HSA: HIERARCHICAL SENSITIVITY-AWARE QUANTIZATION FOR RESOURCE-CONSTRAINED EMBODIED MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied foundation models combine visual perception, language-conditioned reasoning, and action generation, but their memory footprint constrains deployment on resource-limited robotic platforms. Post-training weight quantization reduces this footprint, yet its perturbations can affect continuous action predictions and closed-loop execution. We propose Hierarchical Sensitivity-Aware Quantization (HSA), which augments a 4-bit AWQ backbone with BF16 residual blocks. An action-guided score combines a reconstruction-error proxy with action-output Jacobians, and a layer–channel–block hierarchy determines where the residuals are retained. Across LIBERO evaluations with VLA-Adapter, OpenVLA, and FastWAM, HSA improves average success over other methods by 2.55%–2.90% and remains within 0.30% of the original models. Meanwhile, it reduces GPU memory by 53.2%–67.2%. The quantized policy also matches the original policy in the real-robot setting under various conditions.

1 INTRODUCTION

Generalist robot policies increasingly combine visual perception, language conditioning, and action generation in large embodied models (Brohan et al., 2022; Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024; Bi et al., 2025; Yuan et al., 2026). Their vision encoders, language backbones, projectors, and action-generation modules can exceed the memory budgets of edge devices mounted on robots, making model storage and peak execution memory practical deployment constraints.

Post-training quantization (PTQ) compresses pretrained checkpoints without full retraining. For embodied models, however, the output of a quantized policy is an action or action-conditioned prediction used repeatedly in a closed loop. Weight perturbations must therefore be examined both at the action output and through task execution. The effect is also heterogeneous across model components and local weight blocks, so reconstruction error alone does not identify where retained precision is most relevant to the policy output.

We introduce Hierarchical Sensitivity-Aware Quantization (HSA). HSA scores each candidate block by combining an activation-scaled reconstruction-error proxy with the Jacobian of continuous actions with respect to the corresponding output channel. It then selects BF16 residuals through a layer–channel–block hierarchy and adds them to the quantized linear output at inference, without changing the policy interface or action post-processing.

Our contributions are:

- We study weight-only PTQ for embodied models at two distinct levels: fixed-observation action fidelity and closed-loop task success, with quantization scope and deployment resources evaluated separately.
- We propose HSA, which combines action-output Jacobians with a reconstruction proxy and allocates BF16 residuals through a layer–channel–block hierarchy.
- Across three simulated model settings, HSA recovers most of the success loss from 4-bit quantization while enjoying a reduction in peak memory. Evaluation on a real robot further compares the original and quantized policies under various conditions.

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.

2 Related Work

Embodied foundation models. Large-scale robot policies increasingly use transformer and vision-language backbones. RT-1 and RT-2 connect such representations to multi-task robot control, while Open X-Embodiment broadens training across robots and datasets (Brohan et al., 2022; Zitkovich et al., 2023; Open X-Embodiment Collaboration, 2023). OpenVLA uses a vision encoder, language backbone, and action prediction module, whereas π_0 uses flow-based continuous action generation (Kim et al., 2024; Black et al., 2024). VLA-Adapter, TinyVLA, and SmolVLA pursue smaller or more efficient VLA designs (Wang et al., 2025; Wen et al., 2024; Shukor et al., 2025); Motus and FastWAM instead model action-conditioned dynamics for planning or policy support (Bi et al., 2025; Yuan et al., 2026). These model families motivate a quantization method that operates on shared linear-layer structure without changing their action interfaces.

Post-training quantization. PTQ replaces high-precision weights, activations, or both with lower-bit representations. LLM.int8() isolates outlier channels for stable 8-bit matrix multiplication (Dettmers et al., 2022); GPTQ uses approximate second-order information to reduce layer-wise reconstruction error (Frantar et al., 2023); SmoothQuant migrates activation outliers into weights for W8A8 quantization (Xiao et al., 2023); and AWQ protects activation-salient channels through equivalent scaling before low-bit weight quantization (Lin et al., 2024b). OmniQuant and DuQuant use learnable calibration or outlier redistribution to improve low-bit PTQ (Shao et al., 2024; Lin et al., 2024a). HSA builds on AWQ and uses its reconstructed backbone and error signal to define candidates for sparse high-precision residuals.

