CAEM-BC: CONDITION-ADAPTIVE ELASTIC MODAL FLOW WITH BOUNDED CONTINUATION FOR VISION-LANGUAGE-ACTION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving a pretrained vision-language-action policy with limited demonstrations requires learning useful action corrections and joining them to commands already executed. We present Condition-Adaptive Elastic Modal Flow with Bounded Continuation (CAEM-BC), an adapter that refines action chunks while keeping the pretrained policy frozen. A condition-dependent elastic operator defines a low-rank temporal basis for residual prediction, preserving the base proposal outside the correction subspace. Bounded continuation then joins the corrected chunk to executed history while respecting action limits. CAEM-BC uses fewer than 6M trainable parameters and trains for 3,000 steps on 40 demonstrations per task. Across three backbones and four LIBERO suites, it improves average success for every backbone. An additional Spatial evaluation improves SmolVLA success by 3.2 percentage points while reducing executed-arm jerk by 40.5%, with further gains in execution quality under new initial states and language rewrites. Controlled component interventions show that residual correction repairs task failures, while continuation preserves the repaired outcomes and improves execution continuity. These findings connect efficient adaptation to task completion through a learned correction space and history-aware execution.

1 INTRODUCTION

A pretrained vision-language-action (VLA) policy already provides much of the behavior needed to complete a manipulation task. Adapting it with limited demonstrations calls for a representation that builds on this behavior. Learning a residual offers a direct way to do so: the pretrained proposal supplies the action sequence, and the adapter learns its correction. Lightweight correction and retrieval-assisted specialization already improve pretrained policies (Sendai et al., 2025; Syed et al., 2026). We ask how to organize a compact correction across time and make it compatible with the actions the robot has executed.

These are related but distinct problems. Within a chunk, the correction needed near a grasp can differ from that needed during transport or placement. A fixed temporal basis gives every condition the same correction space. Across chunks, a new proposal must join commands produced at the previous query; smoothing each chunk independently ignores this boundary. Simply reducing variation cannot resolve both problems, since a motion change may be necessary for task completion.

CAEM-BC learns where and how strongly a residual may vary by predicting stiffness and tethering along the action horizon (Figure 1). The resulting operator supplies a small temporal basis; separate branches predict its coefficients and correction strength. Applying the low-rank constraint to the residual preserves the proposal outside that subspace. Bounded continuation then balances the corrected proposal with continuity inside the chunk and at its junction with executed history.

The complete adapter has fewer than 6M parameters and trains for 3,000 steps on 40 demonstrations per task. Recorded runs take 2.7–5.7 minutes on cached features for Adapter-Pro and $\pi_{0.5}$. Evaluations across three frozen backbones and four LIBERO suites show gains in average success, while controlled interventions distinguish failure repair from execution continuity. A conditional residual space and a history-aware execution rule thus provide two complementary means of improving a pretrained policy with a small adaptation budget.

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.

