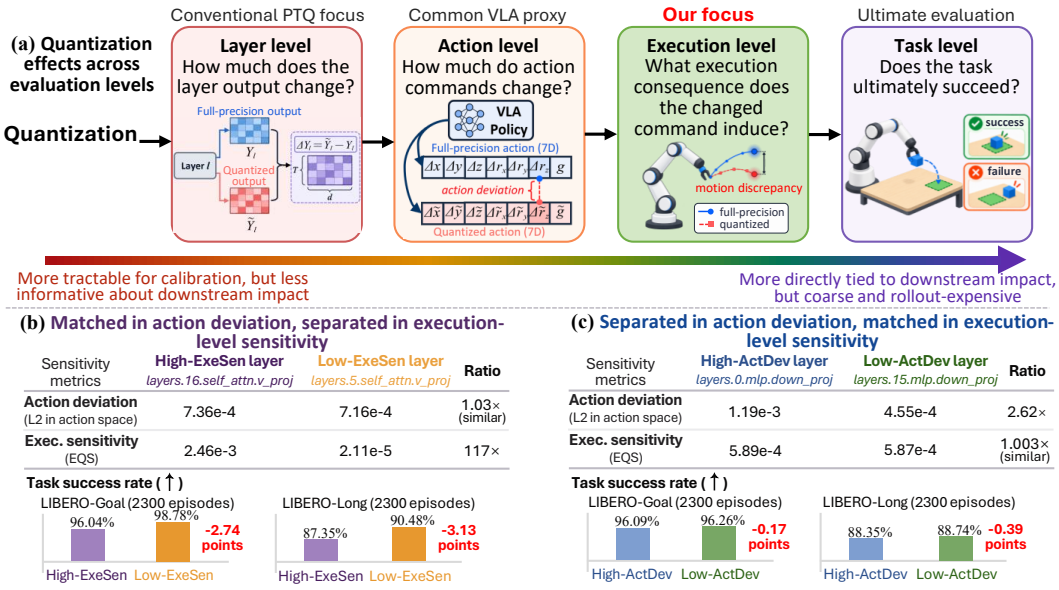


Figure 1: Motivation for execution-level quantization sensitivity. (a) Quantization effects can be assessed at different levels. We formulate the execution level as an intermediate sensitivity target between action and task levels, which is calibration-tractable and informative about how quantization affects robot execution. (b) Layers with nearly identical action deviations exhibit substantially different execution-level sensitivity scores, with the higher-scoring layer yielding lower task success rates when quantized. (c) Conversely, layers with substantially different action deviations exhibit nearly identical execution-level sensitivity scores and observed task success rates.

Accordingly, recent works have adopted *action-level criteria* (Xu et al., 2026b; Akbari et al., 2026) as tractable proxies that assess how quantization affects the action commands produced by the VLA policy. However, action deviations alone do not fully characterize how quantization-induced changes in action commands affect robot execution. The ultimate measure of downstream impact is *task-level outcome*, but this signal is expensive to obtain during calibration as it requires closed-loop rollouts (Xu et al., 2026a). Together, these limitations reveal a gap between action-level sensitivity and task-level outcomes: *existing criteria lack an intermediate, calibration-tractable sensitivity target that captures how quantization-induced action deviations affect robot execution.*

To address this gap, we formulate the ***execution level*** as an intermediate sensitivity target between the action and task levels, incorporating robot-dependent execution information absent from action deviations alone. To empirically motivate this execution-level perspective, we select two pairs of layers in the action generation module of $\pi_{0.5}$ (Intelligence et al., 2025) for matched-pair analyses, quantizing one layer at a time within each pair while keeping all other layers' precision fixed. In Fig. 1(b), two layers with nearly identical action deviations differ by 117× in execution-level sensitivity, measured using the criterion detailed in Sec. 3. Quantizing the higher-sensitivity layer yields success rates 2.74 and 3.13 percentage points lower on LIBERO-Goal and LIBERO-Long (Liu et al., 2023). Conversely, in Fig. 1(c), two layers with a 2.62× difference in action deviation have nearly identical execution-level sensitivity, with success-rate differences of only 0.17 and 0.39 percentage points. These observations suggest that action deviation alone does not fully characterize the downstream impact of quantization perturbations, whereas execution-level sensitivity provides an assessment consistent with the observed task-success patterns.

Two considerations help explain why action- and execution-level assessments can diverge. First, action deviation measures numerical changes in action commands, but the resulting motion discrepancy depends on how these commands compose across successive execution steps. Second, this discrepancy needs to be evaluated according to the robot's current configuration, because motion discrepancies of comparable magnitude can have different execution significance depending on the operational-space inertia and motion direction (Khatib, 1987). Together, these considerations define the requirements for an execution-level sensitivity criterion.

Guided by these requirements, we introduce **Execution-Level Quantization Sensitivity (EQS)**, a layer-wise criterion for mixed-precision allocation in VLA action generation modules (Fig. 2). EQS first translates quantization-induced action deviations into horizon-level end-effector motion discrepancies through ordered motion composition. It then evaluates the configuration-dependent execution significance of these discrepancies using an operational-space inertial metric. We further apply data-driven balancing for heterogeneous action components and convert the execution-level objective into layer-wise sensitivity through gradient-based attribution. In this way, EQS incorporates robot-dependent execution information into sensitivity estimation without requiring task-level rollouts. Our contributions are threefold:

- We identify a gap between action-level sensitivity and task-level outcomes, and formulate the execution level as an intermediate sensitivity target for VLA PTQ that incorporates robot-dependent execution information absent from action deviation alone.
- We develop EQS to instantiate this target through horizon-level ordered motion composition and a configuration-dependent operational-space inertial metric, together with data-driven component balancing and gradient-based layer attribution.
- We evaluate EQS across simulation benchmarks and real-robot manipulation, demonstrating improved task success under EQS-guided mixed-precision quantization compared with existing sensitivity-based allocation strategies. Our code is provided in the supplementary materials and will be publicly released upon publication.

2 Related Work

Post-training quantization. PTQ offers a practical approach to model compression without retraining. For large language models, GPTQ uses approximate second-order information to minimize layer-wise reconstruction error during weight quantization (Frantar et al., 2022), AWQ uses activation statistics to identify salient weight channels (Lin et al., 2024), and SmoothQuant applies equivalent scaling to mitigate activation outliers (Xiao et al., 2023). Recent work has adapted PTQ to VLA policies, covering both the backbone and action generation modules. QVLA and ActQuant explore mixed-precision backbone compression at channel and tensor granularities, respectively (Xu et al., 2026b; Akbari et al., 2026). QuantVLA adopts selective quantization for the language backbone and DiT action head (Zhang et al., 2026). HoloQ-VLA combines composite rotation with denoising-step activation scaling to quantize both the language backbone and the action generator at uniform W4A4 precision (Wang et al., 2026). These studies highlight the sensitivity of action generation modules to low-bit quantization and address this fragility through selective precision preservation or specialized quantization procedures. When mixed precision is adopted to preserve control fidelity, reliable sensitivity estimation becomes critical for determining which layers should retain higher precision (Xu et al., 2026a).

Sensitivity estimation for VLA PTQ. To guide precision allocation, VLA-specific methods have explored sensitivity signals that go beyond conventional measures of model-internal numerical fidelity. DyQ-VLA estimates time-varying sensitivity during manipulation through online proxies and uses it to guide dynamic precision switching (Zheng et al., 2026). It focuses on adapting precision over time, whereas our work concentrates on sensitivity estimation during calibration for a fixed mixed-precision allocation. These address complementary dimensions of precision allocation. For precision allocation across model components, Mix-QVLA measures whether quantization preserves the internal task evidence supporting the full-precision policy decision and accounts for its variation over trajectory progress (Ranjan & Savakis, 2026). QVLA and ActQuant characterize sensitivity at the action level, using action deviation or action relevance to identify important components (Xu et al., 2026b; Akbari et al., 2026). However, action deviations alone do not fully characterize the downstream impact of quantization, as illustrated by the matched-pair analyses in Fig. 1(b) and (c). DA-PTQ uses a virtual planar serial chain as a surrogate for drift accumulation and derives drift-aware component weights for layer-wise sensitivity estimation (Xu et al., 2026a). But its component-wise weighting does not explicitly capture how the effects of perturbations combine or cancel in robot motion. Moreover, error propagation alone does not determine how consequential a motion discrepancy is under the robot's current configuration. EQS instead formulates sensitivity at the execution level by evaluating the resulting horizon-level motion discrepancy and its configuration-dependent execution significance.

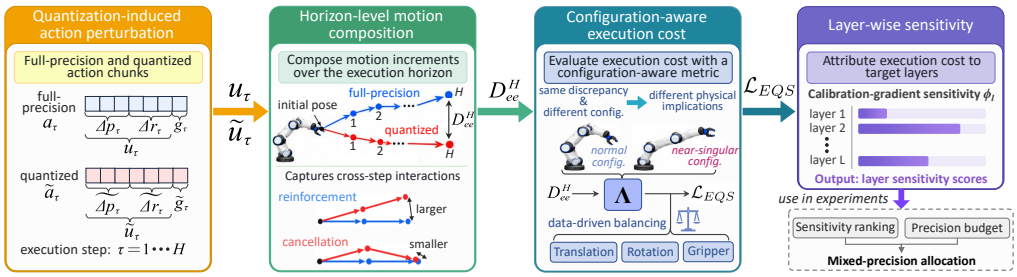


Figure 2: Overview of the EQS flow. First, quantization perturbs action chunks. Next, quantized and full-precision arm commands are composed separately over the execution horizon to obtain the motion discrepancy. Then, execution cost is evaluated through a configuration-aware metric and data-driven component balancing. Finally, calibration gradients yield layer-wise sensitivity scores.

3 Methodology

3.1 From Action Deviation to Motion Discrepancy

Given a candidate layer, quantizing it perturbs the action chunk predicted by the VLA policy. In a receding-horizon execution scheme, only the first H commands in the action chunk are executed before the next policy inference; we refer to H as the execution horizon length. Since commands within this horizon form a temporally ordered sequence, we characterize their joint effect through a horizon-level end-effector motion discrepancy.