Quantization for embodied models. Recent work directly addresses VLA quantization. QVLA uses action-space sensitivity for bit allocation (Xu et al., 2026); QuantVLA studies scale calibration and selective quantization (Zhang et al., 2026); and HBVLA studies 1-bit PTQ for VLAs (Yan et al., 2026). HSA instead retains a uniform 4-bit AWQ backbone and restores BF16 residuals for selected local blocks. We evaluate this residual formulation in both VLA and WAM settings.

3 Method

3.1 AWQ Backbone and Residual Candidates

HSA first constructs a shared low-bit backbone and represents its local reconstruction gaps as candidate corrections. This separates the quantized computation used by every input from the small set of high-precision values selected in later stages.

Let M_{fp} be a trained embodied policy and let $\mathcal{L}$ denote its target linear layers. A layer $l \in \mathcal{L}$ has weight $W_l \in \mathbb{R}^{m_l \times n_l}$ and maps token $x_{l,t}$ to $h_{l,t} = W_l x_{l,t} + b_l$. We quantize weights in input-axis groups of $G = 64$ using affine 4-bit codes with a learned zero point. Activations, embeddings, normalization, softmax, and action post-processing remain unchanged. The action head and cross-modal and proprioceptive projectors are also kept at checkpoint precision; Section 4.3 examines this scope choice.

We apply an equivalent input-channel scaling before quantization (Lin et al., 2024b). For a positive scale vector d_l, its reconstructed weight in the original coordinate system is

$$\hat{W}_l(d_l) = \text{dequant}(Q_4(W_l \,\text{diag}(d_l)))\,\text{diag}(d_l)^{-1}. \tag{1}$$

We choose d_l from the search grid by minimizing layer-output reconstruction error on a bounded activation sample. HSA uses the reconstructed backbone $\hat{W}_l = \hat{W}_l(d_l)$ and adds selected residuals from

$$\Delta W_l = W_l - \hat{W}_l. \tag{2}$$

Residuals are computed and scored in FP32 and stored in BF16 after selection. Equation 2 supplies the local error signal; the next stage weights that signal by its relevance to the action output.

3.2 ACTION-GUIDED BLOCK SENSITIVITY

Reconstruction error identifies where quantization changes a linear layer, but not how strongly the affected output channel influences the predicted action. We therefore score each residual block using both quantities.

We partition each row of ΔW_l into input blocks $B_{l,r}$ of size G. The scale provides a local input-importance proxy

$$I_{l,k} = \frac{\max(|d_{l,k}|, 10^{-6})^2}{\mathrm{mean}_q \max(|d_{l,q}|, 10^{-6})^2 + 10^{-6}}, \qquad e^{\mathrm{scale}}_{l,i,r} = \sum_{k \in B_{l,r}} (\Delta W_{l,i,k})^2 I_{l,k}. \tag{3}$$

This term is a ranking proxy for local reconstruction error, not an unbiased estimate of output mean-squared error. To connect it to policy behavior, we differentiate the de-normalized action chunk $a(s) \in \mathbb{R}^J$ with respect to each target layer output. In Eq. 5, $a(s)$ denotes the model-specific differentiable action readout used for offline sensitivity estimation; the readouts and differentiation paths are specified in Appendix A. Action coordinates are weighted by inverse calibration variance,

$$w_j = \frac{(\mathrm{Var}_s[a_j(s)] + 10^{-8})^{-1}}{\mathrm{mean}_q(\mathrm{Var}_s[a_q(s)] + 10^{-8})^{-1}}, \tag{4}$$

and the sensitivity of output channel i is the weighted mean squared Jacobian

$$g_{l,i} = \frac{\sum_s \sum_{j=1}^{J} w_j \sum_t \left(\partial a_j(s) / \partial h_{l,t,i}(s)\right)^2}{\sum_s \sum_{j=1}^{J} w_j T_l(s) + 10^{-12}}, \tag{5}$$

where $T_l(s)$ is the number of valid output tokens. These vector–Jacobian products are computed offline on calibration observations; they require neither demonstration actions nor environment rollouts.