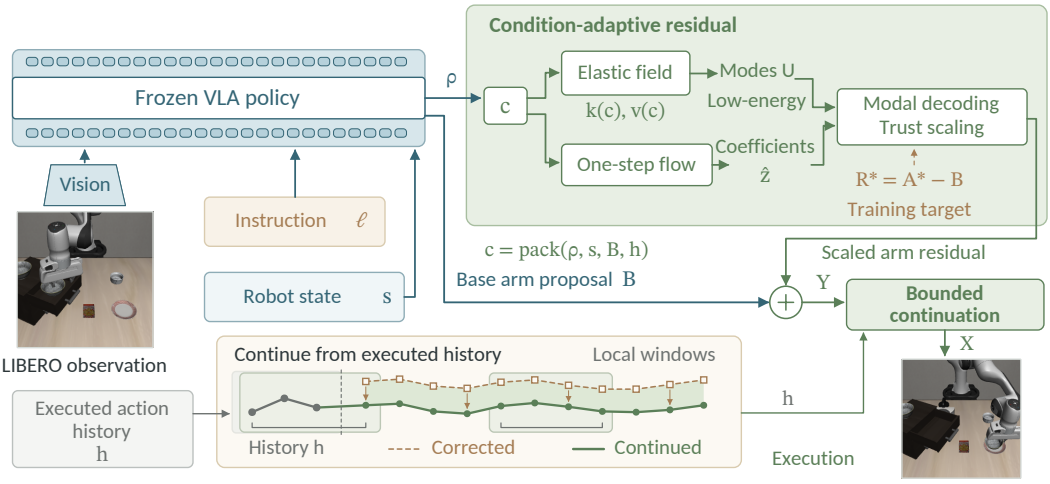

Figure 1: CAEM-BC refines a frozen VLA proposal with a condition-dependent basis, one-step modal coefficients, and learned trust. Bounded continuation joins the corrected arm chunk to executed history; the Base gripper is retained. The dashed arrow denotes residual supervision. Observation and execution frames come from the same LIBERO rollout; the history splice is schematic. Shaded windows show four-command differences across the boundary and within the chunk.

2 Related Work

VLA adaptation can change the action predictor or add a correction around it. OpenVLA-OFT studies decoding, chunking, and continuous objectives for fine-tuning (Kim et al., 2025); ExpReS-VLA uses frozen visual features, retrieval, and experience replay with LoRA adaptation (Syed et al., 2026). Among external corrections, A2C2 updates individual actions from fresh observations while freezing the VLA (Sendai et al., 2025), whereas VLA-Corrector uses latent drift to trigger replanning (Pan et al., 2026). CAEM-BC learns a temporally coupled residual at each policy query, concentrating adaptation in a low-rank correction space while keeping the backbone frozen.

Action chunking, used by ACT and Diffusion Policy (Zhao et al., 2023; Chi et al., 2023), makes the transition between successive predictions part of the execution problem. BID selects coherent chunks, RTC conditions asynchronous generation on committed actions, and REMAC addresses perception–execution mismatch (Liu et al., 2025; Black et al., 2025b; Wang et al., 2026a). Continuation can also be learned within the policy, as in LEGATO, or imposed during inference, as in SEAM (Liu et al., 2026; Zhan et al., 2026). CAEM-BC applies bounded continuation after residual decoding, using executed commands to define the chunk boundary. This lets the learned correction and the execution rule address different objectives.

The residual representation connects temporal compression with conditional basis learning. FAST uses DCT for action tokenization (Pertsch et al., 2025); CAEM-BC instead adapts a continuous residual basis, with fixed DCT as a controlled comparison. Deep Sturm–Liouville learns orthogonal bases for function approximation (Vigouroux et al., 2025); here, a discrete elastic operator is conditioned on each policy query. Temporal context enters through executed actions and frozen features, rather than the visual traces used by TraceVLA (Zheng et al., 2025). Flow-based manipulation policies and one-step distillation motivate efficient action generation (Jiang et al., 2025; Yan et al., 2025; Wang et al., 2025). We use flow matching (Lipman et al., 2023) in the modal space, with a deterministic objective that trains the single coefficient evaluation used at deployment.

3 Method

3.1 Refining a frozen action proposal

At policy query t, the frozen backbone receives observation o_t, instruction ℓ_t, and proprioceptive state s_t, and returns a native proposal $B_t \in [-1, 1]^{H_p \times 7}$ together with internal action-generation

features ρ_t. The proposal contains six arm coordinates and one gripper coordinate. We preserve each family's image preprocessing, normalization, proposal horizon H_p, and executed prefix length H. Adaptation acts on the arm prefix $B_t^{\mathrm{arm}} \in \mathbb{R}^{H\times 6}$.

Let $h_t \in \mathbb{R}^{m_t\times 6}$ contain the latest $m_t \le 3$ executed arm commands. A family-specific condition packs the frozen features, state, and proposal,

$$c_t = \mathrm{pack}(\rho_t, s_t, B_{t,1:H}, h_t). \tag{1}$$

The state contains end-effector position, rotation, and gripper position. History is left-padded with a validity mask. A shared encoder maps the normalized concatenation to 256 features for the basis, coefficient, and trust branches. Aligned demonstration actions define the residual target

$$R_t^\star = A_{t,1:H}^{\star,\mathrm{arm}} - B_t^{\mathrm{arm}}. \tag{2}$$

Learning a residual makes the backbone's proposal the origin of adaptation. The adapter need only represent its correction, rather than reproduce the complete action sequence in a low-rank space.

