CARE-VLA: CAPACITY ALLOCATION VIA RESPONSE EVALUATION FOR 1-BIT VLA QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models are rapidly advancing embodied intelligence, but their large parameter footprints remain a major obstacle to deployment on resource-constrained robotic platforms. One-bit post-training quantization (PTQ) offers an attractive route to extreme model compression, yet preserving model behavior at this precision remains challenging. To the best of our knowledge, vector quantization has not yet been explored for VLA models in the 1-bit regime, where deciding which concrete reconstruction deserves scarce additional capacity becomes critical. We introduce Capacity Allocation via Response Evaluation (CARE-VLA), a post-training vector-quantization framework that allocates capacity among concrete low-bit reconstruction candidates according to their response-level distortion. Rademacher-projected reverse-mode pullbacks efficiently estimate native-response sensitivity without materializing full Jacobians, allowing concrete reconstruction errors to be evaluated directly in response space. Complete-policy component interventions further calibrate heterogeneous native-response gains onto a common action-discrepancy scale, enabling reconstruction candidates across vision, language, action, and auxiliary modules to compete under a shared bit budget. At 1.09 bits per weight (BPW), CARE-VLA achieves 91.0% and 88.9% mean LIBERO success on $\pi_{0.5}$ and OpenVLA-OFT, respectively, and 69.7% visual-matching and 53.2% variant-aggregation success on SIMPLER with CogACT. Executed $\pi_{0.5}$ linear weights achieve $14.5\times$ storage compression relative to 16-bit storage, including codec overhead. On the RTX 4090, the quantized $\pi_{0.5}$ model achieves a measured $2.09\times$ speedup. Our work advances extreme VLA compression toward practical deployment on resource-constrained robotic hardware.

1 INTRODUCTION

Vision-language-action (VLA) models integrate perception, language, and action generation for robotic control (Zitkovich et al., 2023; Kim et al., 2025b; Black et al., 2025), but their parameter footprints impede resource-constrained deployment. Post-training quantization (PTQ) compresses pretrained policies without costly retraining. Near one bit, however, large weight perturbations can alter actions and compound through closed-loop execution.

Recent VLA quantization methods use action-space sensitivity (Xu et al., 2026; Akbari et al., 2026), scale calibration (Zhang et al., 2026), or policy-aware binarization (Yan et al., 2026). Structured vector quantization (VQ) offers expressive near-one-bit reconstruction spaces (Tseng et al., 2024a;b). In this regime, the allocation variable is not merely a component or channel precision level: each additional bit selects between concrete reconstruction candidates whose errors can interact differently with policy responses. Visual features, language states, and actions also have different dimensions and scales. Allocating one budget thus requires both candidate-specific response evaluation and a common measure of cross-component benefit.

Sensitivity alone does not identify a valuable reconstruction. Additional capacity helps when the higher-rate candidate removes errors that the policy amplifies. A sensitive stripe may gain little if those errors remain unchanged. The relevant allocation unit is therefore a pair of concrete reconstructions, evaluated through the policy that consumes them.

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.

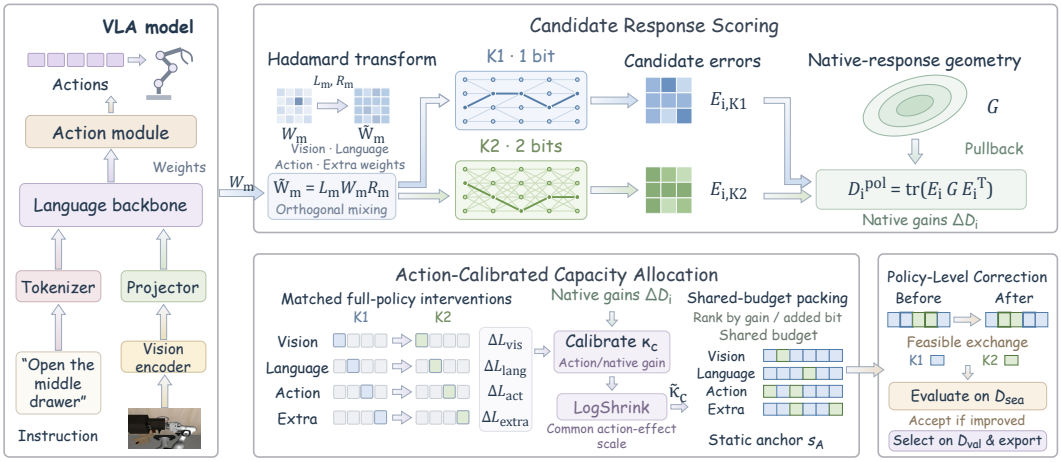


Figure 1: Overview of CARE-VLA. **Candidate Response Scoring**: Hadamard transforms L_m, R_m map module weights W_m into compatible $K1$ (blue) and $K2$ (green) trellis reconstructions; their errors are evaluated with response geometry G. **Action-Calibrated Capacity Allocation**: four-block policies depict Vision, Language, Action, and Extra interventions. Measured gain ΔL_c calibrates native gains through κ_c; LogShrink regularizes scales before shared-budget packing forms anchor s^A. **Policy-Level Correction**: outlined stripes exchange capacity, with acceptance on $\mathcal{D}_{\text{sea}}$ and prefix selection on $\mathcal{D}_{\text{val}}$. Matrices, paths, and quotas are schematic.

We develop *Capacity Allocation via Response Evaluation* (CARE-VLA), a post-training vector-quantization framework for near-one-bit VLA compression. CARE-VLA couples candidate-specific response geometry with action-calibrated global allocation over a fixed bank of compatible trellis reconstructions. Placing the gains of concrete one- and two-bit candidates on a common action scale lets reconstructions throughout the model compete under one storage budget.

Randomized reverse-mode pullbacks estimate reusable response geometry without forming full Jacobians. The geometry retains directions within each input block, so candidates with comparable weight distortion can receive different scores according to their downstream response effects. Component interventions calibrate native gains by their measured effect on complete-policy actions, and LogShrink regularizes their relative scales. Shared-budget packing then ranks candidates by action benefit per added bit. Budget-feasible exchanges correct residual interactions, followed by held-out policy selection. The fixed candidate bank separates reconstruction from allocation throughout this procedure; response evaluation is needed only during quantization.

At 1.09 bits per weight (BPW), CARE-VLA achieves 91.0% and 88.9% mean LIBERO success on $\pi_{0.5}$ and OpenVLA-OFT, and 69.7% visual-matching and 53.2% variant-aggregation success on SIMPLER with CogACT. Executed $\pi_{0.5}$ linear weights achieve $14.5\times$ storage compression relative to 16-bit storage, including codec overhead. Matched scoring and quota controls show that success depends on which concrete reconstructions receive capacity, not component quotas alone.

Our contributions include:

1. We introduce fixed-candidate near-one-bit capacity allocation for VLA models. Concrete $K1/K2$ trellis-vector reconstructions across vision, language, action, and Extra compete under one shared storage budget, shifting allocation from weight reconstruction to response evaluation.
2. We develop randomized candidate-specific response scoring. Rademacher-projected reverse-mode pullbacks estimate native-response geometry without forming full Jacobians, allowing each concrete reconstruction error to be evaluated according to its magnitude and direction.
3. We introduce action-calibrated capacity allocation. Complete-policy component interventions align heterogeneous native gains onto a common action scale for budgeted packing, allowing capacity to move between components according to estimated action benefit. A lightweight policy-level refinement further adjusts the selected reconstructions using complete-policy feedback, without changing the candidate bank or storage budget.

2 Related Work

Vision-Language-Action Models. VLA models transfer pretrained visual and language representations into task-conditioned robot control. RT-2 (Zitkovich et al., 2023) expresses actions as tokens within a vision-language model, while OpenVLA (Kim et al., 2025b) provides an open generalist policy trained on diverse robot demonstrations. Diffusion Policy (Chi et al., 2023) models continuous action sequences through iterative denoising, and Octo (Ghosh et al., 2024) combines diffusion decoding with flexible interfaces for adaptation across robot setups. Building on pretrained vision-language backbones, π_0 (Black et al., 2025) and $\pi_{0.5}$ (Physical Intelligence et al., 2025) use flow-based action generation, whereas CogACT (Li et al., 2024) conditions a diffusion action transformer on multimodal representations. OpenVLA-OFT (Kim et al., 2025a) improves decoding efficiency through parallel prediction, action chunking, and continuous-action regression. The resulting policies differ in how multimodal representations condition control and whether action sequences are predicted directly or through iterative updates. Compression must preserve the responses through which these heterogeneous components interact. The proposed CARE-VLA method addresses this requirement through candidate response evaluation and cross-component capacity allocation.

Low-Bit Post-Training Quantization. PTQ reduces model precision using calibration data without retraining the original model. One line of work improves scalar quantization by controlling reconstruction error and outliers. GPTQ (Frantar et al., 2023) uses approximate second-order information to reconstruct layer outputs. AWQ (Lin et al., 2024) identifies salient weights through activation statistics and protects them with channel scaling. SmoothQuant (Xiao et al., 2023) redistributes activation outliers into weights through equivalent transformations, while OmniQuant (Shao et al., 2024) learns clipping and transformation parameters during calibration. SqueezeLLM (Kim et al., 2024) and SpQR (Dettmers et al., 2024) combine low-bit representations with higher-precision exceptions. A second line develops richer reconstruction spaces at very low rates. QuIP (Chee et al., 2023) introduces incoherence processing; QuIP# (Tseng et al., 2024a) combines Hadamard transforms with lattice codebooks. AQLM (Egiazarian et al., 2024) represents weight groups with additive codebooks, and QTIP (Tseng et al., 2024b) searches trellis paths to encode long vectors without storing an exponentially large codebook. BiLLM (Huang et al., 2024) studies binary PTQ, while CRVQ (Xu et al., 2025) assigns extra vector-quantization capacity to selected channels. Transformations reshape quantization difficulty, while structured codes and selective precision determine how the representation budget is spent. These approaches motivate expressive reconstruction candidates and selective capacity use; applying them to a complete VLA also requires evaluating their effects on policy responses.

For VLA models, QVLA (Xu et al., 2026) uses action-space channel sensitivity for mixed-precision quantization and pruning. ActQuant (Akbari et al., 2026) combines action-guided inter-tensor bit allocation with intra-tensor scale optimization using action-aware curvature, reporting 2–4-bit backbone settings with full-precision projectors and action heads. HB-VLA (Yan et al., 2026) constructs an action-aware rectified Hessian for hybrid group-wise binarization in the Haar wavelet domain. QuantVLA (Zhang et al., 2026) calibrates attention temperature and output energy in selective W4A8/W4A4 layouts, where W and A denote weight and activation precision. These methods establish the importance of action-aware sensitivity, mixed precision, calibration, and policy-aware binarization. CARE-VLA addresses the reconstruction allocation problem induced by structured near-one-bit VQ: it evaluates fixed $K1/K2$ trellis errors in native-response geometry, calibrates heterogeneous component gains onto a common action scale, and allocates a shared model-wide budget.

3 Method

3.1 Preliminaries

Vision-Language-Action (VLA) Models. VLA models receive visual observations V, an instruction ℓ, and proprioception o_{prop}, and predict an action chunk $A \in \mathbb{R}^{H \times d_a}$:

$$A = \pi_W(z; \eta), \qquad z = (V, \ell, o_{\text{prop}}), \quad W = \{W_{\text{vis}}, W_{\text{lang}}, W_{\text{act}}, W_{\text{extra}}\}. \tag{1}$$

Here π_W is the policy parameterized by W, z collects its inputs, o_{prop} is the robot-state vector, H the chunk horizon, and d_a the number of action coordinates. The subscripts on W denote vision,

language, action, and Extra parameters, with Extra grouping executed projectors and adapters. The variable η fixes action-generation randomness. This view includes visual encoders, multimodal backbones, and action generators (Kim et al., 2025b; Xu et al., 2026). Calibration compares policies at the same (z, η).

Vector Quantization. Scalar quantization represents each weight independently, whereas vector quantization (VQ) jointly encodes groups of weights and captures correlations among their coordinates. In conventional VQ at k index bits per weight, a vector of dimension d is mapped to one of 2^{kd} codewords in a codebook $\mathcal{Q} \subset \mathbb{R}^d$. Explicit codebooks grow exponentially with d; structured VQ instead provides efficiently searchable reconstruction spaces (Tseng et al., 2024a;b).

In our setting, each transformed weight block admits compatible one- and two-bit trellis-vector reconstructions, denoted $K1$ and $K2$. These labels indicate index rates rather than numbers of clusters. A stripe is a group of weight rows that shares a reconstruction choice. The candidate bank is constructed before allocation and remains fixed; our CARE-VLA selects one reconstruction for each stripe under the shared bit budget.