For notational simplicity, we consider a single policy inference and denote the corresponding full-precision and quantized action command sequences by $A_{\mathrm{FP}} = \{a_\tau\}_{\tau=1}^{H}$ and $A_Q = \{\tilde{a}_\tau\}_{\tau=1}^{H}$, respectively, where τ indexes the execution step within the horizon. Each action command consists of a six-dimensional Cartesian arm command and a scalar gripper command:

$$a_\tau = (u_\tau, g_\tau), \quad \tilde{a}_\tau = (\tilde{u}_\tau, \tilde{g}_\tau), \tag{1}$$

where $u_\tau, \tilde{u}_\tau \in \mathbb{R}^6$ denote the full-precision and quantized arm commands, and $g_\tau, \tilde{g}_\tau \in \mathbb{R}$ denote their corresponding gripper commands. The arm commands are represented as:

$$u_\tau = (\Delta p_\tau, \Delta r_\tau), \quad \tilde{u}_\tau = (\widetilde{\Delta p}_\tau, \widetilde{\Delta r}_\tau), \tag{2}$$

where $\Delta p_\tau, \widetilde{\Delta p}_\tau \in \mathbb{R}^3$ denote the translational components and $\Delta r_\tau, \widetilde{\Delta r}_\tau \in \mathbb{R}^3$ denote the rotational components of the full-precision and quantized arm commands, respectively. Each arm-command vector specifies translational and rotational increments for the end effector. We interpret this six-dimensional relative pose command as a twist coordinate representation in [translation, rotation] order under the small-increment assumption of the controller-level pose update.

We then convert each command into an incremental rigid-body transformation that can be composed across the execution horizon. This conversion consists of two steps. First, we use the standard hat operator to map the six-dimensional command to a matrix in $\mathfrak{se}(3)$ (Barfoot, 2024):

$$\hat{u}_\tau = \begin{bmatrix} [\Delta r_\tau]_\times & \Delta p_\tau \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathfrak{se}(3), \quad \widehat{\tilde{u}}_\tau = \begin{bmatrix} [\widetilde{\Delta r}_\tau]_\times & \widetilde{\Delta p}_\tau \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathfrak{se}(3), \tag{3}$$

where $[\cdot]_\times$ denotes the standard skew-symmetric cross-product matrix and $\mathfrak{se}(3)$ denotes the matrix Lie algebra of infinitesimal rigid-body motions in three dimensions.

Second, we apply the exponential map to obtain the corresponding incremental rigid-body transformations, which can be composed by matrix multiplication (Murray et al., 1994):

$$T_\tau = \exp(\hat{u}_\tau) \in \mathrm{SE}(3), \quad \tilde{T}_\tau = \exp(\widehat{\tilde{u}}_\tau) \in \mathrm{SE}(3). \tag{4}$$

Here, $\mathrm{SE}(3)$ denotes the Lie group of rigid-body transformations in three dimensions.

These incremental transformations are then composed in execution order to obtain the horizon-level transformations induced by the full-precision and quantized action commands:

$$T_{\mathrm{FP}}^{H} = \exp(\hat{u}_H) \cdots \exp(\hat{u}_1), \quad T_{\mathrm{Q}}^{H} = \exp(\hat{\tilde{u}}_H) \cdots \exp(\hat{\tilde{u}}_1), \tag{5}$$

where increments are left-composed in execution order. To compare these two transformations, we define their relative transformation error as:

$$E_H = T_Q^H (T_{\mathrm{FP}}^H)^{-1}. \tag{6}$$

To obtain a vector representation for subsequent evaluation, we map E_H to the Lie algebra $\mathfrak{se}(3)$:

$$D_{\mathrm{ee}}^H = \big(\log(E_H)\big)^\vee, \tag{7}$$

where $\log(\cdot)$ is the logarithmic map from $\mathrm{SE}(3)$ to $\mathfrak{se}(3)$ and $(\cdot)^\vee$ converts the Lie-algebra matrix representation into a six-dimensional motion vector. In practice, we approximate the log-map term in this discrepancy using a truncated recurrence, as detailed in Appendix D.

Thus, D_{ee}^H measures the horizon-level end-effector motion discrepancy induced by the deviation between the full-precision and quantized action sequences after ordered rigid-body composition. As illustrated by the reinforcement and cancellation examples in Fig. 2, the horizon-level discrepancy reflects how perturbations across execution steps accumulate: their effects may reinforce or partially cancel when evaluated jointly over the horizon (Barfoot & Furgale, 2014).

For comparison, a step-wise formulation evaluates the discrepancy at each execution step independently and then aggregates the resulting scalar costs. Since this aggregation does not preserve the directional and order-dependent relations among perturbations, it cannot, in general, recover the discrepancy produced by the composed motion sequence. The benefit of horizon-level composition is demonstrated by the *StepWise* ablation reported in our experiments (Sec. 4.2).

3.2 Configuration-Aware Execution Cost

The horizon-level motion discrepancy characterizes the quantization-induced geometric difference of the end-effector motions. But a geometric discrepancy alone does not determine its significance for robot execution. As illustrated by the normal and near-singular configurations in Fig. 2, the same discrepancy can have different execution significance depending on the current robot configuration and its direction in operational space (Featherstone, 2008). We therefore introduce a configuration-aware execution cost that maps the motion discrepancy to a physically grounded scalar.

3.2.1 Energy-Grounded Operational-Space Metric

A straightforward scalarization of the motion discrepancy is its squared Euclidean norm: $\mathcal{L}_{\mathrm{Euc}} = \big\|D_{\mathrm{ee}}^H\big\|_2^2$, where D_{ee}^H contains the translational and rotational components. Although intuitive, this formulation combines the two components solely according to their numerical magnitudes, despite their different physical units and meanings. Moreover, it assigns the same cost to a given motion discrepancy regardless of the robot configuration and does not distinguish perturbation directions with different apparent inertia in operational space.

We instead construct an execution cost based on the robot's operational-space inertia. For a robot configuration q, let $M(q)$ denote the joint-space inertia matrix (Liu et al., 2026) and $J(q)$ denote the end-effector Jacobian (Featherstone, 2008). The operational-space inertia matrix (OSIM) is defined as $\Lambda(q) = \big(J(q)M(q)^{-1}J(q)^\top\big)^{-1}$ (Khatib, 1987), which characterizes the configuration-dependent apparent inertia at the end effector. We use a damped formulation for numerical stability:

$$\Lambda_\mu(q) = (J(q)M(q)^{-1}J(q)^\top + D_\mu)^{-1}, \tag{8}$$

where D_μ is a small positive-definite damping matrix. For each calibration chunk, $\Lambda_\mu(q)$ is evaluated at the horizon-start configuration q and kept fixed over the horizon.

To map the translational and rotational motion components to a common physically meaningful scalar, our insight is to formulate the execution cost using an operational-space kinetic-energy form:

$$E = \frac{1}{2} v^\top \Lambda_\mu(q) v, \tag{9}$$

where v denotes a generic end-effector twist velocity containing both translational and rotational components. Through the translational, rotational, and coupling blocks of $\Lambda_\mu(q)$, this quadratic form combines these components into a single scalar in kinetic-energy form while accounting for the robot configuration.

To quantify the execution significance of the motion discrepancy using this quadratic form, we convert D_{ee}^H into an equivalent error twist velocity by dividing it by the fixed execution duration $H\Delta t$: $v_{\text{err}} = D_{\text{ee}}^H/(H\Delta t)$, which represents the horizon-level discrepancy between the full-precision and quantized motions on a per-unit-time basis. Substituting v_{err} for v in the kinetic-energy form gives:

$$E = \frac{1}{2(H\Delta t)^2}(D_{\text{ee}}^H)^\top \Lambda_\mu(q) D_{\text{ee}}^H. \tag{10}$$

Since the factor $1/[2(H\Delta t)^2]$ is constant, we define the end-effector execution cost as:

$$\mathcal{C}_{\text{ee}} = (D_{\text{ee}}^H)^\top \Lambda_\mu(q) D_{\text{ee}}^H. \tag{11}$$

Physically, a quantization-induced motion discrepancy aligned with a direction of larger apparent inertia has greater dynamic significance during execution, because producing or correcting the same end-effector displacement along that direction under comparable motion profiles requires a greater inertial task-space wrench. Our cost reflects this property by evaluating the motion discrepancy through the quadratic form induced by $\Lambda_\mu(q)$, thereby assigning larger penalties to dynamically more demanding directions. Since $\Lambda_\mu(q)$ varies with the robot configuration, $\mathcal{C}_{\text{ee}}$ evaluates the same geometric discrepancy differently across configurations, making it configuration-aware.

3.2.2 Data-Driven Balancing of Heterogeneous Action Components

The cost $\mathcal{C}_{\text{ee}}$ provides a physically grounded metric for the six-dimensional arm-motion discrepancy. However, the complete VLA action representation contains heterogeneous translational, rotational, and gripper components that correspond to distinct coordinate subspaces. Although the cost places translational and rotational motion on a common energy scale, it does not ensure balanced gradient contributions across these subspaces, which have different parameterizations and numerical scales. Existing methods (Xu et al., 2026a) introduce predefined weighting factors based on structural relationships among action dimensions. Such factors provide useful priors, but their relative weights remain fixed and do not adapt to the sensitivity distribution observed during calibration.