3.2 Condition-adaptive residual parameterization

Let $D_1 \in \mathbb{R}^{(H-1)\times H}$ be the first forward-difference matrix. A trainable basis branch maps the shared condition encoding to $H-1$ edge logits g_t^k and H node logits g_t^v. We construct

$$\begin{aligned} k_t &= k_{\min} + (k_{\max} - k_{\min})\sigma(g_t^k), \quad \widetilde{v}_t = v_{\max}\sigma(g_t^v), \\ v_t &= \widetilde{v}_t - \min_j \widetilde{v}_{t,j}\,\mathbf{1}, \qquad L_t = D_1^\top \operatorname{diag}(k_t) D_1 + \operatorname{diag}(v_t), \end{aligned} \tag{3}$$

where $k_{\min} = 0.05$, $k_{\max} = 4.0$, and $v_{\max} = 1$ for all four suites (Appendix B). Positive edge stiffness and nonnegative node tethering imply

$$x^\top L_t x = \sum_{i=1}^{H-1} k_{t,i}(x_{i+1} - x_i)^2 + \sum_{i=1}^{H} v_{t,i} x_i^2 \ge 0. \tag{4}$$

Thus L_t is symmetric positive semidefinite. Edge stiffness couples neighboring corrections; node tethering penalizes displacement from the frozen proposal. This discrete Sturm–Liouville construction connects to learning orthogonal bases through operator fields (Vigouroux et al., 2025). Here, the fields are conditioned on each action query and define the temporal geometry of its residual, allowing coupling and anchoring to vary across the execution horizon.

We retain the eigenvectors associated with the K smallest eigenvalues, denoted $U_{t,K} \in \mathbb{R}^{H\times K}$. Their variational characterization is

$$U_{t,K} \in \operatorname*{arg\,min}_{Q^\top Q = I_K} \operatorname{tr}(Q^\top L_t Q). \tag{5}$$

For a given operator, this selects the orthogonal subspace with minimum total elastic energy. Training adapts that energy to the demonstration residuals through reconstruction and action losses, so the retained modes become task-dependent correction shapes. The six arm coordinates share this temporal frame but have independent modal coefficients. With constant stiffness and zero tethering, L_t becomes a scaled Neumann path-graph Laplacian and its modes form a DCT-II basis. The learned representation therefore generalizes a fixed frequency chart through condition-dependent coupling and anchoring. Each query produces a new basis, coefficients, and trust for its executed prefix; only the network weights are shared across the trajectory.

Let $P_t = U_{t,K}U_{t,K}^\top$ and consider ideal projection at unit trust, before continuation. Residual prediction gives $\widehat{A} = B + P_t(A^\star - B)$, whereas whole-action prediction gives $\widehat{A}_{\mathrm{whole}} = P_t A^\star$. Their representation errors are

$$\widehat{A} - A^\star = -(I - P_t)(A^\star - B), \qquad \widehat{A}_{\mathrm{whole}} - A^\star = -(I - P_t)A^\star. \tag{6}$$

The residual bottleneck preserves $(I-P_t)B$, retaining useful base structure outside the subspace. This yields a testable criterion: discarded *residual* energy should be small. We test it with residual/whole-action and matched-rank reconstruction comparisons.

Projecting Eq. (2) gives $z_1 = \mathrm{vec}(U_{t,K}^\top R_t^\star) \in \mathbb{R}^{6K}$. We use a rectified interpolation with Gaussian noise ε and uniform time τ:

$$z_\tau = (1-\tau)\varepsilon + \tau z_1,$$
$$\mathcal{L}_{\mathrm{FM}} = \frac{1}{6K}\mathbb{E}\|u_\theta(z_\tau, \tau, c_t) - (z_1 - \varepsilon)\|_2^2, \qquad \mathcal{L}_{\mathrm{mean}} = \frac{1}{6K}\mathbb{E}\|u_\theta(0, 0, c_t) - z_1\|_2^2. \tag{7}$$

The second objective explicitly trains the deterministic prediction used at deployment. The joint objective combines these losses with residual reconstruction, action fidelity, and field and carrier regularization (Eq. 16, Appendix B). Squared training errors use elementwise means. A single evaluation yields the modal coefficients and decoded residual:

$$\widehat{z}_t = u_\theta(0, 0, c_t), \quad \overline{R}_t = U_{t,K}\,\mathrm{unvec}_{K\times 6}(\widehat{z}_t), \quad Y_t = B_t^{\mathrm{arm}} + \alpha_t \overline{R}_t. \tag{8}$$