3.2 From Weight Error to Response Evaluation

A native response is the differentiable output interface through which a component influences subsequent policy computation. It may be a feature representation or an action output, depending on the component. Let $r_c(z; W) \in \mathbb{R}^{q_c}$ be this response and M_c a positive-semidefinite metric. Its distortion under quantized weights $\widehat{W}$ is

$$\mathcal{L}_c(\widehat{W}) = \mathbb{E}_z \|r_c(z; \widehat{W}) - r_c(z; W)\|^2_{M_c}, \tag{2}$$

where $\mathcal{C} = \{\text{vis}, \text{lang}, \text{act}, \text{extra}\}$ is the component set, c indexes its members, and q_c is the response dimension. The expectation $\mathbb{E}_z$ averages calibration contexts with matched randomness implicit. For $v \in \mathbb{R}^{q_c}$, $\|v\|^2_{M_c} = v^\top M_c v$, where $M_c \in \mathbb{R}^{q_c \times q_c}$ weights response coordinates and $\top$ denotes transpose. Native responses include visual features, language states, and executed actions. Their coordinates differ, but each measures the effect of quantization at an interface consumed by the policy.

Let $w = \text{vec}(W)$ concatenate reference weights and $\delta w = \text{vec}(\Delta W)$, where $\Delta W = \widehat{W} - W$. Local linearization yields a generalized Gauss–Newton form (Martens, 2020):

$$\mathcal{L}_c(W + \Delta W) = \delta w^\top \mathbb{E}_z[J_{c,w}^\top M_c J_{c,w}] \delta w + O\big((\|\delta w\|_2)^3\big)\,. \tag{3}$$

The Jacobian $J_{c,w} \in \mathbb{R}^{q_c \times n_w}$ maps perturbations of the n_w flattened weights to responses; vec concatenates matrix entries. The remainder is cubic in the Euclidean perturbation norm $\|\delta w\|_2$ under the smoothness assumptions in Section A. Two reconstructions with equal weight error can therefore have different response distortion. This motivates scoring the candidate error itself, rather than attaching a sensitivity scalar to a stripe independently of its reconstruction.

Native scoring and complete-policy action calibration share the full-precision reference.

3.3 Compatible Trellis-Vector Candidates

For every quantized stripe, we construct compatible $K1$ and $K2$ trellis-vector reconstructions before allocation. For a linear weight matrix $W_m \in \mathbb{R}^{d_o \times d_{\text{in}}}$, signed Hadamard transforms give $\widetilde{W}_m = L_m W_m R_m$. Here m indexes executed modules, d_o, d_{in} are output and input widths, $\widetilde{W}_m$ is the transformed weight, and L_m, R_m are orthogonal output and input transforms of matching sizes. This preprocessing spreads coordinate outliers before trellis reconstruction. The transforms, codebooks, scales, and candidate reconstructions are fixed thereafter, so CARE-VLA decides which reconstruction receives capacity.

A stripe i contains h_i transformed output rows in module $m(i)$ and component $c(i)$. Scoring first restores original input coordinates: $E^\star_{m,k} = (\widehat{\widetilde{W}}_{m,k} - \widetilde{W}_m) R_m^\top$, where $\widehat{\widetilde{W}}_{m,k}$ is the transformed reconstruction at index rate $k \in \{1, 2\}$. Let $E_{i,k,b} \in \mathbb{R}^{h_i \times d_b}$ select stripe i and input block b of width d_b from $E^\star_{m(i),k}$. It is paired with the original-input metric $G_{m,b} \in \mathbb{R}^{d_b \times d_b}$:

$$D^{\text{pol}}_{i,k} = \sum_b \text{tr}(E_{i,k,b} G_{m(i),b} E_{i,k,b}^\top). \tag{4}$$

The scalar $D_{i,k}^{\mathrm{pol}}$ scores a stripe's candidate error; the superscript pol denotes the policy-response criterion, tr sums diagonal entries, and the sum covers the stripe's input blocks. All three architectures use the same Generic $K1/K2$ construction. The response metric evaluates these fixed errors and guides their allocation.

Moving a stripe from $K1$ to $K2$ costs one additional index bit per weight: $\Delta B_i = n_i$ bits for a stripe with n_i weights. Selectors, scales, and decoder metadata also enter the serialized budget. We do not add a floating-point residual to the reconstruction. Formal index-rate and transform definitions appear in Section C.1.

3.4 Candidate Response Scoring

Scalar random projections estimate response geometry without forming the full Jacobian. For an executed linear module $h = XW_m^\top$, $X \in \mathbb{R}^{T\times d_{\mathrm{in}}}$ contains T input tokens and $h \in \mathbb{R}^{T\times d_o}$ its outputs. Draw $\xi \in \{-1,+1\}^{q_c}$ with independent equiprobable coordinates and differentiate

$$\zeta = q_c^{-1/2}\xi^\top M_c^{1/2} r_c, \qquad U = \partial\zeta/\partial h, \tag{5}$$

where ζ is a scalar projection of the native response at a fixed context, $M_c^{1/2}$ is the symmetric positive-semidefinite square root, and $U \in \mathbb{R}^{T\times d_o}$ is the gradient with respect to module outputs. Rademacher probing follows stochastic second-moment estimation (Hutchinson, 1989) and recovers the normalized response pullback in expectation before block projection (Section A.2).

Across all three architectures, we form a token-diagonal input metric:

$$\widehat{G}_{m,b}^{\mathrm{diag}} = \mathrm{avg}_{z,\xi}\, X_b^\top \,\mathrm{diag}\big(\|U_{\tau,:}\|_2^2/d_o\big)\, X_b, \tag{6}$$

where $X_b \in \mathbb{R}^{T\times d_b}$ selects input block b, $\tau \in \{1,\dots,T\}$ indexes tokens, and $U_{\tau,:}$ selects all output-gradient coordinates at token τ. The operator diag forms a $T\times T$ diagonal matrix, while $\mathrm{avg}_{z,\xi}$ averages collected context–probe pairs. The superscript diag indicates that token interactions are diagonalized. We use $\widehat{G}_{m,b}^{\mathrm{diag}}$ as $G_{m,b}$ in Eq. (4).

On sampled $\pi_{0.5}$ stripes, the blockwise approximation preserves denser native-response rankings with Spearman correlations of 0.790–0.965 (Section E.11). The metric preserves within-block directions and is reused across candidates. For $\pi_{0.5}$, native responses are projected visual tokens, layerwise prefix key/value states, and the final executed action prefix for vision, language, and action, respectively. Extra uses its executed-interface local metric. All architectures use token-diagonal scoring and component calibration; interface normalization is specified in Section C.2.

3.5 Action-Calibrated Capacity Allocation

Native interfaces need a common scale before competing for bits. Their dimensions, normalization, and downstream roles differ, so equal native-score reductions need not have equal effects on actions. For component c, we measure the mean complete-policy action discrepancies $\overline{L}_c^{(1)}$ and $\overline{L}_c^{(2)}$ under $K1$ and $K2$, holding the other components at full precision. Define the native gain $\Delta D_i = D_{i,1}^{\mathrm{pol}} - D_{i,2}^{\mathrm{pol}}$ for stripe i in the fixed bank. Component moment matching gives

$$\kappa_c = \frac{\overline{L}_c^{(1)} - \overline{L}_c^{(2)}}{\sum_{i\in\mathcal{I}_c}\Delta D_i}. \tag{7}$$

Here $\mathcal{I}_c$ contains component c's stripes, and κ_c converts their aggregate native gain into measured action-discrepancy reduction. Calibration requires a positive numerator and denominator, as specified with the discrepancy definitions in Section C.3. Multiplication by κ_c preserves within-component ranking while making gains comparable across components.

To regularize estimated component ratios, LogShrink contracts deviations from their geometric center:

$$\log\bar{\kappa} = \frac{1}{|\mathcal{C}|}\sum_c \log\kappa_c, \qquad \log\widetilde{\kappa}_c = \alpha\log\kappa_c + (1-\alpha)\log\bar{\kappa}. \tag{8}$$

Here $\log$ is the natural logarithm, $|\mathcal{C}|$ the component count, $\bar{\kappa}$ the geometric-mean scale, $\widetilde{\kappa}_c$ the regularized scale, and $\alpha \in [0,1]$ the retained fraction of each log deviation. Varying α interpolates from a shared scale to the original ratios while preserving within-component stripe ranking. Section A derives these properties; Tables 12 and 18 examine calibration and scale sensitivity.

Shared-budget allocation. The allocation decision is the benefit of a particular reconstruction per additional bit. With calibrated scales, the gain density is

$$\rho_i = \widetilde{\kappa}_{c(i)} \max\{0, \Delta D_i\}/\Delta B_i. \tag{9}$$

Here $\max\{0, \Delta D_i\}$ retains positive candidate gains, so ρ_i estimates action-discrepancy reduction per added index bit. Across all three architectures, allocation starts from the fixed $K1$ bank and promotes $K2$ candidates in descending positive-density order, skipping upgrades that exceed the shared serialized budget. On the frozen $\pi_{0.5}$ instance, density packing matches the dual solution and attains the budget-dual bound within numerical precision (Section B.3).

3.6 Policy-Level Correction

The complete policy contains joint residual interactions beyond the additive static scores. We correct the static anchor through feasible exchanges under the same storage budget, funding $K2$ promotions with unused capacity or $K1$ demotions. Complete-policy feedback evaluates the joint effect of each exchange.

For a proposed configuration s, complete-policy evaluation uses

$$\mathcal{J}(s) = \mathrm{mean}_z\, \mathcal{E}(z; s) + \beta\, \mathrm{CVaR}_\gamma(\mathcal{E}(z; s)). \tag{10}$$

Here $s = (s_i)_i$, with stripe choices $s_i \in \{1, 2\}$, defines a quantized policy. Its context loss $\mathcal{E}(z; s)$ combines action error, temporal differences, cumulative translation-action error, and gripper disagreement. The coefficient $\beta \geq 0$ weights conditional value at risk (CVaR): CVaR_γ averages the largest $\lceil (1-\gamma)n \rceil$ losses among n contexts, where $\gamma \in [0, 1)$ and $\lceil \cdot \rceil$ rounds upward. The mean measures average discrepancy and the upper tail emphasizes contexts with large response errors.

Search contexts $\mathcal{D}_{\text{sea}}$ evaluate improvements in $\mathcal{J}$ subject to response and storage constraints. Candidates and local scores remain fixed. Independent held-out contexts $\mathcal{D}_{\text{val}}$ select the lowest-$\mathcal{J}$ eligible prefix, meaning the configuration after an accepted number of exchanges. Section C.3 specifies proposal construction, acceptance, and selection; Section A.7 derives the residual interactions.

4 Experiments

4.1 Experimental Setup

Models and benchmarks. We evaluate $\pi_{0.5}$ (Physical Intelligence et al., 2025) and OpenVLA-OFT (Kim et al., 2025a) on LIBERO (Liu et al., 2023), averaging success equally over Spatial, Object, Goal, and Long. CogACT (Li et al., 2024) is evaluated on SIMPLER (Li et al., 2025) under Google visual matching (VM) and variant aggregation (VA). VM matches the reference visual conditions; VA aggregates across visual variations. The models cover flow-based, direct-action, and diffusion-based generation, respectively. VM and VA are reported separately, with per-category results in Section D.

Quantization and baselines. All quantized methods cover executed linear weights across vision, language, action, and Extra, including projectors and action heads. Embeddings, normalization parameters, biases, and convolutions retain their original precision; the unused language-model head is excluded in OFT. We compare full precision, BiLLM (Huang et al., 2024), QTIP (Tseng et al., 2024b), QVLA (Xu et al., 2026), and HB-VLA (Yan et al., 2026). We adapt QVLA's mixed-precision allocation and pruning to the same executed-linear scope at 2.50 BPW, retaining 16-bit activations. Uniform one-bit and layer-reconstruction allocation provide controlled $\pi_{0.5}$ references. The latter weights errors by calibration inputs, a stronger local criterion than unweighted weight mean squared error (MSE).

Calibration and allocation. Calibration observations come from LIBERO for $\pi_{0.5}$ and OpenVLA-OFT, and from SIMPLER for CogACT. Response metrics are estimated on fixed observation contexts. Component calibration compares reference and quantized policies under matched action-generation randomness. Exchange search and prefix selection use separate context subsets, while benchmark episodes measure closed-loop success. Response interfaces and data roles can be found in Section C in the appendix. Bits per weight (BPW) use all executed linear weights as the denominator and include serialized codec overhead (Section C.1); CARE-VLA rates are truncated to two decimal places. Mechanism ablations and visual analyses use $\pi_{0.5}$.

