A. General Quantization Formulations

Post-training quantization (PTQ) [25, 38, 49] reduces memory footprint and accelerates inference without additional training. This subsection introduces a generic, bit-parameterized formulation. Here, we use tildes to denote integer tensors and hats to denote their dequantized floating approximations.

Consider a linear layer $Y = XW$ without bias. Let b_X and b_W be the activation and weight bit widths, respectively. Activations are quantized per token using an unsigned grid, and weights are quantized per output channel using a signed grid. Therefore, the integer activations $\tilde{X}$ are obtained as:

$$\tilde{X} \;=\; \mathrm{clip}\Big(\,\mathrm{round}(X/\Delta_X) + z_X,\; 0,\; 2^{\,b_X} - 1\Big), \quad (18)$$

with dequantization

$$\hat{X} \;=\; \Delta_X\big(\tilde{X} - z_X\big). \quad (19)$$

Integer weights for output channel o are:

$$\tilde{W}^{(o)} = \mathrm{clip}\Big(\,\mathrm{round}\big(W^{(o)}/\Delta_W^{(o)}\big), \\ -2^{\,b_W-1},\, 2^{\,b_W-1} - 1\Big). \quad (20)$$

with dequantization

$$\hat{W}^{(o)} \;=\; \Delta_W^{(o)}\,\tilde{W}^{(o)}. \quad (21)$$

Here, $\Delta_X > 0$ is the activation scale estimated from a small unlabeled calibration buffer, and $z_X \in \{0, \ldots, 2^{b_X} - 1\}$ is the integer zero point for the unsigned activation grid. Each output channel o uses a per-channel scale $\Delta_W^{(o)} > 0$ and a symmetric signed grid $\{-2^{b_W-1}, \ldots, 2^{b_W-1} - 1\}$. Dequantization multiplies stored integers by their corresponding scales to obtain floating-point approximations.

B. DuQuant Implementation Details

With DuQuant design, we then instantiate the transform, beginning with the smoothing step. Specifically, to balance the difficulty of quantizing activations and the relative ease of quantizing weights, we apply per-channel smoothing with a diagonal matrix Λ:

$$Y = (X\Lambda)(\Lambda^{-1}W) = X'W'. \quad (22)$$

$$\Lambda_j = \frac{\big(\max|X_{:,j}|\big)^{\alpha}}{\big(\max|W_{j,:}|\big)^{1-\alpha}}, \qquad \alpha \in [0,1]. \quad (23)$$

We then operate on the transformed pair

$$(X', W') = (X\Lambda,\; \Lambda^{-1}W). \quad (24)$$

Following DuQuant, we further factorize the layer with block orthogonal rotations $\hat{R}_{(1)}, \hat{R}_{(2)}$ and a permutation P:

$$\boxed{\begin{aligned} Y &= XW \\ &= \underbrace{\big[(X\Lambda)\,\hat{R}_{(1)}\,P\,\hat{R}_{(2)}\big]}_{G}\;\underbrace{\big[\hat{R}_{(2)}^{\top}\,P^{\top}\,\hat{R}_{(1)}^{\top}\,(\Lambda^{-1}W)\big]}_{G^{-1}} \end{aligned}} \quad (25)$$

All three matrices are orthogonal and therefore $P^{-1} = P^{\top}$. The left bracket G acts on activations before integerization and the right bracket G^{-1} is folded into the weights to preserve equivalence. After this factorization, we can quantize G on the activation side and G^{-1} on the weight side at the chosen bit widths b_X and b_W respectively and then execute the integer matrix multiplication with the corresponding dequantization scales.

C. How logits transfer from the language backbone to DiT in a VLA

These scales appear in DiT attention only through dequantization. For a head of width d we set

$$\hat{Q} = s_q\,\tilde{Q}, \qquad \hat{K} = s_k\,\tilde{K}, \qquad \hat{V} = s_v\,\tilde{V}. \quad (26)$$

The logits matrix that drives attention is

$$L = \frac{\hat{Q}\hat{K}^{\top}}{\sqrt{d}} = \frac{s_q s_k}{\sqrt{d}}\,\tilde{Q}\tilde{K}^{\top}, \quad (27)$$

and the attention matrix is

$$A = \mathrm{softmax}(L). \quad (28)$$

The per-head output is

$$Y = A\,\hat{V} = s_v\,A\,\tilde{V}. \quad (29)$$

Let the output projection be dequantized as $\hat{W}_o = s_o\,\tilde{W}_o$. The block output after concatenating heads and applying the output projection is

$$Z = \mathrm{Concat}(Y_h)\,\hat{W}_o = s_o\;\mathrm{Concat}(Y_h)\,\tilde{W}_o. \quad (30)$$

Therefore, $s_q s_k$ sets an effective temperature $T_{\mathrm{eff}} = \sqrt{d}/(s_q s_k)$ that controls attention sharpness, and $s_v s_o$ primarily determines how much energy is injected into the residual stream.