We therefore introduce lightweight data-driven balancing coefficients from calibration-time gradients. Let G_t, G_r, and G_g denote the average calibration gradient norms associated with the translational, rotational, and gripper components before balancing. We set:

$$\lambda_r = \frac{G_t}{G_r + \epsilon}, \quad \lambda_g = \frac{G_t}{G_g + \epsilon}, \tag{12}$$

where $\epsilon > 0$ is a small constant for numerical stability. The coefficients λ_r and λ_g compensate for differences in the calibration-time gradient scales of the rotational and gripper components relative to the translational component. Combining the configuration-aware cost and component balancing gives the final execution-level objective:

$$\mathcal{L}_{EQS} = \mathcal{C}_{\text{ee}}^{\text{bal}} + \lambda_g \sum_{\tau=1}^{H} (e_g^\tau)^2, \tag{13}$$

where $e_g^\tau = \tilde{g}_\tau - g_\tau$ denotes the quantization-induced gripper command deviation at horizon step τ. The balanced end-effector execution cost is defined as:

$$\mathcal{C}_{\text{ee}}^{\text{bal}} = (S_{\text{tr}} D_{\text{ee}}^H)^\top \Lambda_\mu(q)(S_{\text{tr}} D_{\text{ee}}^H), \quad S_{\text{tr}} = \text{diag}(1, 1, 1, \sqrt{\lambda_r}, \sqrt{\lambda_r}, \sqrt{\lambda_r}). \tag{14}$$

Here, $\Lambda_\mu(q)$ provides an energy-based metric that maps the translational and rotational components to a common scalar energy scale. However, physical comparability within the arm-motion space does not ensure balanced gradient contributions across the translational, rotational, and gripper subspaces. The coefficients λ_r and λ_g therefore perform inter-subspace balancing. These two mechanisms are complementary: $\Lambda_\mu(q)$ provides a physically grounded and configuration-aware metric for arm-motion discrepancy, whereas the balancing coefficients reduce the tendency of any action subspace to dominate the sensitivity objective merely because of its gradient scale.

3.3 From Execution Cost to Layer-wise Sensitivity

We use the execution-level objective $\mathcal{L}_{\text{EQS}}$ to estimate layer-wise quantization sensitivity through gradient-based attribution.

Inspired by the gradient-based layer attribution used in DA-PTQ (Xu et al., 2026a), we define the sensitivity score of a target layer l with weight matrix W_l as:

$$\phi_l = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|W_l|} \left\| \frac{\partial \mathcal{L}_{\mathrm{EQS}}^{(i)}}{\partial W_l} \right\|_1, \tag{15}$$

where N denotes the number of calibration samples and $|W_l|$ denotes the number of scalar parameters in W_l. This score measures the average magnitude of the execution-level loss gradient with respect to the layer weights over the calibration set. A larger ϕ_l indicates higher estimated sensitivity.

This sensitivity score provides a criterion for precision allocation: layers with larger ϕ_l can be preserved at higher precision, while layers with smaller ϕ_l are less prioritized for high-precision retention. We use this ranking to construct the mixed-precision configurations in our experiments.

4 Experiments

4.1 Experimental Setup

Models and benchmarks. We evaluate EQS on two VLA models: $\pi_{0.5}$ (Intelligence et al., 2025) and CogACT (Li et al., 2024a). For $\pi_{0.5}$, we follow prior studies (Wang et al., 2026; Zhang et al., 2026) and evaluate on LIBERO (Liu et al., 2023). For CogACT, we use SimplerEnv (Li et al., 2024b), following DA-PTQ (Xu et al., 2026a).

Quantization setting. Each sensitivity criterion produces layer-wise sensitivity scores that guide mixed-precision quantization of the VLA action generation modules. Following recent studies (Zheng et al., 2026; Ranjan & Savakis, 2026; Zhang et al., 2026), we use W4A4 for low-precision layers. For each model, the backbone uses a DA-PTQ-based quantization configuration (Xu et al., 2026a) that is identical across all compared methods. This controlled setup allows us to assess how the choice of action-module sensitivity criterion affects task-level performance.

Following prior work (Wang et al., 2026), we consider attention and MLP layers in the action generation module as quantization candidates. For each sensitivity criterion, we rank attention and MLP layers separately according to their sensitivity scores, retain the top-ranked layers in BF16 at fixed ratios, and quantize the remaining layers to W4A4, following a retention strategy adapted from DA-PTQ (Xu et al., 2026a). The calibration set, candidate layers, MLP/attention retention ratios, and allocation rule are identical across all criteria. Details of calibration-set construction and calibration settings are provided in Appendices A and B. Our code is provided in the supplementary materials.

Baselines. We compare EQS with methods from prior VLA quantization and PTQ studies, including action-level and drift-aware sensitivity criteria, as well as layer-wise sensitivity proxies. Detailed formulations are provided in Appendix E.

Action-level baselines include *Action-MSE* and a layer-wise adaptation of *QVLA* (Xu et al., 2026b). The former uses the unweighted action-chunk deviation as a differentiable objective for gradient-based layer attribution, while the latter measures the final action-chunk deviation when each candidate layer is quantized individually. The drift-aware sensitivity criterion from *DA-PTQ* (Xu et al., 2026a) is also included, which accounts for the kinematic drift induced by action perturbations.

We further consider four layer-wise sensitivity proxies motivated by prior PTQ studies: *Layer-nMSE*, *W-Outlier*, *A-Outlier*, and *StepDrift*. *Layer-nMSE* measures normalized layer-output reconstruction error, drawing on reconstruction-based PTQ (Frantar et al., 2022). *W-Outlier* and *A-Outlier* characterize extreme values in weight and activation distributions as such outliers can induce large quantization errors (Dettmers et al., 2024; Xiao et al., 2023). *StepDrift* measures activation variation across denoising steps, motivated by timestep-dependent distribution changes reported in diffusion-model quantization (Chen et al., 2024; Li et al., 2023).

4.2 Experimental Results

Simulation Results. Table 1 reports the task success rates of EQS and the baselines under two MLP/attention BF16 retention settings. With the retention ratios held fixed, these results compare the downstream performance of the mixed-precision allocations produced by different sensitivity criteria. Results for additional retention settings are provided in Appendix F.

Table 1: Task success rates (%) comparison under representative BF16 retention settings.

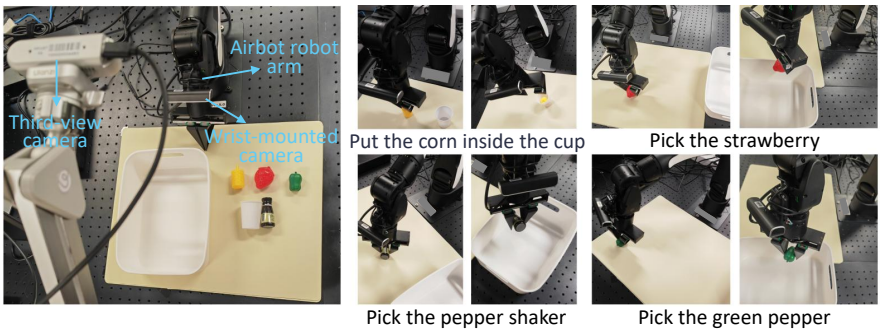
Model	Setting	Method	LIBERO					Δ
			Object	Spatial	Goal	Long	Avg.	
$\pi_{0.5}$	MLP retained in BF16: 10%; attention retained in BF16: 30%	**EQS (Ours)**	**95.17**	**98.13**	**93.75**	**86.25**	**93.33**	0.00
		DA-PTQ	91.78	97.68	92.54	71.88	88.47	-4.86
		Action-MSE	92.09	97.09	92.17	71.74	88.27	-5.06
		QVLA	**95.17**	97.98	89.88	83.15	91.55	-1.78
		Layer-nMSE	82.83	96.61	84.87	75.09	84.85	-8.48
		W-Outlier	93.42	96.80	92.52	85.40	92.04	-1.29
		A-Outlier	75.04	91.74	62.87	51.09	70.19	-23.14
		StepDrift	92.35	97.04	92.65	83.22	91.32	-2.01
		StepWise	93.91	98.07	92.83	85.37	92.55	-0.78
		EQS w/o config-aware	93.48	97.72	93.37	85.54	92.53	-0.80
		EQS w/o balancing	93.51	96.81	93.26	86.07	92.41	-0.92
$\pi_{0.5}$	MLP retained in BF16: 60%; attention retained in BF16: 50%	**EQS (Ours)**	**97.84**	**98.38**	**96.72**	**90.00**	**95.74**	0.00
		DA-PTQ	96.66	98.10	96.56	86.44	94.44	-1.30
		Action-MSE	97.12	98.13	96.70	87.55	94.88	-0.86
		QVLA	97.17	98.32	95.49	87.23	94.55	-1.19
		Layer-nMSE	96.20	98.37	96.14	87.28	94.50	-1.24
		W-Outlier	**97.84**	97.56	95.79	89.99	95.30	-0.44
		A-Outlier	97.61	98.10	96.30	88.75	95.19	-0.55
		StepDrift	95.26	97.87	94.87	86.22	93.56	-2.18
		StepWise	97.15	98.16	96.28	87.73	94.83	-0.91
		EQS w/o config-aware	97.77	98.26	96.47	88.80	95.33	-0.41
		EQS w/o balancing	97.01	97.96	96.52	86.58	94.52	-1.22

Model	Setting	Method	SimplerEnv					Δ
			Carrot	Eggplant	Spoon	Stack Cube	Avg.	
CogACT	MLP retained in BF16: 10%; attention retained in BF16: 30%	**EQS (Ours)**	**66.67**	**62.50**	**82.61**	**51.72**	**65.88**	0.00
		DA-PTQ	54.78	61.36	76.56	46.09	59.70	-6.18
		Action-MSE	54.35	**62.50**	74.64	44.20	58.92	-6.96
		QVLA	56.52	58.54	73.04	**51.72**	59.96	-5.92
		Layer-nMSE	**66.67**	52.17	71.88	44.90	58.91	-6.97
		W-Outlier	56.92	31.52	72.41	48.28	52.28	-13.60
		A-Outlier	62.86	43.48	74.07	46.15	56.64	-9.24
		StepDrift	59.57	42.61	74.42	51.19	56.95	-8.93
		StepWise	56.52	56.52	**82.61**	39.13	58.70	-7.18
		EQS w/o config-aware	56.52	56.52	**82.61**	39.13	58.70	-7.18
		EQS w/o balancing	54.35	**62.50**	74.64	44.20	58.92	-6.96
CogACT	MLP retained in BF16: 60%; attention retained in BF16: 50%	**EQS (Ours)**	**63.41**	**70.65**	**80.43**	**49.02**	**65.88**	0.00
		DA-PTQ	55.43	64.95	76.32	40.48	59.30	-6.58
		Action-MSE	56.52	68.48	73.91	40.91	59.96	-5.92
		QVLA	53.14	59.42	73.91	41.67	57.04	-8.84
		Layer-nMSE	43.48	65.22	73.91	40.00	55.65	-10.23
		W-Outlier	62.28	42.75	76.79	**49.02**	57.71	-8.17
		A-Outlier	54.35	57.61	71.93	42.86	56.69	-9.19
		StepDrift	**63.41**	58.70	69.70	41.67	58.37	-7.51
		StepWise	50.00	65.22	76.09	41.30	58.15	-7.73
		EQS w/o config-aware	43.48	63.04	76.09	34.78	54.35	-11.53
		EQS w/o balancing	56.52	68.48	73.91	40.91	59.96	-5.92