Table 1: LIBERO success (%, higher is better). Average weights the four suites equally. The $\pi_{0.5}$ result for CARE-VLA averages three evaluation banks; mechanism ablations share one matched bank. Bold marks the highest quantized score within each model, including ties; full precision is a reference. BPW denotes bits per weight.

Method	BPW	Spatial	Object	Goal	Long	Average
$\pi_{0.5}$ / LIBERO						
Full precision	16.00	98.5	99.0	97.5	93.5	97.1
QVLA	2.50	0.0	0.0	0.0	0.0	0.0
BiLLM	1.08	76.4	91.2	54.7	12.1	58.6
QTIP	1.00	84.3	85.9	70.6	15.5	64.1
HB-VLA	1.13	92.8	95.6	86.5	76.7	87.9
CARE-VLA (ours)	1.09	**96.4**	**98.9**	**91.5**	**77.3**	**91.0**
OpenVLA-OFT / LIBERO						
Full precision	16.00	97.6	98.4	97.9	94.5	97.1
QVLA	2.50	0.0	0.0	0.0	0.0	0.0
BiLLM	1.08	74.7	87.4	51.8	11.6	56.4
QTIP	1.00	81.6	84.3	70.8	14.7	62.9
HB-VLA	1.13	83.4	88.6	76.0	**85.8**	83.5
CARE-VLA (ours)	1.09	**92.4**	**96.8**	**90.3**	76.2	**88.9**

Table 2: CogACT success on SIMPLER (%). VM denotes visual matching and VA variant aggregation. The settings are kept separate rather than pooled into a single score. Bold marks the best quantized result.

Method	BPW	Google VM	Google VA
Full precision	16.00	74.8	61.3
QVLA	2.50	0.0	0.0
BiLLM	1.08	24.6	9.4
QTIP	1.00	49.6	34.2
HB-VLA	1.13	67.2	49.2
CARE-VLA (ours)	1.09	**69.7**	**53.2**

4.2 Comparison with State-of-the-Art Methods

Table 1 reports LIBERO success. At 1.09 BPW, CARE-VLA reaches 91.0% on $\pi_{0.5}$ and 88.9% on OpenVLA-OFT, improving over QTIP by 27.0 and 26.1 percentage points, respectively. The largest separation occurs on Long: on $\pi_{0.5}$, success rises from QTIP's 15.5% to 77.3%; on OpenVLA-OFT, it rises from 14.7% to 76.2%. These sequential tasks require consistent actions over extended execution, making them particularly sensitive to accumulated control deviations.

CARE-VLA exceeds HB-VLA's average success by 3.1 points on $\pi_{0.5}$ and 5.4 points on OpenVLA-OFT. QVLA yields no successful episodes at the evaluated setting. ActQuant (Akbari et al., 2026) independently reports zero QVLA success on $\pi_{0.5}$ at 2.0 backbone BPW under a different, 60-episode calibration protocol.

On diffusion-based CogACT (Table 2), CARE-VLA reaches 69.7% VM and 53.2% VA success, exceeding QTIP by 20.1 and 19.0 points and HB-VLA by 2.5 and 4.0 points.

4.3 Mechanism Ablations on $\pi_{0.5}$

Table 3 separates method construction from matched controls. Candidate Response Scoring raises average success from 59.6% to 87.2% at the same budget. Action-Calibrated Capacity Allocation produces the 91.1% static policy, and Policy-Level Correction reaches 91.4%. Complete calibration and correction variants appear in Table 11.

Complementary roles of response scoring and component calibration. We replace native-response gains with local-reconstruction gains while retaining the same component-calibration procedure, LogShrink, candidate bank, and budget. Success drops from 91.1% to 62.2%, with losses

Table 3: Allocation ablations on $\pi_{0.5}$ (success, %), with 500 matched episodes per suite. Controls replace scores or assignments in CARE-VLA (static), retaining candidates, budget, calibration, and LogShrink; shuffling preserves component $K2$ quotas and payload. Uniform one-bit uses a lower rate. Bold marks the best construction result, including ties.

Allocation	BPW	Spatial	Object	Goal	Long	Average
Method construction						
Uniform one-bit	1.00	82.0	88.4	69.8	15.6	64.0
Local allocation (uncalibrated)	1.09	76.8	78.6	70.2	12.6	59.6
Response scoring (uncalibrated)	1.09	95.0	**98.6**	92.0	63.0	87.2
CARE-VLA (static)	1.09	**97.6**	98.4	91.2	77.2	91.1
CARE-VLA	1.09	96.8	**98.6**	**92.2**	**78.0**	**91.4**
Matched controls on CARE-VLA (static)						
Replace response with local scores	1.09	85.4	74.0	75.6	13.8	62.2
Shuffle within component quotas	1.09	87.0	83.4	74.4	18.6	65.9

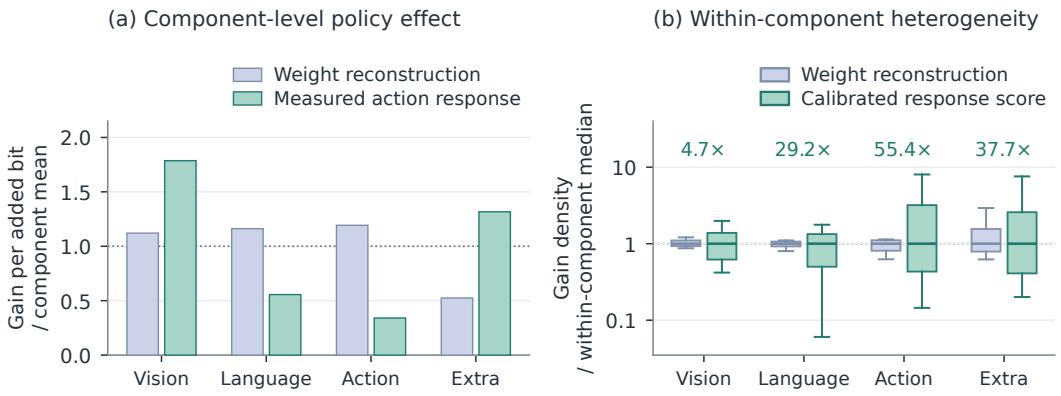


Figure 2: Quantization sensitivity in $\pi_{0.5}$. (a) $K1$-to-$K2$ weight and action gains per added bit on 128 contexts, with other components at full precision. Each criterion is normalized by its component mean (dotted line). (b) Stripe gain densities normalized by component medians: boxes show 25th–75th percentiles, whiskers 10th–90th, and horizontal strokes medians. Teal labels give response-score 90th/10th ratios. Both panels are dimensionless; Extra groups projectors and adapters.

across all suites. The comparison without shrinkage likewise favors native response, 90.8% versus 66.7% (Table 15). Action calibration makes component gains comparable, while native-response scoring identifies useful candidate reconstructions.

Stripe selection beyond component quotas. Shuffling which stripes receive $K2$ within each component reduces average success from 91.1% to 65.9%, and Long from 77.2% to 18.6%. Preserving component quotas is insufficient: task success depends on which reconstructions receive capacity. Section E.3 gives the protocol and three-seed offline results.

4.4 Response Geometry and Joint Allocation

Figure 2 shows why allocation needs both component calibration and stripe selection. Component upgrades with similar weight-reconstruction benefit induce different action responses. Response gains also vary substantially within each component; median normalization in panel (b) cancels scalar calibration and exposes this stripe-level heterogeneity. These observations distinguish two allocation decisions: how much capacity to assign across components, and which concrete reconstructions to promote within each component. Calibration makes heterogeneous native gains comparable on a common action scale, while candidate-response scoring distinguishes stripes that share the same component. The fixed-quota shuffle control in Table 3 complements this analysis: changing stripe assignments while preserving component capacities reduces task success, linking the observed heterogeneity to closed-loop behavior.

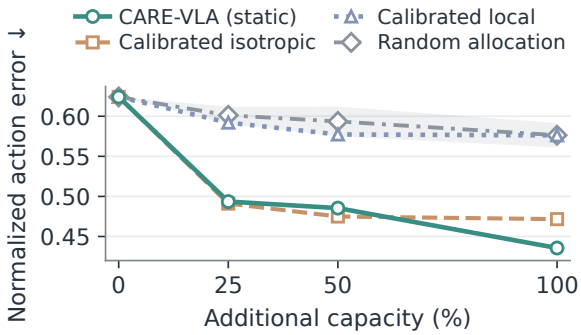


Table 4: Efficiency on $\pi_{0.5}$ across all four components. Weight storage includes codec overhead.

Method	GB↓	Speedup↑
Native BF16	5.65	1.0×
CARE-VLA (ours)	**0.388**	**2.09**×

One RTX 4090, batch size one. Timing spans GPU-resident observations through vision, language, and ten flow updates to the complete 50-action chunk.

Figure 3: Joint allocation on $\pi_{0.5}$ from a common all-$K1$ policy. The error is normalized MSE over the first ten executed actions of a complete 50-action chunk. 100% uses the target 1.09-BPW static allocation's additional capacity; correction is off. Shading shows ± 1 standard deviation (SD) across ten random allocation configurations.

Starting from a common all-$K1$ policy, we compare joint allocations at increasing fractions of the additional capacity used at the target rate (Figure 3). All criteria use the same frozen reconstruction bank and storage cap, but independently determine their component quotas and stripe assignments. Both response-based criteria achieve lower normalized action error than calibrated local reconstruction and random allocation at every evaluated nonzero budget. The isotropic control retains the average sensitivity within each response-metric block while removing directional differences. Comparing the two response-based allocators therefore examines the value of candidate-error directions within calibrated global allocation. At the target budget, CARE-VLA reduces normalized action error by **7.63%** relative to calibrated isotropic scoring. A matched LIBERO evaluation likewise raises average success from **90.50% to 91.55%** (Table 17), a gain of 1.05 percentage points with a 95% bootstrap interval of $[0.65, 2.95]$ (Table 22). Together, these results connect response geometry to capacity use in the complete model and to closed-loop task performance. Correction is disabled throughout the allocation comparison; Section E.8 gives the protocol and complete values.

4.5 Efficiency

CARE-VLA reduces executed-linear-weight storage from 5.65 GB to 0.388 GB, including codec overhead, corresponding to $14.5\times$ compression (Table 4). The comparison covers the same weights across vision, language, action, and Extra; Section C.1 specifies the scope.

On a single NVIDIA RTX 4090 at batch size one, CARE-VLA reduces full-policy latency from 250.89 ms to 120.28 ms, corresponding to a $2.09\times$ measured speedup over the native BF16 baseline. Timing covers the complete 50-action policy forward, including vision, language, and ten flow updates. The compact inference representation retains quantized weights throughout execution without a dense quantized-weight cache.

5 Conclusion

We introduced CARE-VLA, a near-one-bit VLA vector-quantization framework that allocates capacity among fixed trellis reconstructions. Randomized pullbacks score concrete candidate errors in response geometry, while component interventions calibrate heterogeneous gains for shared-budget packing. Candidate-specific scoring captures both the magnitude and direction of reconstruction errors. The fixed bank separates candidate construction from response-guided capacity allocation. Policy-level correction accounts for interactions among selected reconstructions. Evaluations across three architectures demonstrate strong task retention, while matched controls and joint-allocation curves support response-guided capacity allocation. At the target budget, CARE-VLA improves action fidelity over calibrated isotropic scoring, with a corresponding gain in matched closed-loop success. The compact representation reduces storage and enables faster complete-policy inference. These results support allocating a shared model-wide budget to the concrete reconstructions that best preserve downstream responses across vision, language, and action under extreme quantization.

AI USE STATEMENT

No AI tools were used in this research.

ETHICS STATEMENT

This study uses existing simulation benchmarks and involves no human subjects or newly collected personal data.

REPRODUCIBILITY STATEMENT

The appendix provides derivations, algorithms, response interfaces, calibration settings, and evaluation protocols.

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. URL https://arxiv.org/abs/2605.24011.

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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025. URL https://www.roboticsproceedings.org/rss21/p010.html.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2307.13304.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://www.roboticsproceedings.org/rss19/p026.html.

Tim Dettmers, Ruslan 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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/1787533e171dcc8549cc2eb5a4840eec-Abstract-Conference.html.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12284–12303, 2024. URL https://proceedings.mlr.press/v235/egiazarian24a.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024. URL https://www.roboticsproceedings.org/rss20/p090.html.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. BiLLM: Pushing the limit of post-training quantization for LLMs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20023–20042, 2024. URL https://proceedings.mlr.press/v235/huang24q.html.

Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics—Simulation and Computation*, 18(3):1059–1076, 1989. doi: 10.1080/03610918908812806.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025a. URL https://www.roboticsproceedings.org/rss21/p017.html.

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

Sehoon Kim, Coleman Richard Charles Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23901–23923, 2024. URL https://proceedings.mlr.press/v235/kim24f.html.

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*, 2024. URL https://arxiv.org/abs/2411.19650.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, 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. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728, 2025. URL https://proceedings.mlr.press/v270/li25c.html.

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 *Proceedings of Machine Learning and Systems*, volume 6, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

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*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL https://www.jmlr.org/papers/v21/17-678.html.

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. doi: 10.48550/arXiv.2504.16054. URL https://arxiv.org/abs/2504.16054.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/c6483c8a68083af3383f91ee0dc6db95-Abstract-Conference.html.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48630–48656, 2024a. URL https://proceedings.mlr.press/v235/tseng24a.html.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. In *Advances in Neural Information Processing Systems*, volume 37, 2024b. URL https://papers.nips.cc/paper_files/paper/2024/hash/6de2e84b8da47bb2eb5e2ac96c63d2b0-Abstract-Conference.html.

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*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

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 *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fa064215307efaad75bebc7a2e3194a4-Abstract-Conference.html.

Yuzhuang Xu, Shiyu Ji, Qingfu Zhu, and Wanxiang Che. CRVQ: Channel-relaxed vector quantization for extreme compression of LLMs. *Transactions of the Association for Computational Linguistics*, 13:1488–1506, 2025. doi: 10.1162/tacl.a.45. URL https://aclanthology.org/2025.tacl-1.68/.

Xin Yan, Zhenglin Wan, Feiyang Ye, Xingrui Yu, Hangyu Du, Yang You, and Ivor Tsang. 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.

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. URL https://arxiv.org/abs/2602.20309.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Complete Derivations and Proofs

A.1 Response-distortion expansion

For a fixed context, write $r \in \mathbb{R}^q$ for a native response, $w \in \mathbb{R}^{n_w}$ for the flattened reference weights, $\delta \in \mathbb{R}^{n_w}$ for a weight perturbation, and $J = \partial r/\partial w \in \mathbb{R}^{q \times n_w}$ for its Jacobian. Here q is the response dimension and δ abbreviates δw from Eq. (3). Assume r is twice continuously differentiable near w with bounded second derivative. Taylor's theorem gives $r(w+\delta) - r(w) = J\delta + e(\delta)$, where the remainder $e(\delta) \in \mathbb{R}^q$ obeys $\|e(\delta)\|_2 \le C\|\delta\|_2^2$ for a finite constant C. For a fixed positive-semidefinite metric $M \in \mathbb{R}^{q \times q}$, abbreviating $e(\delta)$ as e gives

$$\|J\delta + e\|_M^2 = \delta^\top J^\top M J\delta + 2\delta^\top J^\top M e + e^\top M e. \tag{11}$$

The second and third terms are respectively $O\big((\|\delta\|_2)^3\big)$ and $O\big((\|\delta\|_2)^4\big)$ because $|a^\top M b| \le \|M\|_2\|a\|_2\|b\|_2$ for any $a, b \in \mathbb{R}^q$. Here $\|\cdot\|_2$ is the Euclidean norm for vectors and spectral norm for matrices. Taking expectations under integrable derivative bounds proves Eq. (3). For any weight-space vector $x \in \mathbb{R}^{n_w}$, $x^\top J^\top M J x = \|Jx\|_M^2 \ge 0$, establishing positive semidefiniteness. At the reference the residual is zero, so the Hessian of the squared discrepancy is $2J^\top M J$. Away from it, residual-weighted second derivatives also contribute.

A.2 Unbiased probe identity and variance

Proposition 1 (Unprojected pullback)**.** *Let q be the response dimension and $\xi \in \{-1, +1\}^q$ have independent equiprobable coordinates, so $\mathbb{E}[\xi\xi^\top] = I_q$, the q-dimensional identity. With scalar probe $\zeta = q^{-1/2}\xi^\top M^{1/2} r(w)$ and the symmetric square root of M,*

$$\mathbb{E}_\xi[gg^\top] = q^{-1} J^\top M J, \qquad g = \nabla_w \zeta \in \mathbb{R}^{n_w}. \tag{12}$$

Here g is the gradient with respect to flattened weights and $\mathbb{E}_\xi$ averages probe draws.

Proof. Reverse mode yields the weight gradient $g = \nabla_w \zeta = q^{-1/2} J^\top M^{1/2} \xi \in \mathbb{R}^{n_w}$. Multiplication and expectation give $\mathbb{E}[gg^\top] = q^{-1} J^\top M^{1/2} \mathbb{E}[\xi\xi^\top] M^{1/2} J = q^{-1} J^\top M J$. □

For a fixed perturbation δ, put $v = M^{1/2} J\delta$. A single probe estimates its normalized quadratic by $Q = (\xi^\top v)^2/q$. Since $\mathbb{E}[\xi_j^4] = 1$ and products with an odd occurrence of any coordinate have zero expectation, expanding the fourth moment yields

$$\mathrm{Var}(Q) = \frac{2}{q^2}\left(\|v\|_2^4 - \sum_j v_j^4\right) \le \frac{2}{q^2}\|v\|_2^4. \tag{13}$$

Here $v \in \mathbb{R}^q$, $j \in \{1, \dots, q\}$ indexes response coordinates, and Var is variance over probe draws. For N_p independent probes at the same context, the sample-mean variance is divided by N_p. Reused streams introduce covariance across contexts. The unbiased identity applies before token, row, and block projections or random trace normalization.

A.3 Input-block approximation

For $h = XW_m^\top$, let $X \in \mathbb{R}^{T \times d_{\text{in}}}$ contain T token inputs, $W_m, E \in \mathbb{R}^{d_o \times d_{\text{in}}}$ be a module's reference weight and its perturbation, and $U = \partial\zeta/\partial h \in \mathbb{R}^{T \times d_o}$. Using the Frobenius inner product $\langle A, B\rangle = \mathrm{tr}(A^\top B)$ for equal-sized matrices A, B, the directional derivative $\delta\zeta$ induced by E is

$$\delta\zeta = \langle U, XE^\top\rangle = \langle U^\top X, E\rangle. \tag{14}$$

The exact probe quadratic contains terms (with a, a' indexing output rows and j, j' input coordinates) $\sum_{a,j,a',j'} E_{aj} E_{a'j'} Z_{aj} Z_{a'j'}$, where $Z = U^\top X \in \mathbb{R}^{d_o \times d_{\text{in}}}$ is the probe weight gradient. Summing the gradient outer products over output rows yields the following input-coordinate interactions:

$$Z^\top Z = X^\top U U^\top X = \sum_{\tau,\tau'} \langle U_{\tau,:}, U_{\tau',:}\rangle X_{\tau,:}^\top X_{\tau',:}. \tag{15}$$

The indices $\tau, \tau' \in \{1, \dots, T\}$ select token rows; a colon selects all columns. Eq. (6) discards off-diagonal token terms; input blocking discards cross-block terms, and per-invocation accumulation omits interactions between separate calls. Positive semidefiniteness is preserved: $Z^\top Z$ is a Gram matrix, and $X^\top \operatorname{diag}(\omega) X$ uses the nonnegative token-weight vector $\omega \in \mathbb{R}^T$, with $\omega_\tau = \|U_{\tau,:}\|_2^2 / d_o$. Nonnegative weighted averages preserve this property.

A.4 Trace score, coordinate changes, and stripe additivity

For any positive-semidefinite input metric $G \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ and weight error $E \in \mathbb{R}^{d_o \times d_{\text{in}}}$,

$$\operatorname{tr}(EGE^\top) = \|EG^{1/2}\|_F^2. \tag{16}$$

Here $G^{1/2}$ is the symmetric square root of G, and $\|\cdot\|_F$ is the Frobenius norm. Thus the score is a squared reconstruction error in sensitivity-weighted coordinates. With orthogonal output and input transforms $L \in \mathbb{R}^{d_o \times d_o}$ and $R \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$, let $\widetilde{E} = LER$ and $\widetilde{G} = R^\top G R$ be the transformed error and metric. Then

$$\operatorname{tr}(\widetilde{E}\widetilde{G}\widetilde{E}^\top) = \operatorname{tr}(LERR^\top GRR^\top E^\top L^\top) = \operatorname{tr}(EGE^\top). \tag{17}$$

The output transform preserves the summed quadratic; the input metric must be in the matching domain. The implemented scorer avoids forming $\widetilde{G}$: it restores the input coordinates of the transformed error, $E^\star = \widetilde{E}R^\top = LE$, and evaluates $\operatorname{tr}(E^\star G (E^\star)^\top)$. Allocation stripes partition the rows of $E^\star$, retaining the output-side rotation. For disjoint row stripes E_i, $\operatorname{tr}(EGE^\top) = \sum_i \operatorname{tr}(E_i G E_i^\top)$. If $G = \operatorname{blockdiag}(G_b)$, where blockdiag places input-block metrics G_b on the diagonal and zeros elsewhere, stripe additivity also separates across input blocks, giving the sum over b used in Eq. (4). These identities apply to the quadratic surrogate.

A.5 Moment matching and shrinkage

Let $S_c = \sum_{i \in \mathcal{I}_c} \Delta D_i > 0$ and $\Delta \overline{L}_c = \overline{L}_c^{(1)} - \overline{L}_c^{(2)} > 0$ denote the aggregate native gain and measured action-discrepancy reduction for component c, respectively. Eq. (7) gives $\sum_{i \in \mathcal{I}_c} \kappa_c \Delta D_i = \Delta \overline{L}_c$. It matches the aggregate component gain. Individual stripes retain the relative ordering supplied by their native-response scores. The calibrated components have positive measured and aggregate native-response gains, so their logarithms are defined.

Proposition 2 (LogShrink invariants)**.** *For positive scales and $0 \le \alpha \le 1$, LogShrink preserves the geometric mean. For any components $c, c' \in \mathcal{C}$, $\log(\widetilde{\kappa}_c / \widetilde{\kappa}_{c'}) = \alpha \log(\kappa_c / \kappa_{c'})$. It is equivariant to multiplying all input scales by a common positive constant.*

Proof. Averaging Eq. (8) gives $\log \bar{\kappa}$. Subtracting the equations for c and c' cancels the common center. If every κ_c is multiplied by a common factor $a > 0$, both its log and the mean log increase by $\log a$, so every output is multiplied by a. □

For $0 < \alpha \le 1$, ordering is preserved and log-ratio magnitudes contract. At $\alpha = 0$ all components share the center. There is also a regularized interpretation. Let $l_c = \log \kappa_c$, $\bar{l} = \operatorname{avg}_c l_c$, and ϑ_c be a regularized log scale. For penalty strength $\lambda_s \ge 0$, minimizing

$$\sum_c (\vartheta_c - l_c)^2 + \lambda_s \sum_c (\vartheta_c - \bar{l})^2 \tag{18}$$

has derivative $2(\vartheta_c - l_c) + 2\lambda_s(\vartheta_c - \bar{l}) = 0$ and unique solution $\vartheta_c = (l_c + \lambda_s \bar{l})/(1 + \lambda_s)$. Thus $\alpha = 1/(1 + \lambda_s)$, linking LogShrink to regularization toward a shared component scale.

Component score rescaling. Let $t_c > 0$ rescale every native distortion gain in component c, so $\Delta D_i' = t_{c(i)} \Delta D_i$, while measured component action gains remain fixed. Eq. (7) then gives $\kappa_c' = \kappa_c / t_c$. Define $\bar{t} = (\prod_{c \in \mathcal{C}} t_c)^{1/|\mathcal{C}|}$. Substitution into Eq. (8) yields

$$\bar{\kappa}' = \bar{\kappa}/\bar{t}, \qquad \widetilde{\kappa}_c' = \widetilde{\kappa}_c t_c^{-\alpha} \bar{t}^{-(1-\alpha)}, \qquad \rho_i' = \rho_i \left(t_{c(i)}/\bar{t}\right)^{1-\alpha}. \tag{19}$$

Primes denote recomputation after rescaling; ρ_i is the gain per added bit from Eq. (9), and $\bar{t}$ is the geometric mean rescaling factor. The common $\bar{t}$ factor does not affect ranking. Relative component

densities retain a $t_c^{1-\alpha}$ dependence: exact cancellation holds at $\alpha = 1$, whereas shrinkage regularizes relative scales for $\alpha < 1$. With $\alpha = 7/8$, doubling one component's score changes its density relative to an unchanged component by $2^{1/8} \simeq 1.091$. Within-component stripe ordering is preserved. This characterizes LogShrink as stabilization, with the finite selector response measured in Table 18.