We normalize the reconstruction and action-sensitivity signals within each layer and combine them as

$$\bar{e}_{l,i,r} = \frac{e^{\mathrm{scale}}_{l,i,r}}{\mathrm{mean}_{i,r}\, e^{\mathrm{scale}}_{l,i,r} + 10^{-12}}, \quad \bar{g}_{l,i} = \frac{\max(g_{l,i}, 10^{-12})}{\mathrm{mean}_q \max(g_{l,q}, 10^{-12})}, \quad e^{\mathrm{AG}}_{l,i,r} = \bar{e}_{l,i,r} \bar{g}^{\alpha}_{l,i}. \tag{6}$$

We use $\alpha = 1$. The product in Eq. 6 is an action-guided ranking proxy: it favors blocks that combine a substantial quantization residual with a sensitive output channel. Closed-loop task success is evaluated separately.

3.3 HIERARCHICAL SELECTION AND INFERENCE

The action-guided score ranks individual blocks. HSA converts these dense scores into a structured residual branch by first selecting layers, then channels within those layers, and finally input blocks within those channels.

Let $S_l = \mathrm{mean}_{i,r}\, e^{\mathrm{AG}}_{l,i,r}$ rank layers and $C_{l,i} = \sum_r e^{\mathrm{AG}}_{l,i,r}$ rank channels. Given layer ratio ρ_L, channel ratio ρ_C, and K_B blocks per selected channel, HSA forms

$$\begin{aligned} \mathcal{L}^\star &= \operatorname*{TopK}_{l \in \mathcal{L}}(S_l, \lceil \rho_L |\mathcal{L}| \rceil), \\ \mathcal{C}^\star_l &= \operatorname*{TopK}_{i}(C_{l,i}, \lceil \rho_C m_l \rceil), \\ \mathcal{P}^\star &= \{(l,i,r) : l \in \mathcal{L}^\star,\ i \in \mathcal{C}^\star_l,\ r \in \operatorname*{TopK}_{q}(e^{\mathrm{AG}}_{l,i,q}, K_B)\}. \end{aligned} \tag{7}$$

Zero-score blocks are discarded, and an optional global cap may truncate $\mathcal{P}^\star$. Our main configuration uses $\rho_L = 0.10$, $\rho_C = 0.03$, $K_B = 1$, and no additional global cap. For a selected block, the forward pass adds its BF16 residual to the shared W4 backbone:

$$\tilde{h}_{l,t,i} = (\hat{W}_l x_{l,t} + b_l)_i + \sum_{r:(l,i,r) \in \mathcal{P}^\star} \sum_{k \in B_{l,r}} x_{l,t,k}\ \mathrm{BF16}(\Delta W_{l,i,k}). \tag{8}$$

The hierarchy is a structured budget heuristic. Under an equal-cost additive score, unconstrained global Top-K is optimal for that proxy objective; HSA instead restricts the candidate set to concentrate residual computation in fewer linear layers and channels. All scoring and selection occur offline.