On $\pi_{0.5}$ with LIBERO, EQS achieves the highest average success rate under both retention settings. With 10% of MLP and 30% of attention layers retained in BF16, EQS reaches 93.33% average success, outperforming the baselines by 1.29∼23.14 percentage points. Under the 60% MLP/50% attention setting, EQS achieves 95.74%, maintaining a 0.44∼2.18-point advantage. The improvements extend to CogACT on SimplerEnv, where EQS reaches 65.88% average success under both retention settings, outperforming the baselines by 5.92∼13.60 and 5.92∼10.23 percentage points. EQS achieves the highest or tied-highest success rate in all task-level comparisons in Table 1, showing consistent gains across the evaluated VLA models and tasks.

Combination with quantization procedures. EQS guides layer-wise precision allocation without prescribing how layers are quantized at their assigned precision. It is therefore complementary to quantization procedures involving transformations such as rotation and scaling. We examine this compatibility by combining EQS-guided precision allocation with the quantization procedure proposed in HoloQ-VLA (Wang et al., 2026). The results is shown in Appendix F.

Ablation Study. To assess the contributions of horizon-level composition, configuration-aware execution cost, and component balancing, we evaluate three ablations of EQS in Table 1. *StepWise* re-



(a) Physical Robot Setup (b) Real-world Task Examples

Figure 3: Physical robot setup and real-world task demonstrations. The system consists of an Airbot robot arm, wrist-mounted camera, and third-person view camera.

Table 2: Real-world success-rate comparison on the Airbot robot arm using $\pi_{0.5}$.

Method	Corn into cup	Strawberry	Pepper shaker	Green pepper	Avg. SR
Full precision	**21/40**	**36/40**	**27/40**	27/40	**69.38%**
DA-PTQ	16/40	32/40	20/40	**29/40**	60.63%
QVLA	19/40	33/40	22/40	24/40	61.25%
EQS (Ours)	**21/40**	34/40	23/40	**29/40**	66.88%

places horizon-level composition with independent evaluations of per-step motion discrepancies and sums their execution costs. *EQS w/o config-aware* removes configuration-aware inertial weighting by replacing $\Lambda_\mu(q)$ with I_6, while retaining horizon-level composition and component balancing. *EQS w/o balancing* disables the data-driven balancing of heterogeneous action components.

Comparison with *StepWise* highlights the benefit of horizon-level composition. By retaining cross-step interactions, EQS accounts for how perturbations may reinforce or partially cancel during sequential composition. Consequently, *StepWise* can perform comparably when these interactions have negligible impact, but can underperform when their effects become consequential. For example, the two criteria achieve identical success on SimplerEnv-Spoon and differ by only 0.06 percentage points on LIBERO-Spatial at 10% MLP/30% attention. In contrast, EQS exceeds *StepWise* by 2.27 percentage points on LIBERO-Long at 60% MLP/50% attention and 12.59 points on SimplerEnv-Stack-Cube at 10% MLP/30% attention.

Both *EQS w/o config-aware* and *EQS w/o balancing* yield lower average success rates than EQS across all four settings in Table 1. These results support the effectiveness of configuration-aware execution evaluation and heterogeneous action-component balancing, respectively.

Real-World Robot Deployment. We evaluate EQS-guided mixed-precision quantization of $\pi_{0.5}$ on an Airbot robot arm across four tasks (Fig. 3). Full precision serves as an unquantized reference, while DA-PTQ and our layer-wise adaptation of QVLA serve as kinematic-drift and action-deviation sensitivity baselines, respectively. With 30% of both MLP and attention layers retained in BF16 and identical quantization settings across the quantized methods, EQS achieves a 66.88% success rate (107/160 trials), compared with 60.63% (97/160) for *DA-PTQ*, 61.25% (98/160) for *QVLA*, and 69.38% (111/160) for full precision (Table 2). The evaluation protocol is detailed in Appendix H.

5 Conclusion

We formulate the execution level as a sensitivity target for VLA PTQ and introduce Execution-Level Quantization Sensitivity (**EQS**). EQS maps quantization-induced action perturbations to horizon-level motion discrepancy through ordered composition, evaluates its execution significance with a configuration-aware cost and data-driven component balancing, and attributes the objective to layers via calibration gradients. Experimental results show that EQS-guided allocation exceeds the best-performing baseline by 0.44∼8.06 percentage points. Looking ahead, we will explore mixed-precision allocation that adapts EQS-based layer prioritization to the quantization errors produced by different quantization procedures.

AI USE STATEMENT

In this work, we used generative AI tools only to assist with code debugging and manuscript revision. All AI-assisted debugging suggestions were manually reviewed and tested by the authors, and all identified language and grammar issues were independently checked before any changes were made. Generative AI tools were not used to generate experimental data or to make final decisions regarding the methodology, experiments, analysis, or scientific claims. We take responsibility for the final content of this work, including any text, code, claims, or artifacts produced with the aid of generative AI.

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. *arXiv preprint arXiv:2605.24011*, 2026.

Timothy D. Barfoot. *State Estimation for Robotics: Second Edition*. Cambridge University Press, 2024.

Timothy D. Barfoot and Paul T. Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30(3):679–693, 2014.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jingyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-dit: Accurate post-training quantization for diffusion transformers. *arXiv preprint arXiv:2406.17343*, 2024.

Tim Dettmers, Ruslan A. Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *The Twelfth International Conference on Learning Representations*, 2024.

Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 2008.

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.

Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Springer, 2015.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025.

O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag 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. *arXiv preprint arXiv:2406.09246*, 2024.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024a.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024b.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *MLSys*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023.

Xingyu Liu, Jiawei Liang, Yipu Zhang, Linfeng Du, Chaofang Ma, Hui Yu, Jiang Xu, and Wei Zhang. Draco: A hardware-efficient robot rigid body dynamics accelerator with precision-aware quantization framework. In *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2026.

Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, 2021.

Richard M. Murray, Zexiang Li, and Shankar Sastry. *A mathematical introduction to robotic manipulation.* CRC Press, 1994.

Navin Ranjan and Andreas Savakis. Mix-QVLA: Task-Evidence-Aware Mixed-Precision Quantization of Vision-Language-Action Models. *arXiv preprint arXiv:2606.19565*, 2026.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

Xinyu Wang, Mingze Li, Sicheng Lyu, Dongxiu Liu, Kaicheng Yang, Ziyu Zhao, Yufei Cui, Xiao-Wen Chang, and Peng Lu. Holoq-vla: Uniform w4a4 quantization of vision-language-action models. *arXiv preprint arXiv:2605.28803*, 2026.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Siyuan Xu, Tianshi Wang, Fengling Li, Lei Zhu, and Heng Tao Shen. DA-PTQ: Drift-Aware Post-Training Quantization for Efficient Vision-Language-Action Models. *arXiv preprint arXiv:2604.11572*, 2026a.

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*, 2026b.

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. *arXiv preprint arXiv:2602.20309*, 2026.

Zihao Zheng, Hangyu Cao, Sicheng Tian, Jiayu Chen, Maoliang Li, Xinhao Sun, Hailong Zou, Zhaobo Zhang, Xuanzhe Liu, Donggang Cao, Hong Mei, and Xiang Chen. DyQ-VLA: Temporal-Dynamic-Aware Quantization for Embodied Vision-Language-Action Models. *arXiv preprint arXiv:2603.07904*, 2026.

A Calibration Set Construction

All sensitivity metrics are evaluated using a shared calibration set to ensure a fair comparison. The calibration set is only used for collecting sensitivity statistics and does not involve any parameter update.

$\pi_{0.5}$. We construct the calibration set from demonstration episodes across four LIBERO suites: libero_spatial, libero_object, libero_goal, and libero_10. We first sample 2000 candidate action chunks, where each candidate is extracted from a randomly selected timestep in a demonstration episode. A semantic-kinematic coverage objective is then applied to select 512 representative calibration chunks.

Each selected sample corresponds to an action chunk starting from a recorded timestep t_{start}. The calibration set contains 512 action chunks with horizon $H = 8$, covering 4096 primitive action steps in total. The selected samples span 25 semantic instruction groups and 39 unique demonstration task files. During sensitivity computation, all observations, robot states, and action slices are reconstructed using the same recorded t_{start} to maintain temporal alignment.

CogACT. For CogACT, we construct the calibration set from BridgeData V2 (Walke et al., 2023) demonstrations following the calibration protocol used in prior CogACT quantization studies (Xu et al., 2026a). The calibration set contains 512 action chunks selected through semantic grouping and kinematic coverage. Each sample uses an action horizon of $H = 8$ and is associated with the corresponding robot state for execution-aware sensitivity computation.