A.6 Feasibility and termination

At refinement iteration t, let P_t and M_t be sets of promoted and demoted stripe indices, and R_t the unused payload in bits. A promotion costs ΔB_i bits and a demotion releases the corresponding amount. Budget-feasible exchanges satisfy

$$\sum_{i \in P_t} \Delta B_i \le R_t + \sum_{j \in M_t} \Delta B_j. \tag{20}$$

Proposition 3 (Exchange feasibility)**.** *Let a current assignment have payload $B_t \le B_{\text{cap}}$ and slack $R_t = B_{\text{cap}} - B_t$, where $B_{\text{cap}} = B_{\max} - B_{\text{fixed}}$ is the available payload capacity in bits, $B_{\max}$ the total serialized bit budget, and B_{fixed} the fixed decoder overhead. Any exchange satisfying Eq. (20) has payload at most B_{cap}.*

Proof. Its new payload is $B_{t+1} = B_t + \sum_{i \in P_t} \Delta B_i - \sum_{j \in M_t} \Delta B_j \le B_t + R_t = B_{\text{cap}}$. Induction gives feasibility for the accepted sequence. If metadata depends on selectors, this payload argument alone is insufficient: the complete byte count is checked before acceptance and again for the selected final artifact. □

A strict objective decrease prevents revisiting a configuration because its measured objective is deterministic under fixed contexts and sampling states. The candidate space is finite, so accepted refinement terminates; the implementation also bounds the proposal sequence. Write $\mathcal{J}_t = \mathcal{J}(s^{(t)})$ for the objective at iteration t. A relative improvement gate with margin $\varepsilon_{\text{acc}} \in (0, 1)$ gives $\mathcal{J}_{t+1} \le (1 - \varepsilon_{\text{acc}})\mathcal{J}_t$ when $\mathcal{J}_t > 0$. At zero loss, strict improvement precludes accepting ties. This is a statement about the search objective.

A.7 Interactions beyond static scores

At a fixed context and sampling state, let $u \in \mathbb{R}^q$ be the current policy's response minus the reference response, and $v \in \mathbb{R}^q$ the proposed response minus the current response. Here q is the response dimension and $M \in \mathbb{R}^{q \times q}$ a fixed positive-semidefinite metric. Expanding the squared discrepancy gives the exact identity

$$\|u + v\|_M^2 - \|u\|_M^2 = 2u^\top M v + \|v\|_M^2. \tag{21}$$

The squared metric norm is $\|u\|_M^2 = u^\top M u$. The cross term $2u^\top M v$ can have either sign and depends on the current allocation. A stripe promotion can therefore change the complete-policy discrepancy differently from its frozen local score. Forward evaluation captures these finite interactions when accepting or rejecting a proposed configuration, without recomputing the proposal scores. Eq. (21) applies to each quadratic discrepancy term; the composite objective and its upper tail are evaluated directly as specified in Section C.3.

B Trellis Candidates and Allocation Algorithms

B.1 From weights to compatible trellis candidates

Trellis coding represents a weight vector by a path through a finite-state graph. We build on QTIP's bitshift trellis and blockwise quantization backend (Tseng et al., 2024b). All three architectures use the same Generic $K1/K2$ candidate construction described below.

Transform and grouping. A linear weight matrix $W_m \in \mathbb{R}^{d_o \times d_{\text{in}}}$ is mapped to signed Hadamard coordinates, $\widetilde{W}_m = L_m W_m R_m$, with the orthogonal factors defined in Eq. (25). Here m indexes the module, d_o and d_{in} are its output and input widths. Compatible candidates reuse the same transform and lookup table. Candidate construction processes 16×16 tiles; allocation selects an entire stripe of

16 output rows across the input width. Each tile is reordered by the backend's fixed permutation into a sequence of 128 two-dimensional vectors. This distinguishes the emitted vector dimension from both the tile size and the allocation granularity.

States, transition bits, and reconstruction. Let ℓ_{tr} be the state bit width, d_{v} the number of weights emitted per state, and K the index bits per weight. At vector position t, the integer state $q_t \in \{0, \dots, 2^{\ell_{\mathrm{tr}}} - 1\}$ emits $\mathcal{T}(q_t) \in \mathbb{R}^{d_{\mathrm{v}}}$, where $\mathcal{T}$ is the fixed state-to-vector lookup. A legal forward transition is

$$q_{t+1} = (2^{Kd_{\mathrm{v}}} q_t + \omega_t) \bmod 2^{\ell_{\mathrm{tr}}}, \qquad \omega_t \in \{0, \dots, 2^{Kd_{\mathrm{v}}} - 1\}. \tag{22}$$

Here ω_t is the integer represented by the next Kd_{v} input bits, and $\bmod$ keeps the lowest ℓ_{tr} bits after the shift. Each state has $2^{Kd_{\mathrm{v}}}$ successors. A path specifies a sequence of real-valued reconstruction vectors; the binary labels encode transitions, not binary weights.

Table 5: Compatible generic trellis candidates. Both rates use the same state-to-vector map and tile layout; increasing the transition word length changes the admissible paths. Index payload excludes shared decoding metadata.

Property	$K1$	$K2$
State bit width ℓ_{tr}	16	16
Emitted dimension d_{v}	2	2
Input bits per transition Kd_{v}	2	4
Successors per state $2^{Kd_{\mathrm{v}}}$	4	16
Shared compact lookup table	512×2 real entries	
Index payload per 16×16 tile	32 bytes	64 bytes

The full trellis has 2^{16} states per position. A deterministic state mapping and sign rule expand the shared compact lookup table into reconstruction vectors. The two rates differ in admissible paths rather than in the state-to-vector map.

Path search and error compensation. Viterbi dynamic programming minimizes summed vector reconstruction error over legal paths. A preliminary search on a cyclically shifted tile fixes the boundary overlap for the final path search, enabling cyclic packing. $K1$ and $K2$ are searched separately; the higher-rate reconstruction is a distinct path rather than a refinement of the lower-rate code.

Outside the tile search, QTIP's block LDL-decomposition-based quantization (LDLQ) uses the local activation Gram to compensate reconstruction error while processing input blocks from right to left. Consequently, each candidate is generated through a matrix-level quantization pass, not by independently fitting each allocation stripe. In the generic bank, this construction uses local reconstruction statistics; the native-response metric subsequently scores the resulting fixed errors. Candidate generation and response-guided selection are distinct operations.

Fixed scales and decoding. For this generic bank, the matrix scale is computed as $s_m = \mathrm{RMS}(\widetilde{W}_m)/(0.9\,\mathrm{RMS}(\mathcal{T}))$, where $s_m > 0$ is the normalization factor and RMS is the root mean square of the entries. Here $\mathcal{T}$ denotes the expanded state lookup. Quantization receives the normalized matrix, and reconstruction multiplies emitted values by s_m. Compatible $K1/K2$ candidates can therefore share a scale; scales are not refitted during allocation. The inverse tile permutation and orthogonal transforms recover weights in the original coordinates.

Cyclic packing stores Kd_{v} bits per emitted d_{v}-vector without a separate initial-state payload. Thus a tile uses $256K$ index bits and upgrading a stripe adds one bit per weight (Eq. (26)). The selected artifact stores one path per tile, stripe selectors, scales, and shared decoding metadata. Candidate construction and response-guided allocation are separated: the trellis backend fixes compatible $K1/K2$ reconstructions, while CARE-VLA evaluates these concrete errors in response space and assigns capacity under a shared budget.

B.2 COMPLETE QUANTIZATION PROCEDURE

Algorithm 1 Capacity Allocation via Response Evaluation

Input: Pretrained policy π_W, calibration contexts $\mathcal{D}_{\text{cal}}$, separate search and prefix-selection context sets $\mathcal{D}_{\text{sea}}, \mathcal{D}_{\text{val}}$, physical budget $B_{\max}$.
Output: Serialized quantized policy $\widehat{\pi}$ within $B_{\max}$, or selection failure.

Candidate and calibration phase:
Construct and fix the Generic $K1/K2$ bank, transforms, codebooks, and scales (Section B.1; Eq. (25)).
for each component $c \in \mathcal{C}$ **do**
  Select its native response or executed-interface local metric from Table 7.
  for each executed module m **do**
    Compute native-response probes using Eq. (5) where applicable.
    Accumulate input-block metrics (Eq. (6)); for Extra, use its executed-interface local Gram.
    Restore candidate-error input coordinates and score stripes using Eq. (4).
  end for
  Measure component interventions; check positive gains and calibrate with Eq. (7).
end for
Apply LogShrink using Eq. (8); freeze candidates and local scores.
Capacity allocation and refinement phase:
Start from all-$K1$ and construct the static anchor by density packing (Eq. (9)).
while the round limit is not reached and proposals remain **do**
  Form proposals from frozen gains and current selectors; discard budget violations (Eq. (20)).
  Evaluate each complete candidate policy on $\mathcal{D}_{\text{sea}}$ using Eq. (10).
  Apply the improvement and response gates; rank eligible proposals as in Section C.3.
  if an eligible proposal remains **then**
    Accept the highest-ranked proposal; refresh configuration and residual, keeping scores fixed.
  else keep the current configuration and stop the search.
  end if
end while
Apply anchor-relative guards on $\mathcal{D}_{\text{val}}$; select the eligible prefix minimizing Eq. (10).
If no prefix is eligible, return selection failure without exporting a refined artifact.
Serialize the selected policy and verify $B_{\max}$; return the selected quantized policy.

B.3 DUAL VERIFICATION OF THE BUDGETED ALLOCATION

The frozen $K1/K2$ bank defines a binary capacity-allocation problem. Let $x_i \in \{0,1\}$ select $K1$ when zero and $K2$ when one. Write $g_i = \widetilde{\kappa}_{c(i)} \Delta D_i$ for the calibrated signed gain and $F_0 = \sum_i \widetilde{\kappa}_{c(i)} D_{i,1}^{\text{pol}}$ for the all-$K1$ surrogate distortion. Let B_0 include all-$K1$ payload and fixed codec overhead, so $C_B = B_{\max} - B_0$ is the available additional capacity in bits. The budgeted problem is

$$\min_{x \in \{0,1\}^{N_s}} F(x) = F_0 - \sum_i g_i x_i, \qquad \text{s.t.} \quad \sum_i \Delta B_i x_i \le C_B. \tag{23}$$

Here N_s is the number of stripes, $x = (x_i)_{i=1}^{N_s}$ is the selector vector, F is the calibrated surrogate objective, and ΔB_i is the additional index payload from Eq. (26). The evaluated bank has fixed-size selectors and metadata, so the payload constraint gives the complete serialized cap. Positive-gain promotions have density $g_i/\Delta B_i$, as in Eq. (9); nonpositive gains need no added capacity.

For a nonnegative budget multiplier λ_B, minimizing the Lagrangian over each binary choice independently gives the dual lower bound

$$\Phi(\lambda_B) = F_0 - \lambda_B C_B + \sum_i \min\{0, \lambda_B \Delta B_i - g_i\}, \qquad \max_{\lambda_B \ge 0} \Phi(\lambda_B). \tag{24}$$

The scalar λ_B prices an added bit and Φ denotes the dual objective. For every feasible x, $\Phi(\lambda_B) \leq F(x)$: the Lagrangian is a lower bound on $F(x)$ because the budget residual is nonpositive. A feasible assignment attaining this bound is optimal for the frozen surrogate.

On the evaluated $\pi_{0.5}$ instance, density packing and the dual solution select the same stripes and saturate the serialized cap (Table 6). A multiplier between the lowest selected and highest excluded gain densities certifies the solution. The feasible objective and lower bound differ by approximately 1.1×10^{-16}.

Table 6: Budgeted-allocation verification on the frozen $\pi_{0.5}$ instance. Codec size includes overhead; selector agreement is measured against density packing.

Method	Surrogate ↓	Codec size	Selector agreement
Density packing	0.4608	388.35 MB	100%
Dual solution	0.4608	388.35 MB	100%

The matching feasible objective reaches the dual lower bound within numerical precision.

C Experimental Details

C.1 Quantized representation

Section B.1 defines the trellis candidates. Here we specify their coordinates, quantization scope, and calibration settings.