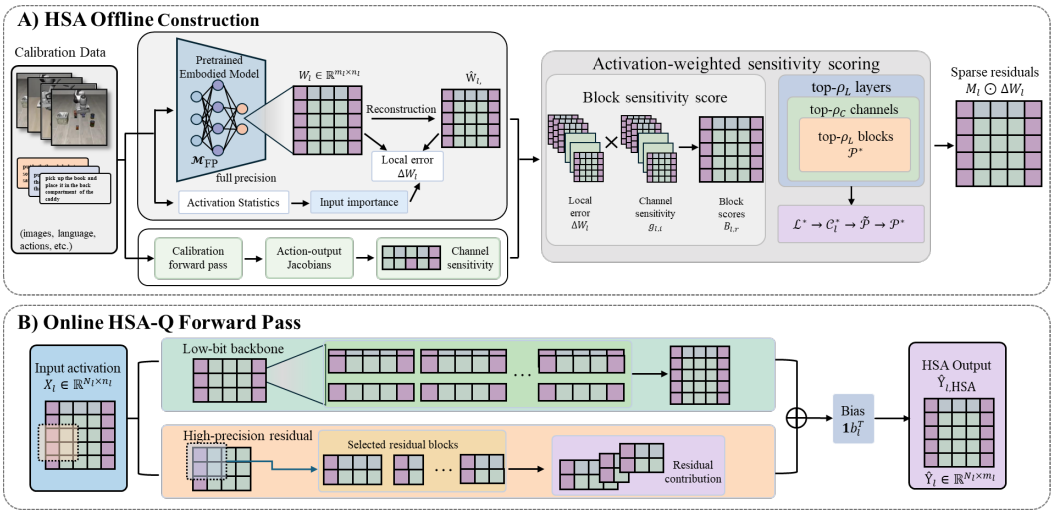

Figure 1: Overview of HSA. Offline, reconstruction produces the local residual ΔW_l, which is combined with input importance to obtain the scale-weighted block term in Eq. 3. The complete score further incorporates action sensitivity through Eq. 6 before hierarchical layer, channel, and block selection. Online, the selected BF16 residual branch is added to the shared W4 linear output.

Offline construction and runtime state. Calibration, residual construction, action-Jacobian evaluation, score normalization, and hierarchical selection are completed before deployment. Algorithm 1 summarizes this sequence. The deployed model retains the shared W4 backbone, the selected BF16 residual values, and their block indices; the Jacobians, scores, and unselected residuals are not used by the inference pass. The index metadata and BF16 residual payload are distinct storage terms, while peak allocated GPU memory also depends on the low-bit backend and runtime buffers and is measured separately in Section 4.4.

For layer l, define $\mathcal{P}_l^\star = \{(i,r) : (l,i,r) \in \mathcal{P}^\star\}$. Applying Eq. 8 to T_l valid tokens introduces an arithmetic estimate proportional to $T_l G|\mathcal{P}_l^\star|$ multiply–accumulate terms. Across layers, the corresponding estimate is $\sum_l T_l G|\mathcal{P}_l^\star|$; it excludes index handling, kernel launches, and scheduling effects. The policy-query latency reported in Section 4.4 measures those implementation effects under the current backend.

Algorithm 1 Offline construction of HSA

Require: Policy M_{fp}, calibration bank $\mathcal{D}$, target linear layers $\mathcal{L}$
Ensure: W4 policy with selected BF16 residuals

1: Calibrate the shared backbone and reconstruct $\hat{W}_l$ using Eq. 1.
2: Compute $\Delta W_l = W_l - \hat{W}_l$ and $e_{l,i,r}^{\text{scale}}$ for all candidate blocks.
3: Evaluate action Jacobians on the guidance subset and compute $g_{l,i}$.
4: Form $e_{l,i,r}^{\text{AG}}$ with Eq. 6.
5: Select layers, channels, and blocks using Eq. 7.
6: Store block indices and BF16 residual values; apply Eq. 8 at inference.
7: **return** M_{HSA}

4 Experiments

4.1 Setup

Models and tasks. We evaluate VLA-Adapter, OpenVLA, and FastWAM in simulation. They represent a compact VLA, a larger open-source VLA, and a WAM-style action-conditioned dynamics model, respectively. The evaluations use the Spatial, Object, Goal, and Long suites of LIBERO (Liu et al., 2023). We report success rate for each suite and the average across the four suites. We

also evaluate a fine-tuned $\pi_{0.5}$ policy (Black et al., 2025) on a real-robot pickup task with variation in object color and initial position.

Methods and main quantization scope. We compare the original precision model, RTN, AWQ, and HSA. The VLA-Adapter comparison includes RTN and AWQ; the OpenVLA and FastWAM comparisons use AWQ. Within each setting, all 4-bit methods use input-axis groups of 64, the same target-layer scope, and the same evaluation interface. In the main VLA configuration, quantization targets the vision-language linear layers, while the action head and cross-modal and proprioceptive projectors remain at checkpoint precision. For VLA-Adapter, the target set contains 378 linear layers: 96 DINO, 113 SigLIP, and 169 language layers. Section 4.3 analyzes this scope choice. HSA reuses the AWQ backbone and calibration bank and adds action-guided residual selection and compensation.