D. QuantVLA Parameters

Appendix: Quantization and calibration. GR00T N1.5 and OPENPI π0.5 use the same DuQuant configuration and the same statistical calibration. We set the weight bit width to 4 and the activation bit width to 8, which reduces memory

and bandwidth while keeping accuracy stable. The block size is 64 on both the input and the output, so that block orthogonal rotations and collected statistics share the same granularity. For the LLM and DiT backbone, we enable channel permutation to redistribute large channels and reduce outliers. The row rotation modrestoredstore, which applies a rotation before each linear map and the inverse after the map, so that the real-valued function is preserved while improving the suitability of the layer for quantization. During post-training calibration, we set the activation percentile to 99.9 to determine the clipping range, we use 32 batches to estimate scales, and we apply per-channel smoothing with a coefficient of 0.15 to prevent a few channels from dominating a shared scale.

After integerization, both models use the same statistical matching. Attention temperature matching learns one scalar α for each head and aligns the scale of the student logits with the teacher so that the attention distribution is neither overly sharp nor overly flat. Output head balancing learns one scalar β for each layer and restores the residual stream energy at the module output. In our runs the scope of β is limited to the diffusion transformer head. We fit α and β from a small unlabeled buffer using 128 steps with at most 5 trials for each task, we clamp $\log \alpha$ and $\log \beta$ with a limit of 0.30, and we keep a neutrality band ε of 0.03 so that both scalars remain close to 1 when the estimate is uncertain. We fold α into the dequantization scale for inference and apply β at the module output.

E. Comparison with other PTQ Method

As shown in Table 5, SmoothQuant, a built-in PTQ method in NVIDIA-OPT, performs reasonably at **W8A8** precision. In contrast, QuantVLA achieves comparable or slightly better results under the more aggressive **W4A8** setting. When SmoothQuant is extended to quantize both the LLM and the DiT MLP, performance remains competitive under W8A8 precision. However, QuantVLA operates at a lower bitwidth while maintaining stable performance across all task suites. In particular, improvements are observed on the long-horizon task, where low-precision inference typically accumulates greater drift over sequential generation. The average success rate under W4A8 is also slightly higher than the floating-point baseline.

Method	Spatial	Object	Goal	Long	Avg
π0.5	98.5%	99.0%	97.5%	93.5%	97.1%
+SmoothQuant (LLM)	97.5%	98.5%	98.0%	92.5%	96.6%
+SmoothQuant (LLM + DiT (MLP))	98.0%	99.0%	99.0%	92.0%	97.0%
+QuantVLA (LLM)	98.5%	99.0%	96.5%	96.5%	97.6%
+QuantVLA	**98.5%**	**98.0%**	**98.0%**	**96.0%**	**97.6%**

Table 5. Additional quantization comparison on the LIBERO benchmark for OpenPI π0.5.

These results indicate that QuantVLA sustains task performance under more aggressive quantization and remains robust in long-sequence scenarios. Therefore, it provides a more favorable accuracy–efficiency trade-off for low-bit inference in VLA models when deployment efficiency is a primary consideration.

F. Extended Benchmark Evaluation

As shown in Table 6, we further evaluate QuantVLA on the Pick-and-Can manipulation benchmark. Under **W4A8** precision, SmoothQuant exhibits a noticeable drop in performance compared to the FP16 baseline. In contrast, QuantVLA substantially narrows the gap and maintains a higher success count under the same precision setting.

Method	Precision	PickCan
GR00T	FP16	31 / 50
+ SmoothQuant	W4A8	16 / 50
+ QuantVLA	W4A8	27 / 50

Table 6. Quantization results on Pick-and-Can.

Although performance does not fully match the floating-point baseline, the results demonstrate that QuantVLA better preserves task performance under aggressive quantization. This suggests that the proposed design mitigates the sensitivity of the action head to quantization noise in manipulation scenarios.

G. Applicability Beyond DiT-Based VLA Models

We evaluated OpenVLA, which uses a deeper 32-layer language backbone than the 18-layer backbones studied here and a non-DiT action head, resulting in different language–action coupling. Thus, the DiT-specific ATM and OHB mechanisms are not directly applicable. Nevertheless, QuantVLA matches OpenVLA performance (Table 7).

Model	Precision	Spatial
OpenVLA	FP16	84.7%
+ QuantVLA	**W8A16**	**86.0%**

Table 7. Quantization results on LIBERO-Spatial for OpenVLA.