For module m, the orthogonal preprocessing is

$$\widetilde{W}_m = L_m W_m R_m, \qquad L_m^\top L_m = I, \quad R_m^\top R_m = I. \tag{25}$$

Here $L_m \in \mathbb{R}^{d_o \times d_o}$ and $R_m \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ are signed output and input transforms, I is the identity of matching size, and $\widetilde{W}_m$ is the transformed weight. Errors and metrics must use the same coordinate system, as shown in Eq. (17). If $B_{i,K1}$ and $B_{i,K2}$ are a stripe's index payloads in bits, then

$$\Delta B_i = B_{i,K2} - B_{i,K1} = n_i, \tag{26}$$

where n_i is its number of weights. This difference accounts for indices; fixed and selector-dependent metadata are included when checking the serialized budget.

Quantization covers executed linear matrices in vision, language, action, and Extra, which includes projectors and adapters. Each stripe selects a one- or two-bit trellis reconstruction. Transforms, codebooks, and precomputed scales remain fixed during allocation and correction. The selected representation contains no floating-point residual branch and requires no calibration data at inference.

Storage accounting. Executed-linear BPW divides serialized codec bits, including overhead, by the number of executed linear weights. The FP storage baseline represents these weights at 16-bit precision; the quantized representation includes indices, selectors, scales, and decoder metadata. The storage comparison in Table 4 covers the same matrices across vision, language, action, and Extra. Retained-precision tensors and model configuration assets bring complete model storage to 1.974 GB.

C.2 Native responses and metric construction

For $\pi_{0.5}$, vision uses projected visual tokens consumed by the language backbone. The language response comprises its layerwise prefix key/value states. The action response is the first ten actions of the final generated chunk, with the physical normalization used for discrepancy evaluation. Visual features are energy-normalized within the differentiated response map; language key/value tensors are normalized separately with equal layer weights. These choices are incorporated in the response definition and metric M_c in Eq. (2). Eight shared Rademacher streams cycle across 128 geometry contexts, with one projected vector–Jacobian product (VJP) per context. Calibrated scores and allocations are stable across one to eight shared streams, with score correlations above 0.996 and selector Jaccard above 0.950 (Section E.12).

Table 7: Implementation across three architectures. All use token-diagonal response scoring and the final executed action prefix as the action response. The input metric is shared across output rows. Extra includes projectors and adapters and uses its executed-interface local metric.

Stage	$\pi_{0.5}$	OpenVLA-OFT	CogACT
Input surrogate	Token-diagonal, Eq. (6)		
Response metric	Native-response blocks		
Action interface	Final executed action prefix		
LogShrink α	0.875	0.9375	0.9375
Calibration source	LIBERO	LIBERO	SIMPLER
Geometry contexts	128	128 per suite	128
Response probes	One per context; eight fixed streams	Two per context	Two per context
Component calibration	Complete-action K1/K2 interventions		
Candidate family	Generic K1/K2		
Static allocator	Calibrated gain-per-bit packing		
Refinement	Complete-policy feasible exchanges		
Prefix selection	Anchor-relative response guards		

Final-action interface. The action response is the final executed action prefix for $\pi_{0.5}$, OpenVLA-OFT, and CogACT. Each policy completes its native action-generation procedure before the executed prefix is selected and normalized. For the iterative generators, differentiation follows the generation trajectory to this final response. Reference and quantized evaluations share the same initial action noise and sampling randomness. OpenVLA-OFT uses its continuous action output with the corresponding observation and proprioceptive conventions. CogACT uses the final generated actions, with calibration observations collected from SIMPLER. The evaluated prefix length and physical action normalization follow the policy's execution interface.

C.3 Complete-policy action discrepancy

Let $a_{t,j}$ and $\widehat{a}_{t,j}$ be reference and quantized physical actions, where $t \in \{1, \dots, H_e\}$ indexes the evaluated prefix of length $H_e \geq 2$ and $j \in \{1, \dots, d_a\}$ the action coordinate. Let $\sigma_j > 0$ be a frozen reference root-mean-square (RMS) scale. Define normalized error $e_{t,j} = (\widehat{a}_{t,j} - a_{t,j})/\sigma_j$ and the following dimension-averaged quantities:

$$L_{\text{act}} = \frac{1}{H_e d_a} \sum_{t=1}^{H_e} \sum_{j=1}^{d_a} e_{t,j}^2, \tag{27}$$

$$L_{\text{temp}} = \frac{1}{(H_e - 1) d_a} \sum_{t=1}^{H_e - 1} \sum_{j=1}^{d_a} (e_{t+1,j} - e_{t,j})^2, \tag{28}$$

$$L_{\text{pre}} = \frac{1}{3H_e} \sum_{t=1}^{H_e} \left\| \frac{1}{t} \sum_{\tau=1}^{t} e_{\tau,1:3} \right\|_2^2, \tag{29}$$

$$L_{\text{grip}} = \frac{1}{H_e} \sum_{t=1}^{H_e} \mathbf{1}\left[\widehat{b}_t \neq b_t\right], \tag{30}$$

$$\mathcal{E} = L_{\text{act}} + \lambda_{\text{temp}} L_{\text{temp}} + \lambda_{\text{pre}} L_{\text{pre}} + \lambda_{\text{grip}} L_{\text{grip}}. \tag{31}$$

Here L_{act} measures normalized action error, L_{temp} its change between successive steps, L_{pre} cumulative translation-action error, and L_{grip} gripper disagreement. The scalar $\mathcal{E}$ combines them at one context; context and configuration arguments are suppressed. The index τ runs over preceding action steps, and $e_{\tau,1:3} \in \mathbb{R}^3$ selects their translation coordinates. The binary variables b_t and $\widehat{b}_t$ are reference and quantized gripper decisions; $\mathbf{1}[\cdot]$ is one when its condition holds and zero otherwise. For $\pi_{0.5}$ and OFT, $b_t = \mathbf{1}[a_{t,g} \geq 0]$, where g indexes the physical gripper coordinate. CogACT thresholds its normalized gripper output at 0.5. The same rule defines $\widehat{b}_t$ from the quantized output. The prefix term uses the cumulative *mean* normalized translation-action error, including the factor $1/t$. The evaluated horizons are $H_e = 10$, 8, and 16 for $\pi_{0.5}$, OFT, and CogACT,

respectively. The nonnegative weights $(\lambda_{\text{temp}}, \lambda_{\text{pre}}, \lambda_{\text{grip}})$ are $(1/8, 1/16, 1/16)$ for $\pi_{0.5}$ and OFT, and $(1/16, 1/16, 1/16)$ for CogACT. The discrete gripper term is evaluated in forward comparisons, not differentiated for pullback estimation.

For component calibration, $\overline{L}_c^{(k)}$ averages L_{act} when component c uses candidate rate $k \in \{1, 2\}$, with the other components at full precision. Refinement uses $\mathcal{E}$, with $\beta = 0.25$ and $\gamma = 0.9$ in Eq. (10). These coefficients control the weight and extent of the upper tail; the temporal, prefix, and gripper coefficients control the corresponding discrepancies. They remain fixed throughout proposal search and prefix selection.

Calibration domain. Calibration assumes $\overline{L}_c^{(1)} > \overline{L}_c^{(2)} \geq 0$ and $S_c = \sum_{i \in \mathcal{I}_c} \Delta D_i > 0$, satisfied by all four components. The denominator retains signed gains; the positive-part operation applies only to allocation in Eq. (9).

Bounded proposal generation. Let N be the number of quantized weights. Each round forms one proposal per component by promoting $K1$ stripes in descending frozen density ρ_i, up to $\lfloor N/512 \rfloor$ weights; $\lfloor \cdot \rfloor$ rounds downward. When unused capacity is insufficient, other components supply $K2$ donors in ascending density. Proposals must satisfy the serialized budget and remain within $\lfloor N/32 \rfloor$ changed weights of the anchor. Search evaluates at most four proposals per round for eight rounds, with deterministic tie-breaking.

Search acceptance. For $\pi_{0.5}$, a proposal on $\mathcal{D}_{\text{sea}}$ must reduce $\mathcal{J}$ by a relative margin $\varepsilon_{\text{acc}} = 1/512$. The mean and $\text{CVaR}_{0.9}$ of L_{act} must remain within 1.005 and 1.01 times their current values; mean L_{grip} may increase by at most $1/128$. The complete serialized artifact must fit the budget. Define the changed-weight distance from the static anchor s^A and proposal priority by

$$d_A(s) = \sum_i n_i \mathbf{1}[s_i \neq s_i^A], \qquad \Gamma(s' \mid s) = \frac{\mathcal{J}(s) - \mathcal{J}(s')}{\max\{1, d_A(s') - d_A(s)\}}. \tag{32}$$

Here s is the current configuration, s' a proposal, s_i^A the anchor's stripe choice, and n_i its weight count. The indicator weights each changed stripe by its size. The denominator measures additional distance from the anchor, floored at one. The eligible proposal with largest Γ is accepted, with ties favoring lower $\mathcal{J}$ and then smaller d_A. Search ends when no proposal qualifies or the round limit is reached. All tolerances remain fixed independently of closed-loop evaluation.

Prefix selection and data roles. For $\pi_{0.5}$, geometry uses 128 calibration contexts; search and prefix selection each use 64 further contexts. Each context comes from a distinct episode, and the three sets are episode-disjoint. Geometry and component scales are fixed before search. On $\mathcal{D}_{\text{val}}$, an eligible prefix has $\mathcal{J}$ no larger than the anchor's, mean and tail L_{act} within factors 1.01 and 1.02 of the anchor, and mean L_{grip} within $1/128$ of it. Selection minimizes $\mathcal{J}$ among eligible prefixes, breaking ties by d_A; it returns no refined configuration if none qualifies. Separate closed-loop episodes evaluate task success.

Scope. Calibration coverage and native interfaces determine which policy sensitivities inform allocation. Physical rate measures storage; inference speed additionally depends on decoding and kernel support.

C.4 Computational cost

Table 8 summarizes one-time offline quantization costs on $\pi_{0.5}$ using NVIDIA RTX 4090 GPUs. The three stages total 4.78 GPU-hours across four workers; candidate construction accounts for 86%. The subtotal excludes model loading, local-statistic preparation, CPU allocation and serialization, and closed-loop evaluation.

Table 8: Recorded PTQ stage costs on $\pi_{0.5}$ using NVIDIA RTX 4090 GPUs, summed over workers.

Stage	GPU time
Trellis-vector candidate construction	4.13 h
Response estimation, scoring, and calibration	27.61 min
Refinement and prefix-selection forwards	11.47 min
Recorded-stage subtotal	4.78 h

D Additional Evaluation Results

D.1 SIMPLER category results

Table 9 gives the category-level results corresponding to Table 2. Visual matching and variant aggregation are evaluated separately.

Table 9: CogACT success by SIMPLER category (%). VM and VA denote visual matching and variant aggregation. Open/close drawer groups the drawer tasks; place apple denotes placement of an apple in the drawer. Operating points match Table 2.

Setting	Task category	QTIP	QVLA	CARE-VLA
VM	Pick coke	67.3	0.0	88.6
	Move near	58.4	0.0	83.1
	Open/close drawer	45.9	0.0	69.7
	Place apple	26.8	0.0	37.4
VA	Pick coke	59.7	0.0	83.9
	Move near	55.3	0.0	76.4
	Open/close drawer	10.2	0.0	25.6
	Place apple	11.6	0.0	26.8

D.2 Two-bit reconstruction comparison

Table 10 isolates the vision-language model (VLM) backbone of $\pi_{0.5}$ at two-bit weight precision, with the remaining components unchanged. Round-to-nearest (RTN) quantization yields no successful episodes, GPTQ (Frantar et al., 2023) recovers 57.0%, and QTIP (Tseng et al., 2024b) reaches 98.0%. This restricted-scope comparison illustrates the effect of the reconstruction method at a fixed nominal precision.

Table 10: Two-bit VLM-backbone diagnostic on $\pi_{0.5}$ / LIBERO-Goal. Only the VLM backbone weights are quantized; the other components retain their original precision. SR denotes task success rate.

VLM-only W2 quantizer	SR (%)
RTN	0.0
GPTQ	57.0
QTIP	98.0

E Complete Ablations on $\pi_{0.5}$

E.1 Scoring, calibration, and refinement

Native-response scoring and cross-component calibration account for the main improvement over layer reconstruction. LogShrink produces a static allocation with 91.1% average success, and complete-policy refinement with prefix selection reaches 91.4%. Restricting exchanges to action or vision gives 90.5% and 90.4%, respectively. Prefix selection raises the aggregate from the unselected endpoint's 90.7% to 91.4%, while individual suites respond differently to the final configuration.

Table 11: Complete $\pi_{0.5}$ ablations on LIBERO (%), using the same evaluation bank and 500 episodes per suite. The upper block follows the cumulative construction in Table 3. The lower block compares refinement and selection choices starting from the calibrated static allocation. BPW includes scoped serialization overhead.