Calibration and diagnostic data. AWQ uses 512 ordered calibration observations. For VLA-Adapter, action guidance uses the first eight observations and differentiates all 56 coordinates of the 8×7 de-normalized action chunk, including the gripper coordinate; it is computed offline without demonstration action labels. Model-specific calibration and differentiation settings are detailed in Appendix A. The VLA-Adapter Long diagnostics use a separate fixed set of 236 observations from 59 trajectories and 10 tasks. They report task- and trajectory-balanced standardized MSE against the same floating-point actions over six continuous dimensions, excluding the gripper. Quantized weights remain fixed across these observations. The single-block diagnostic uses a common vision-language-plus-action-head W4 base, whereas the main VLA model keeps the action head at checkpoint precision. Further details appear in Appendix A.

Metrics. Closed-loop evaluations report task success. Fixed-observation diagnostics report standardized continuous-action MSE or its reduction after restoring one residual block; these measurements are distinct from rollout success. Resource measurements use peak allocated GPU memory during evaluation rather than weight storage alone, and policy-query latency is reported when available. The latency measurement includes residual-branch overhead under the current low-bit backend.

4.2 Main Results

Table 1: Success rates on LIBERO task suites (%). ΔAVG is relative to the original precision model for the same backbone. Bold marks the best 4-bit result within each model.

Model	Method	Bits	Spatial	Object	Goal	Long	AVG / ΔAVG
VLA-Adapter	Original	–	97.8	99.2	97.2	95.0	97.30 / 0.00
	RTN	4	88.4	99.2	93.0	90.8	92.85 / -4.45
	AWQ	4	94.4	99.6	92.8	91.8	94.65 / -2.65
	HSA	4	**97.8**	**100.0**	**96.6**	**94.4**	**97.20** / -0.10
OpenVLA	Original	–	84.7	88.4	79.2	53.7	76.50 / 0.00
	AWQ	4	82.3	87.2	74.6	50.7	73.70 / -2.80
	HSA	4	**84.2**	**88.6**	**78.8**	**53.4**	**76.25** / -0.25
FastWAM	Original	–	96.0	99.4	96.2	94.8	96.60 / 0.00
	AWQ	4	92.4	98.2	92.2	90.8	93.40 / -3.20
	HSA	4	**95.8**	**99.4**	**96.4**	**93.6**	**96.30** / -0.30

Across the three model settings in Table 1, 4-bit AWQ reduces average success by 2.65–3.20 percentage points relative to the corresponding original model. HSA recovers 2.55–2.90 points of that gap and finishes within 0.10–0.30 points of the original models. The gains over AWQ occur across all four LIBERO suites; they are largest on Goal, where the three improvements range from 3.8 to 4.2 points.

4.3 Quantization and Residual Analysis

Quantization scope. Quantization error varies substantially across VLA-Adapter components and scopes. Under the fixed-observation diagnostic in Figure 2(a), quantizing the language model or action head in isolation produces higher standardized continuous-action MSE than quantizing either vision encoder. Extending the diagnostic W4 scope from the vision-language stack to the vision-language stack plus action head increases the MSE from 6.42×10^{-3} to 13.27×10^{-3}; the panel labels this latter scope "All modules," with projectors kept at checkpoint precision. Table 2 provides a separate closed-loop comparison in which "All" denotes all eligible linear modules. At 4 bits, this broader table scope reduces average success from 92.85% to 48.10% for RTN and from 94.65% to 77.55% for AWQ, whereas the corresponding 8-bit scope differences are smaller. The main VLA configuration therefore applies W4 quantization to the vision-language linear backbone and retains the action head and projectors at checkpoint precision.

Table 2: Closed-loop success rates for different VLA-Adapter quantization scopes on LIBERO (%). VL denotes the vision-language backbone; All extends quantization to all eligible linear modules.