B Calibration Settings

For $\pi_{0.5}$ on LIBERO, the candidate pool contains 2,000 demonstration episodes, with 500 from each of the Spatial, Object, Goal, and LIBERO-10 suites. Using a fixed random seed, we select one valid starting timestep per episode and form an $H = 8$ action chunk. Candidates are grouped by the first four words of their lowercased language instructions. We allocate 512 samples approximately uniformly across the resulting 25 instruction-prefix groups. Within each group, we apply MiniBatch K-means to min–max normalized seven-dimensional joint configurations and three-dimensional end-effector positions, sample across the resulting clusters, and fill any remaining quota from unselected candidates in that group. Action-magnitude statistics are recorded for coverage analysis but are not used to select samples.

For CogACT, we use 512 BridgeData V2 calibration chunks with the same execution horizon, as described in Appendix A. In the $\pi_{0.5}$ QVLA adaptation, each full-precision reference and its layer-wise quantization interventions share the same sample-specific diffusion noise and are compared using their decoded eight-step action chunks.

C Calibration Overhead Details

Table 3 reports the calibration-time profiling of different sensitivity criteria using 512 calibration samples and execution horizon $H = 8$, deployed on NVIDIA RTX 4090D GPUs. On $\pi_{0.5}$, EQS requires 253.3s, which is lower than DA-PTQ at 329.7s and slightly higher than StepWise at 245.2s, since EQS has extra computation workload for horizon-level composition. QVLA requires 11290.2s

Table 3: Calibration-time profiling of different sensitivity criteria. All calibration-based criteria use 512 calibration samples with execution horizon $H = 8$. Peak memory denotes the maximum GPU memory observed during sensitivity scoring.

Model	Method	Layers	Time	Peak Mem.
$\pi_{0.5}$	EQS	126	253.3s	15.5GB
	StepWise	126	245.2s	15.5GB
	EQS w/o config-aware	126	232.2s	15.5GB
	EQS w/o balancing	126	240.3s	15.5GB
	Action-MSE	126	276.2s	15.5GB
	DA-PTQ	126	329.7s	15.6GB
	QVLA	126	11290.2s	16.5GB
	Layer-nMSE	126	723.8s	8.0GB
	W-Outlier	126	109.5s	7.4GB
	A-Outlier	126	637.3s	7.9GB
	StepDrift	126	636.6s	7.9GB
CogACT	EQS	48	481.9s	29.8GB
	StepWise	48	458.2s	29.8GB
	EQS w/o config-aware	48	454.4s	29.9GB
	EQS w/o balancing	48	488.8s	29.8GB
	Action-MSE	48	459.9s	29.8GB
	DA-PTQ	48	442.2s	29.9GB
	QVLA	48	3388.4s	29.9GB
	Layer-nMSE	48	462.2s	29.8GB
	W-Outlier	48	169.2s	25.1GB
	A-Outlier	48	425.3s	29.8GB
	StepDrift	48	412.6s	29.8GB

Table 4: One-time OSIM cache construction overhead.

Model	Entries	Valid Queries	Time
$\pi_{0.5}$	512	512/512	0.935s
CogACT	512	512/512	0.180s

because it executes the complete downstream action generator for each layer-wise quantization intervention. Among the four layer-wise sensitivity proxies, Layer-nMSE, W-Outlier, A-Outlier, and StepDrift require 723.8s, 109.5s, 637.3s, and 636.6s, respectively. W-Outlier is substantially faster because its score is computed directly from weight statistics without calibration forward passes.

On CogACT, EQS requires 481.9s, compared with 458.2s for StepWise and 442.2s for DA-PTQ. QVLA, Layer-nMSE, A-Outlier, and StepDrift require 3388.4s, 462.2s, 425.3s, and 412.6s, respectively, while W-Outlier requires 169.2s. Overall, EQS incurs calibration cost comparable to other execution-aware criteria while avoiding any additional computation during deployment.

EQS additionally uses an OSIM cache to evaluate execution-level discrepancy during calibration. The cache is constructed once from the calibration robot configurations and reused throughout sensitivity scoring. As shown in Table 4, constructing all 512 entries takes only 0.935s for $\pi_{0.5}$ and 0.180s for CogACT, making this preprocessing cost negligible relative to sensitivity scoring.

D Computation of the Horizon-Level Motion Discrepancy

Sec. 3.1 defines the exact product-log discrepancy $D_{\text{ee}}^{H} = \big(\log(E_H)\big)^{\vee}$ under the $\mathrm{SE}(3)$ composition model through the relative rigid-body transformation between the full-precision and quantized action sequences:

$$T_{\text{FP}}^{H} = \exp(\hat{u}_H)\cdots\exp(\hat{u}_1), \quad T_{\text{Q}}^{H} = \exp(\hat{\tilde{u}}_H)\cdots\exp(\hat{\tilde{u}}_1), \quad E_H = T_Q^H (T_{\text{FP}}^H)^{-1},$$

where the product follows the execution order over the horizon H.

Directly evaluating this exact product-log form is possible, but it is computationally expensive as a repeated scoring primitive for PTQ sensitivity estimation. The sensitivity score must be evaluated across all candidate quantized layers and calibration chunks; each evaluation would require composing an $\mathrm{SE}(3)$ matrix product over the execution horizon and computing the matrix logarithm of the resulting relative transformation. Furthermore, when used in gradient-based sensitivity scoring, it requires differentiating through long chains of matrix exponentials and a matrix logarithm.

To obtain an efficient, closed-form, and differentiable formulation, we employ the Baker–Campbell–Hausdorff (BCH) expansion (Hall, 2015; Barfoot & Furgale, 2014). Because VLA action chunks are represented as small incremental commands and quantization perturbations remain small during calibration, the low-order terms of this expansion capture the dominant drift behavior. We retain terms up to second order, yielding a closed-form recursion in twist coordinates. This BCH_2 recursion preserves the first-order residual term and all second-order Lie-bracket correction terms induced by rigid-body motion composition, while dropping third- and higher-order nested brackets.

D.1 Recursive Relative Transformation Formulation

For any execution prefix step $\tau \in \{1, \dots, H\}$, let T^{τ}_{FP} and T^{τ}_{Q} denote the accumulated transformations under the full-precision and quantized action sequences, respectively:

$$T^{\tau}_{\mathrm{FP}} = \exp(\hat{u}_{\tau}) \cdots \exp(\hat{u}_1), \quad T^{\tau}_{\mathrm{Q}} = \exp(\hat{\tilde{u}}_{\tau}) \cdots \exp(\hat{\tilde{u}}_1), \tag{16}$$

with initial conditions $T^{0}_{\mathrm{FP}} = T^{0}_{Q} = I$. We express u_{τ} and $\tilde{u}_{\tau}$ as twist coordinates in a common fixed reference frame and compose the corresponding incremental motions from the left, yielding:

$$T^{\tau}_{\mathrm{FP}} = \exp(\hat{u}_{\tau}) T^{\tau-1}_{\mathrm{FP}}, \qquad T^{\tau}_{Q} = \exp(\widehat{\tilde{u}}_{\tau}) T^{\tau-1}_{Q}. \tag{17}$$

We define the relative rigid-body transformation at prefix step τ as $E_{\tau} = T^{\tau}_{Q}(T^{\tau}_{\mathrm{FP}})^{-1}$. Its step-to-step propagation is given by:

$$\begin{aligned} E_{\tau} &= \exp(\widehat{\tilde{u}}_{\tau}) T^{\tau-1}_{Q} \left(T^{\tau-1}_{\mathrm{FP}}\right)^{-1} \exp(-\hat{u}_{\tau}) \\ &= \exp(\widehat{\tilde{u}}_{\tau}) E_{\tau-1} \exp(-\hat{u}_{\tau}), \end{aligned} \tag{18}$$

with $E_0 = I$. Defining the quantization perturbation on the arm command as $\delta u_{\tau} = \tilde{u}_{\tau} - u_{\tau}$, we have $\tilde{u}_{\tau} = u_{\tau} + \delta u_{\tau}$, which yields:

$$E_{\tau} = \exp(\widehat{u_{\tau} + \delta u_{\tau}}) E_{\tau-1} \exp(-\hat{u}_{\tau}). \tag{19}$$

Let $\xi_{\tau} = \left(\log(E_{\tau})\right)^{\vee} \in \mathbb{R}^6$ denote the twist coordinates of the relative transformation at prefix step τ, where $\xi_0 = 0$ and $\xi_H = D^{H}_{\mathrm{ee}}$. The exact vector-form update in logarithmic coordinates is:

$$\xi_{\tau} = \log\left(\exp(\widehat{u_{\tau} + \delta u_{\tau}}) \exp(\hat{\xi}_{\tau-1}) \exp(-\hat{u}_{\tau})\right)^{\vee}. \tag{20}$$

D.2 Second-Order BCH Expansion

For two twist vectors $a, b \in \mathbb{R}^6$, the BCH operator converts the logarithm of the product of two exponentials into a Lie-algebra series:

$$\log\left(\exp(\hat{a}) \exp(\hat{b})\right)^{\vee} = \mathrm{BCH}(a, b) = a + b + \frac{1}{2}[a, b] + \mathcal{O}_3, \tag{21}$$

where $[\cdot, \cdot]$ denotes the Lie bracket on $\mathfrak{se}(3)$ in vector coordinates, defined by $[a, b] = (\hat{a}\hat{b} - \hat{b}\hat{a})^{\vee}$, and $\mathcal{O}_3$ denotes nested Lie brackets of order three and higher.