Variant	BPW	Spatial	Object	Goal	Long	Average
Uniform one-bit	1.00	82.0	88.4	69.8	15.6	64.0
Local allocation (uncalibrated)	1.09	76.8	78.6	70.2	12.6	59.6
Response scoring (uncalibrated)	1.09	95.0	98.6	92.0	63.0	87.2
Calibrated allocation, no shrinkage	1.09	96.2	99.4	89.0	78.6	90.8
CARE-VLA (static)	1.09	97.6	98.4	91.2	77.2	91.1
Action-only single-step exchange	1.09	95.6	98.8	91.0	76.6	90.5
Vision-only single-step exchange	1.09	95.8	99.0	89.4	77.4	90.4
Without prefix selection	1.09	96.0	100.0	91.2	75.6	90.7
CARE-VLA	1.09	96.8	98.6	92.2	78.0	91.4

E.2 Static rate and calibration-scale studies

Table 12: Component-scale calibration with fixed candidate geometry and scoring contexts. Entries are $\pi_{0.5}$ LIBERO average success (%) at 500 episodes per suite. Only the contexts used to estimate component scales vary, not the complete PTQ data budget. The difference is LogShrink minus no shrinkage, in percentage points.

Scale-calibration contexts	16	32	64	128
Without shrinkage	89.9	87.7	90.8	90.8
With LogShrink	90.0	89.8	89.9	91.1
Difference	+0.2	+2.1	−1.0	+0.3

LogShrink regularizes cross-component scale dispersion while preserving within-component ranking; the default 128-context setting achieves 91.1% average success (Table 12).

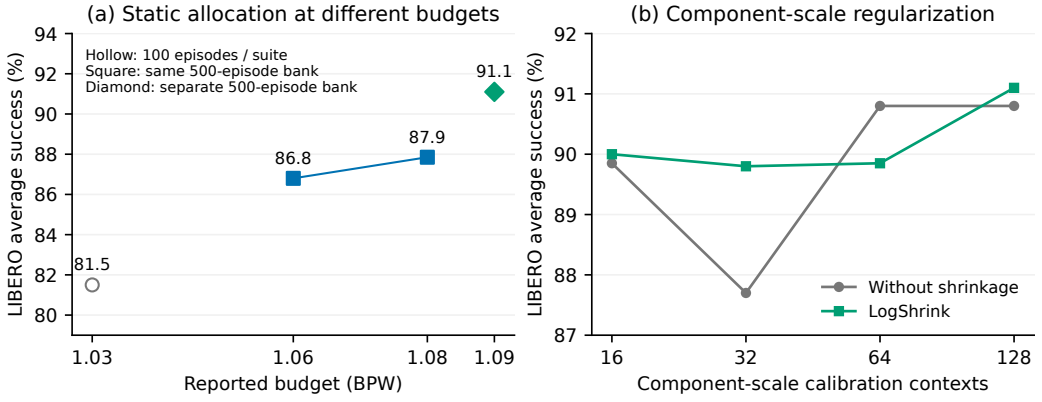


Figure 4: Static-allocation analyses on $\pi_{0.5}$. (a) Success at the supplied rate budgets. Blue squares share a 500-episode-per-suite bank and are connected; the green diamond uses a different bank. The hollow gray circle is an earlier 100-episode-per-suite evaluation, shown separately from the larger evaluations without joining unmatched banks. (b) Varying contexts for component-scale calibration, with geometry fixed. Green squares use LogShrink; gray circles do not. Lines connect supplied settings and are not fitted curves or uncertainty estimates.

Table 13: Underlying static-rate results for Figure 4(a). Entries are success percentages. Evaluation banks and episode counts distinguish the studies.

Budget	Bank	Episodes/suite	Spatial	Object	Goal	Long	Average
≈1.03	20001	100	90.0	99.0	84.0	53.0	81.5
1.06	21001	500	93.8	99.6	87.6	66.2	86.8
1.08	21001	500	95.2	98.4	87.2	70.6	87.9
≈1.09	17001	500	97.6	98.4	91.2	77.2	91.1

Mechanism controls. The following controls use the frozen $\pi_{0.5}$ candidate bank. Unless specified otherwise, offline measurements use the 128 independent contexts described in Section F.2, with shared reference actions and action-sampling states across configurations. The joint-allocation and correction studies in Sections E.8 and E.10 use a separate independent set. Lower $\mathcal{J}$ indicates better policy-response preservation.

E.3 Fixed-component-quota stripe shuffle

We randomize which stripes receive $K2$ while preserving the static anchor's component quotas. Sampling is stratified by stripe weight count within each component, keeping the candidate bank, index payload, and complete codec size fixed.

Table 14: Fixed-quota stripe shuffle on $\pi_{0.5}$. All rows preserve the anchor's component quotas and codec size. Jaccard measures overlap between two-bit stripe selections; $\mathcal{J}$ measures offline response distortion.

Allocation	Selector Jaccard	$\mathcal{J}\downarrow$
Static anchor	1.000	0.8370
Within-component shuffle 1	0.267	1.0581
Within-component shuffle 2	0.268	1.0399
Within-component shuffle 3	0.265	1.1182

Shuffling raises the response objective by 24.2–33.6% relative to the static anchor, isolating the contribution of stripe identity beyond component capacity.

We select the shuffle with median offline $\mathcal{J}$ before closed-loop evaluation. The anchor and shuffle share 500 episodes per suite, 256×256 observations, and ten executed actions per chunk. Among paired outcomes, 542 succeed only under the anchor and 37 only under the shuffle. Table 3 reports all four suites.

E.4 Native-response scoring under matched component calibration

The local-reconstruction control replaces the native-response metric in Eq. (4) with the activation Gram. It retains the measured complete-policy action-MSE reduction in the calibration numerator and uses its own signed aggregate gain in the denominator. Both criteria thus receive the same component-calibration procedure before density packing.

Candidate reconstructions, component interventions, and the budget are shared. Both criteria use $\alpha = 1$ (unshrunk) or $\alpha = 0.875$ (default LogShrink), with independently allocated component quotas and no correction. The four policies share 500 matched LIBERO episodes per suite.

Native-response scoring improves every suite at both matched settings (Table 15), raising average success by 28.9 points under default LogShrink. Component calibration establishes a common scale; candidate-level geometry identifies which reconstructions preserve task behavior.

E.5 Candidate-error direction and global allocation

To isolate error direction, we replace $G_{m,b}$ by the trace-matched isotropic metric $[\mathrm{tr}(G_{m,b})/d_b]I_{d_b}$, where I_{d_b} is the d_b-dimensional identity. The control retains mean block sensitivity and error magnitude while removing directional preferences.

Table 15: Native-response versus local-reconstruction scoring with matched component calibration on $\pi_{0.5}$ (success, %). Both criteria use the same candidate bank and serialized budget, displayed as 1.09 BPW. Each block fixes the LogShrink exponent α and compares the scoring criteria. Average weights suites equally; bold marks the highest score within each block.

Scoring criterion	α	Spatial	Object	Goal	Long	Average
Local reconstruction	1	84.4	84.0	76.8	21.4	66.7
Native response	1	**96.2**	**99.4**	**89.0**	**78.6**	**90.8**
Local reconstruction	0.875	85.4	74.0	75.6	13.8	62.2
Native response	0.875	**97.6**	**98.4**	**91.2**	**77.2**	**91.1**

Fixed-quota controls. Two controls preserve $K2$ quotas at the component or weight-matrix level. Stratification by stripe weight count preserves codec size; candidates, transforms, codebooks, and scales remain fixed, with correction disabled. For Extra, isotropization acts on its local-input metric.

Table 16: Direction-aware versus trace-matched isotropic scoring on $\pi_{0.5}$ / LIBERO (success, %). All rows use the static candidate bank and the same serialized budget, with 500 episodes per suite. Quotas specify the level at which $K2$ counts are fixed. Average weights the four suites equally. Bold marks the highest value in each column.

Allocation	Spatial	Object	Goal	Long	Average
Direction-aware static	**97.6**	98.4	**91.2**	**77.2**	**91.1**
Isotropic, component quotas	95.8	**99.8**	89.0	76.2	90.2
Isotropic, matrix quotas	95.8	99.2	90.0	74.4	89.9

Direction-aware scoring improves average success under both fixed-quota controls (Table 16), supporting the use of candidate-error geometry for stripe selection.

Independent global allocation. Native-response and isotropic allocators each recompute κ_c using their own signed gains and determine global component quotas. Both use $\alpha = 0.875$, the same bank and byte cap, and no correction. A separate matched evaluation with 500 LIBERO episodes per suite yields 91.55% average success for native-response allocation and 90.50% for isotropic allocation (Table 17). The paired gain is 1.05 percentage points, with a 95% bootstrap interval of $[0.65, 2.95]$ entirely above zero (Table 22).

Table 17: Independent global allocation on $\pi_{0.5}$ / LIBERO (success, %). Each criterion has its own component calibration and global quotas at the target 1.09-BPW budget, with $\alpha = 0.875$ and correction disabled. Bold marks the higher value within this matched comparison.

Allocation	Spatial	Object	Goal	Long	Average
Calibrated isotropic	95.6	**99.2**	91.0	76.2	90.50
CARE-VLA (static)	**97.6**	98.4	**91.2**	**79.0**	**91.55**

E.6 Component score rescaling

We rescale one component's frozen gains at a time by $t_c \in \{0.5, 2\}$ and recompute calibration, LogShrink, and allocation. Other factors remain one, with the bank and budget fixed. The identity $t_c = 1$ reproduces the original selector; Eq. (19) gives the residual $t_c^{1-\alpha}$ dependence at $\alpha = 0.875$.

The two-bit selector Jaccard remains between 0.954 and 0.990. The response objective ranges from 0.8242 to 0.8483, within 1.6% of the unperturbed value. These finite perturbations support local scale stability: LogShrink attenuates component-scale changes while retaining their effect on allocation.

Table 18: LogShrink rescaling check. Quotas give the number of two-bit stripes in Vision, Language, Action, and Extra, respectively. Jaccard compares the resulting selector with the static anchor.

Rescaled component	t_c	Vision	Language	Action	Extra	Jaccard	$\mathcal{J}$
None	1	8936	1861	1503	1039	1.000	0.8370
Vision	0.5	8610	1944	1658	1103	0.954	0.8351
Vision	2	9218	1766	1384	985	0.960	0.8392
Language	0.5	9069	1732	1555	1056	0.976	0.8396
Language	2	8805	1966	1484	1026	0.980	0.8350
Action	0.5	9038	1875	1356	1052	0.980	0.8483
Action	2	8807	1839	1702	1026	0.973	0.8273
Extra	0.5	8979	1863	1520	957	0.989	0.8242
Extra	2	8895	1856	1494	1120	0.990	0.8365

E.7 Candidate Ranking around the Full-Precision Reference

We directly evaluate whether predicted candidate gains rank finite changes in executed actions around the full-precision reference. We sample 96 stripes across Vision, Language, Action, and Extra, with eight stripes from each component's low-, medium-, and high-native-gain strata. For each stripe, we separately insert its frozen $K1$ and $K2$ reconstructions into the full-precision model in the transformed weight coordinates, leaving all other stripes unchanged. We measure the reduction in physical-RMS-normalized action MSE on the final ten-step executed prefix using 32 fixed contexts, balanced across the four LIBERO suites, and matched action-generation randomness.

Table 19: Candidate-level prediction on $\pi_{0.5}$ around the full-precision reference. Spearman correlation compares predicted gains with measured executed-action MSE reductions across 96 stripes. Top-24 overlap is the fraction of the 24 largest measured gains recovered by each score; higher is better for both metrics.

Prediction criterion	Spearman ρ	Top-24 overlap (%)
Weight-error gain	0.134	37.5
Local-reconstruction gain	0.079	25.0
Isotropic response gain	0.367	41.7
Native-response gain	0.400	50.0
Calibrated isotropic gain	0.648	54.2
Calibrated native-response gain	**0.658**	**58.3**

Native-response scoring improves reference-background ranking over weight error and local reconstruction (Table 19). Criterion-specific calibration with $\alpha = 0.875$ strengthens both native and isotropic ranking; their top-24 overlaps recover 14 and 13 measured gains. Section E.8 tests their complete-policy allocations.

E.8 Whole-model Joint Allocation from an All-One-Bit Policy

We compare joint upgrades from a common all-$K1$ policy across vision, language, action, and Extra. Native-response, trace-matched isotropic, and local-reconstruction scoring share the candidate bank, transforms, scales, and serialized byte cap. Each criterion recomputes its own component calibration using the signed denominator in Eq. (7), applies LogShrink with $\alpha = 0.875$, and allocates globally without fixed component quotas. Random allocation packs stripes using random priorities; ten independently generated configurations are evaluated at each nonzero budget. Policy-level correction is disabled throughout.