Method	Scope / Bits	Spatial	Object	Goal	Long	AVG	ΔAVG
Original	– / –	97.8	99.2	97.2	95.0	97.30	0.00
RTN	All / 8	96.8	99.4	95.6	93.4	96.30	-1.00
RTN	All / 4	66.6	45.2	53.0	27.6	48.10	-49.20
AWQ	All / 8	96.4	99.4	96.8	94.4	96.75	-0.55
AWQ	All / 4	83.8	98.4	78.0	50.0	77.55	-19.75
RTN	VL / 8	96.8	99.6	96.6	92.8	96.45	-0.85
RTN	VL / 4	88.4	99.2	93.0	90.8	92.85	-4.45
AWQ	VL / 8	97.8	98.8	96.6	93.4	96.65	-0.65
AWQ	VL / 4	94.4	99.6	92.8	91.8	94.65	-2.65

Local repair. The effect of restoring precision also varies within components. Figure 2(b) starts from the common diagnostic W4 base and restores one BF16 residual block at a time. Most sampled DINO and SigLIP blocks and many language-model and action-head blocks reduce action MSE, while other interventions are neutral or increase it. The signed repair gain therefore measures the local effect of each intervention under fixed observations; closed-loop task success remains a separate evaluation.

Selection ablation. The fixed-budget comparison in Table 3 separates the benefit of adding residuals from differences among the reported selectors. Random-R and WeightErr-R recover 1.30 and 1.95 average-success points over AWQ, respectively, showing that residual capacity is useful even without the complete HSA selector. Global Top-K reaches 96.30%, a 1.65-point gain over AWQ, matching HSA-L and trailing complete HSA by 0.90 points. The table also includes the HSA-L and HSA-LC variants under the same budget. Complete layer–channel–block HSA reaches 97.20%, 0.60 points above the stronger WeightErr-R residual baseline and 0.65 points above HSA-LC.

Table 3: Ablation of residual selection on VLA-Adapter (%). Δ_{AWQ} is relative to AWQ 4-bit. All residual variants use the same residual budget.

Method	Spatial	Object	Goal	Long	AVG	Δ_{AWQ}
Original	97.8	99.2	97.2	95.0	97.30	+2.65
AWQ 4-bit	94.4	99.6	92.8	91.8	94.65	0.00
Random-R	96.8	99.4	95.4	92.2	95.95	+1.30
WeightErr-R	97.0	99.6	96.2	93.6	96.60	+1.95
Global Top-K	96.8	99.2	96.0	93.2	96.30	+1.65
HSA-L	96.2	99.2	96.4	93.4	96.30	+1.65
HSA-LC	97.2	99.4	96.2	93.4	96.55	+1.90
HSA	**97.8**	**100.0**	**96.6**	**94.4**	**97.20**	**+2.55**