To expand ξ_{τ}, define the temporary variables:

$$A = u_{\tau}, \quad E = \delta u_{\tau}, \quad \delta = \xi_{\tau-1}. \tag{22}$$

The exact update can be written as the composition of two successive BCH operations. First, applying BCH to the first two exponentials yields:

$$M = \mathrm{BCH}(A + E, \delta) = A + E + \delta + \frac{1}{2}[A + E, \delta] + \mathcal{O}_3. \tag{23}$$

Expanding the bracket by bilinearity gives:

$$M = A + E + \delta + \frac{1}{2}[A, \delta] + \frac{1}{2}[E, \delta] + \mathcal{O}_3. \tag{24}$$

Second, composing M with the rightmost term $-A$ gives:

$$\xi_\tau = \mathrm{BCH}(M, -A) = M - A + \frac{1}{2}[M, -A] + \mathcal{O}_3. \tag{25}$$

Inside the bracket $[M, -A]$, only the first-order component of M, denoted $M_1 = A + E + \delta$, contributes to terms up to second order:

$$\frac{1}{2}[M, -A] = \frac{1}{2}[A + E + \delta, -A] + \mathcal{O}_3 = \frac{1}{2}[A, -A] + \frac{1}{2}[E, -A] + \frac{1}{2}[\delta, -A] + \mathcal{O}_3. \tag{26}$$

By the alternating property $[A, -A] = 0$ and the antisymmetry of the Lie bracket ($[X, -Y] = [Y, X]$), we obtain:

$$[E, -A] = [A, E], \qquad [\delta, -A] = [A, \delta]. \tag{27}$$

Thus,

$$\frac{1}{2}[M, -A] = \frac{1}{2}[A, E] + \frac{1}{2}[A, \delta] + \mathcal{O}_3. \tag{28}$$

Substituting M and $\frac{1}{2}[M, -A]$ into the expression for ξ_τ yields:

$$\begin{aligned} \xi_\tau &= \left(A + E + \delta + \frac{1}{2}[A, \delta] + \frac{1}{2}[E, \delta] \right) - A + \frac{1}{2}[A, E] + \frac{1}{2}[A, \delta] + \mathcal{O}_3 \\ &= \delta + E + [A, \delta] + \frac{1}{2}[A, E] + \frac{1}{2}[E, \delta] + \mathcal{O}_3. \end{aligned} \tag{29}$$

Returning to the original physical variables, the second-order recursion (BCH_2) used for sensitivity scoring is:

$$\xi_\tau^{(2)} = \xi_{\tau-1}^{(2)} + \delta u_\tau + [u_\tau, \xi_{\tau-1}^{(2)}] + \frac{1}{2}[u_\tau, \delta u_\tau] + \frac{1}{2}[\delta u_\tau, \xi_{\tau-1}^{(2)}], \tag{30}$$

initialized with $\xi_0^{(2)} = 0$. We denote the resulting BCH_2 approximation of the horizon-level motion discrepancy as:

$$\widetilde{D}_{\mathrm{ee}}^{H} = \xi_H^{(2)}. \tag{31}$$

Here, the superscript H denotes the action-chunk execution horizon, and the tilde indicates the second-order truncation of the exact discrepancy D_{ee}^{H}. In our implementation, $\widetilde{D}_{\mathrm{ee}}^{H}$ is directly substituted into the execution cost computation in Section 3.2.

D.3 Lie Bracket Structure

For generic twist vectors $a, b \in \mathbb{R}^6$ partitioned into translational and rotational components following Eq. 2:

$$a = (p_a, r_a), \quad b = (p_b, r_b), \qquad p_a, p_b, r_a, r_b \in \mathbb{R}^3, \tag{32}$$

the $\mathfrak{se}(3)$ Lie bracket evaluates to

$$[a, b] = \begin{bmatrix} r_a \times p_b - r_b \times p_a \\ r_a \times r_b \end{bmatrix}. \tag{33}$$

The bracket terms in Eq. 30 capture the leading-order physical interactions induced by non-commutative rigid-body motion composition. They account for how the nominal motion, the current quantization residual, and the previously accumulated drift interact across the action chunk.

D.4 Validation of the Second-order BCH Approximation

To examine whether the BCH_2 approximation preserves the sensitivity decisions induced by the exact product-log formulation, we recompute layer sensitivities on the full 512-sample $\pi_{0.5}$ calibration set using the exact differentiable $\mathrm{SE}(3)$ product-log discrepancy. The resulting rankings are highly consistent with BCH_2, with Spearman correlations of 0.9962, 0.9995, and 0.9970 over all, MLP, and attention layers, respectively, and corresponding Kendall correlations of 0.9606, 0.9930, and 0.9664. The resulting precision allocations also differ only at a small number of boundary selections across the evaluated retention settings. In a formula-level GPU benchmark, the exact product-log formulation requires **3.24**$\times$ the computation time of BCH_2. These results indicate that BCH_2 closely preserves the sensitivity ordering of the exact formulation while providing a more efficient closed-form recursion for calibration.

E Sensitivity Baseline Formulations

We describe the sensitivity metrics used for comparison. All metrics produce a layer-wise sensitivity score, where a larger value indicates that the corresponding layer is more sensitive to quantization.

For *Action-MSE*, *DA-PTQ*, and the EQS ablations, which are defined through differentiable calibration objectives, we use the same gradient-based attribution as in Sec. 3.3. Given a criterion-specific calibration objective $\mathcal{L}$, the sensitivity score of layer l is computed as

$$\phi_l = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|W_l|}\left\|\frac{\partial \mathcal{L}^{(i)}}{\partial W_l}\right\|_1. \tag{34}$$

Here, $\mathcal{L}^{(i)}$ denotes the criterion-specific objective for the i-th calibration sample, W_l is the weight matrix of layer l, $|W_l|$ is its number of parameters, and N is the number of calibration samples. These criteria share the same attribution rule and differ only in the definition of $\mathcal{L}$.

The remaining layer-wise sensitivity proxies directly compute layer scores from local reconstruction errors or weight and activation statistics.

Action-MSE. *Action-MSE* uses the unweighted deviation between full-precision and quantized action outputs as the sensitivity objective. For a calibration sample, the objective is defined as:

$$\mathcal{L}_{\mathrm{MSE}} = \|A_{\mathrm{FP}} - A_Q\|_2^2, \tag{35}$$

where A_{FP} and A_Q denote the full-precision and quantized action chunks, respectively. The layer sensitivity score is then obtained using the gradient attribution formulation above.

DA-PTQ. DA-PTQ models action-error propagation using a structural Jacobian derived from a virtual planar serial chain (Xu et al., 2026a). Let J^+ denote its damped pseudoinverse. The drift sensitivity of action dimension j is:

$$s_j = \mathbb{E}_{a\sim\mathcal{D}}\left[\sum_{c\in\{x,y,\theta\}} w_c |J^+_{j,c}|\right], \qquad \hat{s}_j = \frac{s_j}{\frac{1}{7}\sum_{i=1}^{7} s_i}. \tag{36}$$

These weights are incorporated into the diffusion calibration objective:

$$\mathcal{L}_{\mathrm{DA}} = \mathbb{E}_{x,\epsilon,t}\left[\sum_{j=1}^{7}\hat{s}_j\left(\hat{\epsilon}_j(x_t,t,z)-\epsilon_j\right)^2\right]. \tag{37}$$

Layer sensitivity is then obtained from the average gradient magnitude of $\mathcal{L}_{\mathrm{DA}}$, following the common attribution formulation above.

QVLA. We adapt the channel-wise action-centric sensitivity criterion of QVLA (Xu et al., 2026b) to our layer-wise setting by measuring the final action deviation caused by quantizing each candidate layer. For calibration sample i, let $A_i^{\mathrm{FP}} \in \mathbb{R}^{H\times d_a}$ denote the decoded reference action chunk and $A_{i,l}^{Q} \in \mathbb{R}^{H\times d_a}$ denote the decoded action chunk obtained when only candidate layer l is quantized to W4A4. We define:

$$\phi_l^{\mathrm{QVLA}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Hd_a}\left\|A_{i,l}^{Q} - A_i^{\mathrm{FP}}\right\|_F^2, \tag{38}$$

Table 5: Task success rates (%) for the remaining two representative $\pi_{0.5}$ mixed-precision configurations on LIBERO. The reporting protocol is identical to Table 1.

Model	Setting	Method	LIBERO					Δ
			Object	Spatial	Goal	Long	Avg.	
$\pi_{0.5}$	MLP retained in BF16: 10%; attention retained in BF16: 50%	**EQS (Ours)**	**94.50**	**98.10**	**96.09**	**87.83**	**94.13**	0.00
		DA-PTQ	91.96	97.07	92.28	71.30	88.15	-5.98
		Action-MSE	91.09	97.61	92.07	75.33	89.03	-5.10
		QVLA	94.42	97.83	90.65	84.20	91.78	-2.35
		Layer-nMSE	81.63	96.63	86.30	76.30	85.22	-8.91
		W-Outlier	94.02	96.63	93.15	85.22	92.26	-1.87
		A-Outlier	70.00	92.39	63.37	50.11	68.97	-25.16
		StepDrift	92.28	95.98	92.28	82.50	90.76	-3.37
		StepWise	**94.50**	97.98	93.88	87.76	93.53	-0.60
		EQS w/o config-aware	**94.50**	97.98	94.01	87.76	93.56	-0.57
		EQS w/o balancing	94.08	96.39	93.51	86.74	92.68	-1.45
	MLP retained in BF16: 30%; attention retained in BF16: 50%	**EQS (Ours)**	**97.83**	**97.83**	**96.30**	**88.91**	**95.22**	0.00
		DA-PTQ	95.65	96.96	94.78	87.17	93.64	-1.58
		Action-MSE	97.39	96.96	95.22	85.65	93.81	-1.41
		QVLA	96.09	**97.83**	94.57	86.41	93.73	-1.49
		Layer-nMSE	79.13	96.00	88.43	78.04	85.40	-9.82
		W-Outlier	95.65	97.00	92.96	85.22	92.71	-2.51
		A-Outlier	60.13	91.22	59.35	34.91	61.40	-33.82
		StepDrift	94.09	96.83	93.13	84.39	92.11	-3.11
		StepWise	97.01	97.61	95.84	87.69	94.54	-0.68
		EQS w/o config-aware	97.63	97.71	96.26	87.75	94.84	-0.38
		EQS w/o balancing	94.42	97.63	94.78	82.92	92.44	-2.78

where N is the number of calibration samples, H is the action horizon, and d_a is the action dimension. We use $N = 512$, $H = 8$, and $d_a = 7$, and directly rank layers by ϕ_l^{QVLA} without gradient attribution.