Contexts and action metric. We use 128 independent demonstration-episode contexts, 32 from each LIBERO suite, separate from calibration, search, and prefix selection. All policies share reference actions and action-generation randomness. Each forward generates the complete 50-action chunk with ten flow updates; the metric is the physical-RMS-normalized MSE L_{act} over its first ten executed actions, as defined in Section C.3. We average contexts within each suite and weight the

four suites equally. The normalization is fixed across methods and budgets. Table 20 supplies the values plotted in Figure 3.

Table 20: Whole-model joint allocation on $\pi_{0.5}$. Normalized action error (↓) at measured fractions of the target additional capacity. Random reports the mean and sample SD across ten allocation configurations. All methods share the same 0% all-$K1$ policy.

Capacity	CARE-VLA (static)	Calibrated isotropic	Calibrated local	Random
0%	0.624070	0.624070	0.624070	0.624070 ± 0.000000
25%	0.493580	0.491057	0.592022	0.601158 ± 0.010758
50%	0.485417	0.474929	0.577242	0.593527 ± 0.018556
100%	0.435639	0.471637	0.576163	0.576309 ± 0.015178

Capacity definition. Let $C_0 = 353{,}393{,}063$ bytes denote the all-$K1$ codec and $C_\star = 388{,}354{,}247$ bytes the target static codec, including metadata. At fraction $f \in \{0, 0.25, 0.5, 1\}$, every method uses the cap $C(f) = C_0 + f(C_\star - C_0)$, where $C_\star - C_0 = 34{,}961{,}184$ bytes is the available additional capacity. Thus 100% corresponds to the target 1.09-BPW static allocation. The selected stripes and component quotas are independently determined by each criterion. Table 21 gives the common caps and actual codec sizes; discrete stripe sizes leave at most 1,824 bytes unused.

Table 21: Exact byte budgets for joint allocation. The common 0% codec is 353,393,063 bytes. Random gives the minimum–maximum codec size over ten configurations at each budget.

Codec bytes	25% capacity	50% capacity	100% capacity
Common cap	362,133,359	370,873,655	388,354,247
CARE-VLA (static)	362,133,159	370,872,423	388,354,247
Calibrated isotropic	362,131,879	370,873,447	388,352,423
Calibrated local	362,131,623	370,872,999	388,353,959
Random minimum	362,133,319	370,873,607	388,354,215
Random maximum	362,133,351	370,873,639	388,354,247

Paired comparisons. Table 22 collects the paired statistics for joint allocation, matched closed-loop evaluation, and independent correction evaluation. Offline intervals use 10,000 suite/task-stratified paired episode bootstrap draws, with one context per episode and fixed within-suite task weights. CVaR is recomputed in each draw. Random comparisons first average the ten fixed configurations for each context. Closed-loop intervals use suite-stratified task-cluster resampling with paired episodes within each sampled task. These intervals describe evaluation variability for the fixed policies.

E.9 Single-component Subset Selection

Starting from one component's all-$K1$ reconstruction, we compare ranked and random upgrades at 25%, 50%, and 100% of its static $K2$ quota. Other components remain at full precision. Stripe-weight-count matching preserves payload; candidates and calibration factors remain fixed.

We measure normalized action MSE over the first ten executed actions using 32 independent contexts (eight per suite; Section F.2) and matched randomness. Positive reductions from the component's all-$K1$ state indicate improved fidelity. Random averages two draws. This component-isolated study complements Section E.8.

Ranked selection gives larger reductions than random selection in all four components at each evaluated quota (25%, 50%, and 100%; Table 23). Positive component calibration preserves within-component ordering, so this experiment examines subset selection; Section E.8 evaluates the resulting complete-policy allocation.

E.10 Independent Evaluation of Policy-Level Correction

We reevaluate the originally selected correction prefix on the independent contexts from Section E.8, keeping the selection fixed. It reduces $\mathcal{J}$ from 0.899113 to 0.880262, a 2.10% reduction relative to the

Table 22: Paired differences with 95% bootstrap intervals. Positive differences favor native-response allocation or the selected correction prefix. Offline action-error differences are comparator minus native; the closed-loop difference is native minus isotropic success in percentage points. The correction difference is $\mathcal{J}(\text{anchor}) - \mathcal{J}(\text{selected})$.

Comparison	Capacity	Difference	95% interval
Whole-model action error			
Calibrated isotropic	25%	−0.002522	[−0.018511, 0.013882]
Calibrated isotropic	50%	−0.010488	[−0.029264, 0.007410]
Calibrated isotropic	100%	0.035998	[0.019988, 0.052739]
Calibrated local	25%	0.098442	[0.061756, 0.136821]
Calibrated local	50%	0.091825	[0.054098, 0.130110]
Calibrated local	100%	0.140524	[0.096479, 0.185508]
Random mean	25%	0.107579	[0.072017, 0.144908]
Random mean	50%	0.108110	[0.073990, 0.143436]
Random mean	100%	0.140670	[0.101250, 0.181337]
Matched LIBERO success (points)			
Native versus isotropic	100%	1.05	[0.65, 2.95]
Independent correction objective			
Anchor versus selected prefix	100%	0.018851	[0.000410, 0.041064]

Table 23: Partial-subset selection on $\pi_{0.5}$: normalized executed-action MSE reduction (↑) at fractions of the static component quota. Ranked and random subsets have matching index payloads; other components remain at full precision. Random averages two draws. Bold marks the larger reduction within each component–quota pair.

	25% quota		50% quota		100% quota	
Component	Ranked	Random	Ranked	Random	Ranked	Random
Vision	**0.0178**	0.0069	**0.0471**	0.0251	**0.0731**	0.0442
Language	**0.0565**	−0.0067	**0.0810**	0.0122	**0.1033**	0.0072
Action	**0.0039**	0.0005	**0.0079**	0.0013	**0.0129**	0.0025
Extra	**0.0176**	0.0073	**0.0225**	0.0103	**0.0282**	0.0226

static anchor. The paired reduction of 0.018851 has a 95% bootstrap interval of $[0.000410, 0.041064]$ entirely above zero (Table 22). Table 24 retains all discrepancy terms.

Table 24: Policy-level correction on independent contexts. All losses use the definitions in Section C.3; lower is better. $\mathrm{CVaR}_{0.9}$ averages the largest 10% of context losses. The selected prefix is fixed by the original selection set.

Metric	Static anchor	Selected prefix
Action error L_{act}	0.435639	0.428068
Temporal error L_{temp}	0.024596	0.024332
Translation-prefix error L_{pre}	0.358801	0.347370
Gripper disagreement L_{grip}	0.145312	0.142188
Mean composite response mean $\mathcal{E}$	0.470220	0.461707
Action tail $\mathrm{CVaR}_{0.9}(L_{\text{act}})$	1.571430	1.536907
Composite tail $\mathrm{CVaR}_{0.9}(\mathcal{E})$	1.715573	1.674220
Objective $\mathcal{J}$	0.899113	0.880262

E.11 Approximation fidelity

Each component uses 12 preselected stripes and matched contexts. Table 25 compares Eq. (6) with a denser native-response estimate that retains token, output, and input interactions in the candidate's projected response change.

E.12 Shared Rademacher stream stability

We vary the number of shared Rademacher streams over the same 128 calibration contexts, retaining one projected VJP per context. Each schedule recomputes component scales and global allocation with the fixed candidate bank, budget, and $\alpha = 0.875$. Table 26 compares calibrated scores and selected $K2$ stripes with the eight-stream reference. Jaccard is the intersection size divided by the union size of the two selected stripe sets.

Table 25: Rank correlation between Eq. (6) and a denser native-response estimate on sampled $\pi_{0.5}$ stripes.

Component	Spearman ρ
Vision	0.790
Language	0.965
Action	0.916

Table 26: Shared-stream stability on $\pi_{0.5}$. Score correlation and selector overlap use the eight-stream reference.

Streams	Score ρ vs. 8	Jaccard
1	0.9995	0.982
2	0.9983	0.956
4	0.9964	0.950
8	1.0000	1.000

F Response Analysis and Visualization

F.1 Sensitivity analysis protocol

For Figure 2(a), let $\Delta W_c^{\mathrm{err}}$ and $\Delta \overline{L}_c$ be the reductions in squared Frobenius weight error and complete-policy action MSE when component c moves from $K1$ to $K2$. Other components remain at full precision; interventions share the reference policy, 128 contexts, and sampling states. With added payload $\Delta B_c = \sum_{i \in \mathcal{I}_c} \Delta B_i$, benefits per bit are $u_c = \Delta W_c^{\mathrm{err}} / \Delta B_c$ and $v_c = \Delta \overline{L}_c / \Delta B_c$. Bars show $u_c / \operatorname{avg}_{c'} u_{c'}$ and $v_c / \operatorname{avg}_{c'} v_{c'}$, where $\operatorname{avg}_{c'}$ averages components $c' \in \mathcal{C}$. Normalization makes the two profiles dimensionless.

For Figure 2(b), define weight distortion $D_{i,k}^{\mathrm{w}} = \sum_b \|E_{i,k,b}\|_F^2$ for stripe i at rate k. Stripe benefits are $u_i = \max\{0, D_{i,1}^{\mathrm{w}} - D_{i,2}^{\mathrm{w}}\} / \Delta B_i$ and $v_i = \widetilde{\kappa}_{c(i)} \max\{0, D_{i,1}^{\mathrm{pol}} - D_{i,2}^{\mathrm{pol}}\} / \Delta B_i$. Every bank stripe contributes; each criterion is divided by its within-component median, cancelling component scaling. Boxes and whiskers summarize the empirical distribution. Vision, language, and action use native-response metrics; Extra uses its calibrated executed-interface local metric. Panel (a) measures complete-policy changes; panel (b) describes allocation-score geometry.

F.2 Offline response objective and closed-loop success

The nine configurations in Table 11 share 128 LIBERO contexts, full-precision references, and action-sampling states. There are 32 contexts per suite from three or four episodes per task, disjoint from calibration, search, and prefix selection.

For configuration i, Spearman $\rho(-P_i, \mathrm{SR}_i)$ relates offline loss P_i to macro success SR_i. Weight MSE averages weight errors; local reconstruction weights them by input second moments. Mean response averages $\mathcal{E}(z; s)$; $\mathcal{J}$ adds the tail term ($\beta = 0.25$, $\gamma = 0.9$). Empirical $\mathrm{CVaR}_{0.9}$ averages the largest $\lceil 0.1n \rceil$ of n losses.

Table 27: Offline-to-success alignment: nine matched-bank $\pi_{0.5}$ configurations.

Offline criterion	Spearman $\rho \uparrow$
Weight MSE	0.583
Local reconstruction	0.000
Mean response $\overline{\mathcal{E}}$	0.733
Response objective $\mathcal{J}$	**0.800**

The response objective aligns more closely with success than weight or local reconstruction. Among the six configurations from calibration onward, $\rho(-\mathcal{J}, \mathrm{SR}) = 0.371$; alignment is stronger over the broader quality range.

F.3 SPATIAL VISUALIZATION OF NATIVE RESPONSES

Following HB-VLA's activation diagnostics (Yan et al., 2026), we compare activation energy and native-response sensitivity on four preselected LIBERO contexts (indices 0, 32, 64, 96).

Let $h_{t,j}$ denote the middle vision layer's query projection at token t and channel j. With d_o channels and N_p probes,

$$A_t = \frac{1}{d_o} \sum_{j=1}^{d_o} h_{t,j}^2, \qquad P_t = \frac{1}{N_p d_o} \sum_{v=1}^{N_p} \sum_{j=1}^{d_o} \left(\frac{\partial \zeta_v}{\partial h_{t,j}} \right)^2, \tag{33}$$

Here A_t is activation energy, P_t is response sensitivity, and v indexes independent probes. Each ζ_v projects energy-normalized visual tokens after the multimodal projector; $N_p = 8$ per context.

The top 26 of 256 patches contain 20.6–22.1% of response sensitivity, versus 12.4–12.5% of activation energy. Sensitivity peaks cover both objects and contextual regions at the native visual interface.



Figure 5: Native-response maps on four LIBERO scenes: RGB input, activation energy A_t, and sensitivity P_t. Overlays preserve the 16×16 grid at 45% opacity. The shared log scale is normalized by each map's mean: white is 1, blue below 1, red above 1. Labels give the mass in the top 26 patches.