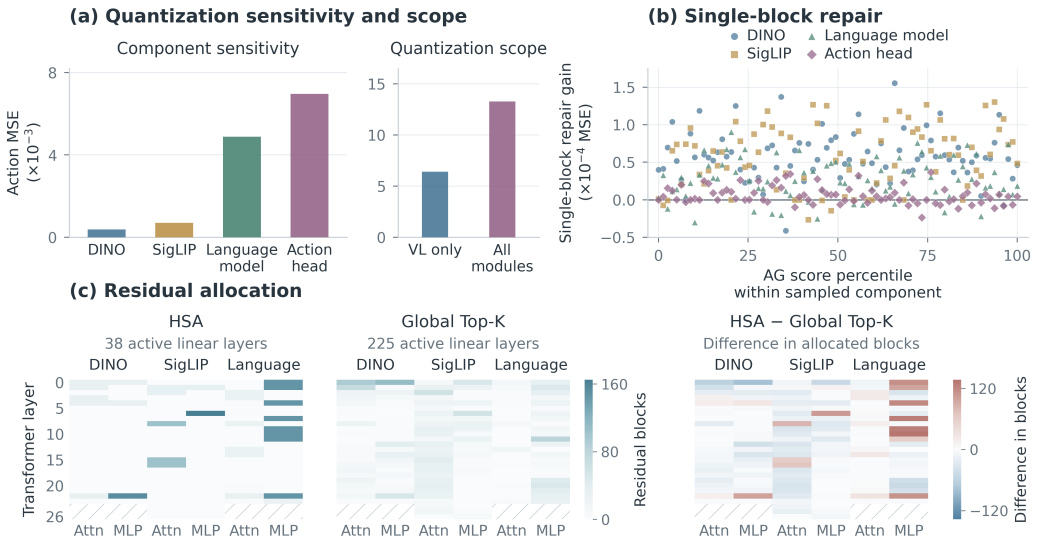

Figure 2: Quantization sensitivity, single-block repair, and residual allocation on VLA-Adapter Long with AWQ W4/G64. (a) Component sensitivity quantizes one named component at a time while the others remain at checkpoint precision; Quantization scope compares VL Only with the panel label All modules, which here denotes the vision-language stack plus action head, with projectors at checkpoint precision. Bars report task- and trajectory-balanced standardized continuous-action MSE against the same floating-point reference actions under fixed observations, in units of $\times 10^{-3}$. (b) Each point restores one sampled 64-weight BF16 residual block; 80 blocks are sampled per component. The horizontal axis is the action-guided score percentile within that sampled component, not a shared absolute score. Restoration uses a common diagnostic W4 base containing the vision-language stack and action head. Repair gain is the MSE before restoration minus the MSE after restoration, so a positive value denotes lower action MSE; the vertical axis is in units of $\times 10^{-4}$. The main VLA configuration uses a VL-only W4 base. (c) HSA and global Top-K use the same action-guided score and retain 2,334 BF16 blocks of 64 weights each. The first two heatmaps share an absolute block-count scale; the third is HSA minus global Top-K, where a positive red value means that HSA allocates more blocks. Hatched cells have no corresponding layer, whereas unhatched zero-valued cells are existing locations with no selected block. The reported 38 and 225 values count active linear layers with residuals, not transformer depth.

Residual allocation. The hierarchy changes where a matched residual budget is placed. With the same action-guided score and 2,334-block budget, Figure 2(c) shows that HSA places residuals in 38 active linear layers, compared with 225 for global Top-K. It assigns 59.2% of the blocks to the language model, whereas global Top-K assigns 27.4%. The depth–module heatmaps and their difference panel localize this redistribution: HSA uses a smaller set of active linear layers and shifts a larger share of the fixed budget toward the language backbone. Panel (c) reports allocation counts under matched conditions; the corresponding task-success comparison appears in Table 3.

4.4 Resource Usage

Table 4 relates task performance to peak allocated GPU memory and policy-query latency. On VLA-Adapter and OpenVLA, HSA retains 53.2% and 67.2% peak-memory reductions relative to the original models; its residual branch increases peak memory by 0.20% and 0.23% over AWQ, respectively. On VLA-Adapter, the branch adds 1.427 ms to AWQ policy-query latency, and neither quantized configuration is faster than the original model under the current backend. The measured resource benefit is therefore memory reduction rather than query acceleration.

Table 4: Resource usage and performance for VLA models. Peak mem. denotes peak allocated GPU memory during evaluation; policy-query latency is available for VLA-Adapter only.

Model	Method	Bits	Peak mem. (MB)	Mem. reduction	AVG / ΔAVG	Query latency (ms)
VLA-Adapter	Original	–	4278.00	0.0%	97.30 / 0.00	78.560
	AWQ	4	**1998.00**	**53.3%**	94.65 / -2.65	85.628
	HSA	4	2002.00	53.2%	**97.20** / -0.10	87.055
OpenVLA	Original	–	14758.75	0.0%	76.50 / 0.00	–
	AWQ	4	**4835.03**	**67.2%**	73.70 / -2.80	–
	HSA	4	4846.13	67.2%	**76.25** / -0.25	–

4.5 Real-Robot Validation

We evaluate the original and HSA-quantized versions of a $\pi_{0.5}$ policy fine-tuned for real-robot pickup. Both use the same reported task definition, with object color and initial object position varied across the evaluation, and each policy is tested in 20 trials. As shown in Table 5, both policies complete all 20 trials. The available setup and protocol details, together with representative images from the test conditions, are provided in Appendix C.