Layer-nMSE. *Layer-nMSE* directly measures the local output distortion of each candidate layer:

$$\phi_l^{\mathrm{nMSE}} = \mathbb{E}_{x\sim\mathcal{D}_{\mathrm{calib}}}\left[\frac{\left\|Y_l^Q(x) - Y_l^{FP}(x)\right\|_F^2}{\left\|Y_l^{FP}(x)\right\|_F^2 + \epsilon}\right]. \tag{39}$$

Here, x denotes a calibration sample drawn from $\mathcal{D}_{\mathrm{calib}}$, $Y_l^{\mathrm{FP}}(x)$ and $Y_l^Q(x)$ denote the outputs of layer l under the original and W4A4 computations for the same input, respectively, and ϵ is a numerical stability constant. Unlike *QVLA*, which propagates each layer-wise quantization intervention through the complete action generator and measures the resulting decoded action-chunk deviation, *Layer-nMSE* stops at the target layer and ranks layers according to local output distortion.

W-Outlier. *W-Outlier* measures the severity of weight distribution outliers under low-bit quantization:

$$\phi_l^W = \frac{q_{99.9}(|W_l|)}{\mathrm{RMS}(W_l) + \epsilon}, \tag{40}$$

where $q_{99.9}(\cdot)$ denotes the 99.9th percentile magnitude and $\mathrm{RMS}(\cdot)$ denotes the root mean square value. A larger value indicates that layer l contains more extreme weight values relative to its overall scale.

A-Outlier. *A-Outlier* measures activation distribution outliers observed on the calibration set:

$$\phi_l^A = \mathbb{E}_{x\sim\mathcal{D}_{\mathrm{calib}}}\left[\frac{q_{99.9}(|X_l(x)|)}{\mathrm{RMS}(X_l(x)) + \epsilon}\right], \tag{41}$$

where $X_l(x)$ denotes the activation of layer l for calibration sample x. The score characterizes whether the activation range of a layer is dominated by extreme values.

StepDrift. *StepDrift* captures activation variation across denoising steps in diffusion-based action generation. We first compute the high-percentile activation magnitude:

$$r_{l,t} = q_{99.9}(|X_{l,t}|), \tag{42}$$

where t denotes the denoising step. The sensitivity score is defined as:

$$\phi_l^{SD} = \frac{\mathrm{std}_t(r_{l,t})}{\mathrm{mean}_t(r_{l,t}) + \epsilon}. \tag{43}$$

A larger value indicates stronger temporal variation of layer activations during denoising.

EQS Ablations. We further evaluate three variants of EQS to isolate the contribution of individual components.

StepWise replaces horizon-level motion composition with an independent comparison of the full-precision and quantized motion increments at each execution step. Using the incremental transformations T_τ and $\tilde{T}_\tau$ defined in Sec. 3.1, the step-level end-effector motion discrepancy is

$$D_{\mathrm{ee},\tau}^{\mathrm{SW}} = \big(\log(\tilde{T}_\tau T_\tau^{-1})\big)^\vee. \tag{44}$$

Here, $D_{\mathrm{ee},\tau}^{\mathrm{SW}}$ compares the transformations commanded at step τ, rather than the poses reached after executing the first τ commands. We then apply the same configuration-aware metric and balancing factors as in EQS to each step-level discrepancy and sum the resulting scalar costs:

$$\mathcal{L}_{\mathrm{SW}} = \sum_{\tau=1}^{H} \big(S_{\mathrm{tr}} D_{\mathrm{ee},\tau}^{\mathrm{SW}}\big)^\top \Lambda_\mu(q) \big(S_{\mathrm{tr}} D_{\mathrm{ee},\tau}^{\mathrm{SW}}\big) + \lambda_g \sum_{\tau=1}^{H} (e_g^\tau)^2. \tag{45}$$

The balancing factors S_{tr} and λ_g are kept the same as in EQS; only the treatment of motion increments changes.

EQS w/o config-aware replaces the configuration-dependent operational-space inertia matrix $\Lambda_\mu(q)$ with the identity matrix I_6, while retaining horizon-level ordered motion composition and adaptive component balancing. The resulting objective is

$$\begin{aligned} \mathcal{L}_{\text{EQS-no-config}} &= \big(S_{\mathrm{tr}} D_{\mathrm{ee}}^H\big)^\top I_6 \big(S_{\mathrm{tr}} D_{\mathrm{ee}}^H\big) + \lambda_g \sum_{\tau=1}^{H} (e_\tau^g)^2 \\ &= \big\|D_t^H\big\|_2^2 + \lambda_r \big\|D_r^H\big\|_2^2 + \lambda_g \sum_{\tau=1}^{H} (e_\tau^g)^2, \end{aligned} \tag{46}$$

where D_t^H and D_r^H denote the translational and rotational components of D_{ee}^H, respectively. The balancing coefficients λ_r and λ_g are kept the same as in EQS rather than set to one. This variant therefore removes the configuration-aware inertial weighting while preserving the composed motion discrepancy and the balancing of heterogeneous action components.

EQS w/o balancing removes the adaptive balancing mechanism among heterogeneous action components while keeping the execution-level formulation unchanged:

$$\mathcal{L}_{\mathrm{EQS-no-bal}} = (D_{\mathrm{ee}}^H)^T \Lambda_\mu(q) D_{\mathrm{ee}}^H + \sum_{\tau=1}^{H} (e_g^\tau)^2. \tag{47}$$

F Additional Experimental Results

We first summarize the evaluation scale used throughout our sensitivity comparisons, including the results reported in the main text and this appendix. For $\pi_{0.5}$ on LIBERO, each method is evaluated under each mixed-precision configuration on all four task suites, using 10 evaluation seeds per suite and 1000 episodes per seed. For CogACT on SimplerEnv, each method is evaluated under each mixed-precision configuration on all four tasks, 10000 episodes per task. Detailed evaluation settings are provided in the supplementary materials.

Results for more BF16 retention settings. We further evaluate EQS under additional mixed-precision configurations to examine whether its advantage is preserved across different BF16 retention settings. Table 5 reports another two $\pi_{0.5}$ configurations on LIBERO. EQS achieves the highest average success rate in both settings, reaching 94.13% with 10% MLP and 50% attention layers retained in BF16, and 95.22% with 30% MLP and 50% attention layers retained in BF16.

Table 6: Task success rates (%) for the remaining six representative CogACT mixed-precision configurations on SimplerEnv. The reporting protocol is identical to Table 1.

Model	Setting	Method	SimplerEnv					Δ
			Carrot	Eggplant	Spoon	Stack Cube	Avg.	
CogACT	MLP retained in BF16: 30%; attention retained in BF16: 30%	**EQS (Ours)**	**69.70**	**67.39**	**80.43**	**51.43**	**67.24**	0.00
		DA-PTQ	53.91	58.26	73.91	50.67	59.19	-8.05
		Action-MSE	47.83	65.22	71.74	41.30	56.52	-10.72
		QVLA	54.35	58.70	71.74	44.78	57.39	-9.85
		Layer-nMSE	**69.70**	57.25	74.81	47.41	62.29	-4.95
		W-Outlier	55.81	44.93	77.78	**51.43**	57.49	-9.75
		A-Outlier	65.79	52.17	78.18	51.35	61.87	-5.37
		StepDrift	67.44	42.93	76.54	51.16	59.52	-7.72
		StepWise	36.96	63.04	73.91	45.65	54.89	-12.35
		EQS w/o config-aware	36.96	63.04	73.91	45.65	54.89	-12.35
		EQS w/o balancing	51.09	64.86	75.28	43.82	58.76	-8.48
	MLP retained in BF16: 30%; attention retained in BF16: 60%	**EQS (Ours)**	**69.57**	**69.57**	**82.61**	**46.73**	**67.12**	0.00
		DA-PTQ	56.58	61.07	74.64	45.71	59.50	-7.62
		Action-MSE	57.66	62.05	76.81	43.37	59.97	-7.15
		QVLA	44.93	60.87	76.81	39.13	55.44	-11.68
		Layer-nMSE	56.58	58.10	76.04	46.48	59.30	-7.82
		W-Outlier	53.91	43.48	76.52	46.15	55.02	-12.10
		A-Outlier	52.94	60.29	76.81	**46.73**	59.19	-7.93
		StepDrift	60.00	55.43	77.17	42.86	58.87	-8.25
		StepWise	52.17	65.22	69.57	30.43	54.35	-12.77
		EQS w/o config-aware	52.17	65.22	69.57	30.43	54.35	-12.77
		EQS w/o balancing	53.57	65.12	67.39	39.13	56.30	-10.82
	MLP retained in BF16: 50%; attention retained in BF16: 30%	**EQS (Ours)**	**62.16**	**73.91**	**81.74**	**50.00**	**66.95**	0.00
		DA-PTQ	52.51	64.55	80.76	42.28	60.03	-6.92
		Action-MSE	51.51	64.30	74.25	44.25	58.58	-8.37
		QVLA	50.43	64.35	74.78	40.87	57.61	-9.34
		Layer-nMSE	54.65	51.09	78.26	**50.00**	58.50	-8.45
		W-Outlier	**62.16**	39.13	78.57	**50.00**	57.47	-9.48
		A-Outlier	53.33	45.65	73.91	**50.00**	55.72	-11.23
		StepDrift	52.17	54.78	73.04	49.21	57.30	-9.65
		StepWise	47.83	67.39	76.09	36.96	57.07	-9.88
		EQS w/o config-aware	56.52	60.87	78.26	34.78	57.61	-9.34
		EQS w/o balancing	47.83	65.22	69.57	43.48	56.53	-10.42
	MLP retained in BF16: 50%; attention retained in BF16: 60%	**EQS (Ours)**	**67.44**	**72.17**	**78.07**	**47.83**	**66.38**	0.00
		DA-PTQ	55.56	67.15	76.47	43.48	60.67	-5.71
		Action-MSE	49.50	68.23	**78.07**	45.15	60.24	-6.14
		QVLA	48.79	65.22	76.85	46.96	59.46	-6.92
		Layer-nMSE	60.00	59.42	78.05	40.58	59.51	-6.87
		W-Outlier	60.00	46.38	76.19	40.00	55.64	-10.74
		A-Outlier	56.76	55.65	77.61	47.06	59.27	-7.11
		StepDrift	**67.44**	61.59	77.46	46.88	63.34	-3.04
		StepWise	44.93	65.22	76.09	39.13	56.34	-10.04
		EQS w/o config-aware	51.30	64.35	76.09	**47.83**	59.89	-6.49
		EQS w/o balancing	57.97	63.77	78.05	42.11	60.48	-5.90
	MLP retained in BF16: 60%; attention retained in BF16: 30%	**EQS (Ours)**	**64.41**	**78.26**	**82.61**	**48.89**	**68.54**	0.00
		DA-PTQ	53.61	67.17	77.13	44.00	60.48	-8.06
		Action-MSE	52.66	67.06	73.43	42.03	58.80	-9.74
		QVLA	52.17	64.13	76.09	42.39	58.70	-9.84
		Layer-nMSE	47.83	65.22	69.57	34.78	54.35	-14.19
		W-Outlier	62.90	41.30	73.91	42.86	55.24	-13.30
		A-Outlier	**64.41**	56.52	78.12	40.91	59.99	-8.55
		StepDrift	63.64	45.22	73.04	**48.89**	57.70	-10.84
		StepWise	46.43	62.50	75.00	41.07	56.25	-12.29
		EQS w/o config-aware	50.00	66.07	76.79	42.86	58.93	-9.61
		EQS w/o balancing	52.17	**78.26**	73.91	43.48	61.96	-6.58
	MLP retained in BF16: 60%; attention retained in BF16: 60%	**EQS (Ours)**	**65.71**	**71.74**	**86.96**	**50.00**	**68.60**	0.00
		DA-PTQ	53.62	65.22	72.46	39.39	57.67	-10.93
		Action-MSE	53.47	68.94	75.61	40.68	59.68	-8.92
		QVLA	53.91	66.09	74.31	40.34	58.66	-9.94
		Layer-nMSE	44.93	71.01	81.82	37.04	58.70	-9.90
		W-Outlier	55.22	41.30	73.91	45.71	54.04	-14.56
		A-Outlier	**65.71**	53.62	75.00	41.38	58.93	-9.67
		StepDrift	64.56	60.00	77.61	48.28	62.61	-5.99
		StepWise	53.04	64.35	78.26	42.61	59.57	-9.03
		EQS w/o config-aware	60.87	60.87	67.39	41.30	57.61	-10.99
		EQS w/o balancing	50.75	65.22	69.84	39.58	56.35	-12.25

The corresponding margins over the best-performing alternatives, including EQS ablations, are 0.57 and 0.38 percentage points, respectively.

Table 6 reports another six CogACT configurations on SimplerEnv. EQS achieves the highest average success rate in all six settings, with improvements over the best-performing alternatives ranging

Table 7: Inference latency (ms) of $\pi_{0.5}$ on NVIDIA GeForce RTX 4090 D GPUs.

Method	BF16 layer-retention ratios			
	MLP: 10% Attention: 30%	MLP: 10% Attention: 50%	MLP: 30% Attention: 50%	MLP: 60% Attention: 50%
EQS (Ours)	343.11	359.80	370.45	372.09
DA-PTQ	348.45	364.11	368.07	384.02
Action-MSE	357.96	359.00	363.45	396.33
QVLA	349.55	366.67	370.24	377.98
Layer-nMSE	347.88	360.41	367.35	377.39
W-Outlier	345.98	363.72	376.88	397.75
A-Outlier	347.01	353.72	363.37	367.82
StepDrift	346.39	351.65	367.08	379.57
StepWise	356.57	358.60	364.67	374.77
EQS w/o config-aware	344.88	358.36	364.32	381.37
EQS w/o balancing	339.82	356.74	367.59	385.65

from 3.04 to 8.06 percentage points. Together with the representative configurations reported in the main text, these results show that the advantage of EQS is maintained across a broad range of mixed-precision budgets.

Inference Latency. EQS helps to guide the precision allocation offline and introduces no additional computation during inference. With the BF16 retention ratios and underlying inference implementation held fixed, we expect similar inference latency across sensitivity criteria. We examine this empirically using the resulting mixed-precision allocations.

We compare $\pi_{0.5}$ allocations produced by different sensitivity criteria at fixed MLP and attention BF16 retention ratios, using the same inference backend and CUDA kernel implementations on NVIDIA GeForce RTX 4090 D GPUs. Table 7 reports mean inference latency over 1000 measurements for each method and setting. Across all four settings, EQS exhibits comparable inference latency to the other criteria. Together with the task-success results in Tables 1 and 5, these measurements show that EQS improves task success while maintaining comparable inference latency under the shared implementation.

Compatibility with HoloQ-VLA. EQS estimates layer-wise quantization sensitivity and guides the precision allocation, but does not prescribe a specific quantization procedure. We therefore further combine the EQS-guided allocation with HoloQ-VLA to examine whether EQS can complement an existing VLA quantization procedure. For each setting in Table 8, the EQS row is identical to the corresponding result reported in our sensitivity experiments, while the EQS + HoloQ-VLA row applies the HoloQ-VLA quantization procedure using the same EQS-derived BF16/W4A4 allocation.

As shown in Table 8, combining EQS with HoloQ-VLA improves the average success rate under all four evaluated mixed-precision configurations. The improvements are 2.38, 1.52, 0.06, and 0.13 percentage points when the BF16 retention ratios for MLP/attention are 10%/30%, 10%/50%, 30%/50%, and 60%/50%, respectively. The largest gain is observed under the most aggressive configuration, where the average success rate increases from 93.33% to 95.71%.

These results indicate that EQS is complementary to HoloQ-VLA rather than being tied to the quantization procedure used in our main evaluation. EQS guides the layer-wise precision allocation, while other quantization procedures like HoloQ-VLA can subsequently be applied to the layers assigned to low precision.

G BF16 Parameter Retention Analysis

In our experiments, all methods are evaluated at identical MLP and attention layer-retention ratios within each configuration, following DA-PTQ (Xu et al., 2026a). Because candidate layers differ in parameter count, identical retention ratios can lead to differences in the number of retained BF16

Table 8: Compatibility of EQS with HoloQ-VLA on $\pi_{0.5}$ with LIBERO. For each mixed-precision configuration, we compare EQS with the same EQS-derived precision allocation combined with the HoloQ-VLA quantization procedure. Δ denotes the change in average success rate relative to EQS.

Setting	Method	Object	Spatial	Goal	Long	Avg.	Δ
MLP retained in BF16: 10%; attention retained in BF16: 30%	EQS	95.17	98.13	93.75	86.25	93.33	0.00
	EQS + HoloQ-VLA	**97.83**	**98.91**	**96.09**	**90.00**	**95.71**	+2.38
MLP retained in BF16: 10%; attention retained in BF16: 50%	EQS	94.50	98.10	96.09	87.83	94.13	0.00
	EQS + HoloQ-VLA	**98.70**	**98.48**	**96.30**	**89.13**	**95.65**	+1.52
MLP retained in BF16: 30%; attention retained in BF16: 50%	EQS	**97.83**	97.83	**96.30**	88.91	95.22	0.00
	EQS + HoloQ-VLA	**97.83**	**98.70**	95.22	**89.35**	**95.28**	+0.06
MLP retained in BF16: 60%; attention retained in BF16: 50%	EQS	97.84	**98.38**	**96.72**	90.00	95.74	0.00
	EQS + HoloQ-VLA	**98.48**	98.26	96.30	**90.43**	**95.87**	+0.13

parameters across different sensitivity criteria. We examine these differences alongside task success to assess the relationship between parameter retention and the performance gains of EQS.

For $\pi_{0.5}$, the number of retained BF16 parameters under EQS ranges from 7.34M fewer to 9.18M more than competing criteria, corresponding to approximately -2.45% and +3.06% of the action generator's total parameter count, respectively. For CogACT, EQS retains up to 9.46M fewer BF16 parameters, corresponding to approximately 10.6% of the action generator's total parameter count, while its maximum increase is 2.37M, corresponding to approximately 2.7%.

In multiple configurations, EQS achieves higher task success even when retaining fewer BF16 parameters than competing baselines, demonstrating the effectiveness of its layer prioritization under a smaller BF16 parameter budget. These results show that the gains are not solely attributable to retaining more parameters in BF16. Extending the allocation procedure to enforce a fixed BF16 parameter budget is a direction for future work, as discussed in Appendix I.

H Real-World Evaluation Protocol

All compared methods are evaluated using the same initial object placements. The 40 trials for each task cover 10 distinct placements, including positions near the corners of the robot's workspace to probe performance in challenging, potentially near-singular configurations. For each task and method, we conduct 4 trials at each of the same 10 initial placements, resetting the robot and objects before every trial. A trial is counted as successful only if the robot grasps the target object and places it in the designated white cup or white basket. For cup-placement tasks, the cup must also remain upright at the end of the trial.

I Discussion and Limitations

EQS evaluates quantization-induced motion discrepancies and their configuration-dependent execution significance without task-level rollouts. This design makes execution-level sensitivity estimation tractable during calibration, while leaving observation feedback and contact dynamics outside the scope of the criterion. EQS therefore provides an intermediate assessment between action deviations and task outcomes, rather than a complete model of closed-loop task performance.

Our evaluation compares sensitivity criteria under a common protocol with fixed MLP and attention layer-retention ratios. Because candidate layers differ in parameter count, this protocol does not enforce exactly matched BF16 parameter counts or nominal weight-storage budgets, as quantified in Appendix G. Integrating EQS with precision allocation under explicit parameter, storage, or latency constraints is a direction for future work toward hardware-constrained deployment. Finally, our experiments focus on BF16/W4A4 mixed precision, following previous works (Xu et al., 2026a;b; Wang et al., 2026); extending the evaluation to other bit widths and quantization formats remains future work.