Table 5: Real-robot pickup success under object-color and initial-position variation. Each policy is evaluated for 20 trials.

Model	Method	Bits	Task	Generalization factors	Successes	Success rate (%)
$\pi_{0.5}$	Original	–	Pickup	Color and initial position	20 / 20	100.0
	HSA	4	Pickup	Color and initial position	**20 / 20**	**100.0**

5 Discussion and Limitations

The evaluation covers three model settings on LIBERO and one real-robot $\pi_{0.5}$ pickup task with 20 trials per policy. The real-robot test varies object color and initial position but does not cover sensor noise, contact-rich tasks, longer horizons, or recovery behavior. Fixed-observation action MSE and closed-loop success are reported as separate measurements. The action-guided score is a first-order, offline proxy for ranking residual candidates rather than an estimate of rollout return, consistent with the neutral and harmful single-block repairs observed in the diagnostic analysis.

The experiments also use fixed layer, channel, and block selection ratios. Adapting these ratios to model structure, task difficulty, or a deployment-specific memory and latency budget remains future work. At the system level, the measured reduction concerns peak allocated GPU memory; the current backend does not reduce policy-query latency. Latency gains would require VLA-aware low-bit kernels, scheduling, and caching mechanisms analogous to optimized LLM serving systems (Kwon et al., 2023).

6 Conclusion

We presented HSA, which combines an AWQ reconstruction-error proxy with continuous-action sensitivity and allocates BF16 residuals through a layer–channel–block hierarchy. Across three LIBERO model settings, the structured residual branch recovers most of the task-success loss from 4-bit AWQ while preserving its peak-memory reduction with little additional peak memory. The quantized policy also matches the original policy in the tested real-robot pickup setting. These results position HSA as a memory-oriented PTQ approach for the evaluated embodied models; policy-query acceleration remains a backend-level objective.

AI USE STATEMENT

In this work, we used OpenAI Codex to revise and polish the manuscript and to organize the associated code repository. We reviewed all AI-assisted changes and take responsibility for the final content of this work.

REFERENCES

Hongzhe Bi et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025. URL https://arxiv.org/abs/2512.13030.

Kevin Black, Noah Brown, James Darpinian, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Kevin Black et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Tim Dettmers et al. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2208.07339.

Elias Frantar et al. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *Proceedings of the International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.17323.

Moo Jin Kim et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Woosuk Kwon et al. vLLM: Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165. URL https://arxiv.org/abs/2309.06180.

Haokun Lin et al. DuQuant: Distributing outliers via dual transformation makes stronger quantized llms. *arXiv preprint arXiv:2406.01721*, 2024a. URL https://arxiv.org/abs/2406.01721.

Ji Lin et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024b. URL https://arxiv.org/abs/2306.00978.

Bo Liu et al. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. https://libero-project.github.io/main.html, 2023.

Open X-Embodiment Collaboration. Open x-embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023. URL https://arxiv.org/abs/2310.08864.

Wenqi Shao et al. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *Proceedings of the International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.13137.

Mustafa Shukor et al. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/2506.01844.

Yixuan Wang et al. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*, 2025. URL https://arxiv.org/abs/2509.09372.

Junjie Wen et al. TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024. URL https://arxiv.org/abs/2409.12514.

Guangxuan Xiao et al. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2211.10438.

Yixuan Xu et al. QVLA: Not all channels are equal in vision-language-action model's quantization. *arXiv preprint arXiv:2602.03782*, 2026. URL https://arxiv.org/abs/2602.03782.

Xiaolong Yan et al. HBVLA: Pushing 1-bit post-training quantization for vision-language-action models. *arXiv preprint arXiv:2602.13710*, 2026. URL https://arxiv.org/abs/2602.13710.

Tianyuan Yuan et al. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Jian Zhang et al. QuantVLA: Scale-calibrated post-training quantization for vision-language-action models. *arXiv preprint arXiv:2602.20309*, 2026. URL https://arxiv.org/abs/2602.20309.

Brianna Zitkovich et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2307.15818.