Learned scalar trust $\alpha_t \in (0, 1)$ scales the decoded correction. It is trained jointly through downstream action and carrier losses, without a separate trust label. All three branches train jointly. Basis reconstruction compares $U_{t,K}U_{t,K}^\top R_t^\star$ with $R_t^\star$, rewarding modes that retain the required correction. Coefficient and continued-action losses also differentiate through the eigendecomposition, training k_t, v_t without field labels. Appendix B specifies regularization and numerical conventions.

3.3 Bounded continuation from executed history

The modal predictor describes a correction within the new chunk. It does not alone ensure compatibility with previously executed commands. To handle that boundary, let $D_3^{(n)}$ be the third forward-difference matrix on a sequence of length n. Let S_{m_t} select the first m_t rows whose support crosses the splice between history and the new chunk. We define

$$X^\star(Y_t; h_t) = \operatorname*{arg\,min}_{X \in [-1,1]^{H\times 6}} \mathcal{J}(X),$$
$$\mathcal{J}(X) = \tfrac{1}{2}\|X - Y_t\|_F^2 + \tfrac{\lambda}{2}\|D_3^{(H)} X\|_F^2 + \tfrac{\mu}{2}\|S_{m_t} D_3^{(m_t+H)}[h_t; X]\|_F^2. \tag{9}$$

The identity fidelity term makes the objective strictly convex. Writing the boundary difference as $EX + b(h_t)$ gives the unconstrained system

$$\left(I + \lambda D_3^\top D_3 + \mu E^\top E\right) X = Y_t - \mu E^\top b(h_t). \tag{10}$$

History provides an affine forcing term; setting $\mu = 0$ removes it. Clipping an unconstrained solution generally does not solve the bounded problem. At fixed history the exact minimizer is a nonexpansive proximal map (Appendix A). Deployment uses 64 projected-gradient steps initialized at $\mathrm{clip}(Y_t)$, with step size $1/\lambda_{\max}(Q)$ for the system matrix $Q = I + \lambda D_3^\top D_3 + \mu E^\top E$ and box projection at every iteration. All four suites use $\lambda = \mu = 0.03$ and merge the frozen Base gripper after arm realization.

3.4 Architecture and training

The complete adapter has 5.60M, 5.99M, and 4.90M trainable parameters for SmolVLA, $\pi_{0.5}$, and Adapter-Pro, respectively, including the shared encoder, all three branches, and the auxiliary gripper head. A width-256 encoder supplies the basis and trust branches and a four-block width-512 coefficient predictor. The backbone remains frozen. Table 7 reports exact totals, layer dimensions, and recorded training time.

A small basis network can exploit this structure by reusing frozen VLA features to predict the operator fields; eigendecomposition supplies orthogonality. Learning therefore concerns the temporal shape of the required correction, while the frozen proposal supplies the underlying behavior. Reconstruction and action losses backpropagate through the eigendecomposition, jointly adapting the residual space and its predicted coefficients without field labels.

Each task supplies 40 training and ten held-out validation demonstrations: 400/100 per ten-task suite. With cached frozen proposals and features, adapters train for 3,000 steps at batch size 64 using task-and-demonstration-balanced sampling and validation-based checkpoint selection. Within each comparison, CAEM-BC, trainable baselines, and ablations share demonstrations, preparation, splits, sampling, and evaluation-state protocols, and all complete training. Ablations change only the designated components; selector- or guard-assisted deployments are identified separately.

4 EXPERIMENTS

4.1 EVALUATION SETUP

We evaluate LIBERO-Spatial, Object, Goal, and Long (Liu et al., 2023), with ten tasks per suite and 50 episodes per task. SmolVLA, $\pi_{0.5}$, and Adapter-Pro retain their native observation/action interfaces (Shukor et al., 2025; Black et al., 2025a; Wang et al., 2026b; OpenHelix Team, 2025). Each adapted model is compared with its frozen backbone under the shared data protocol.

Success rate is the proportion of completed episodes. For executed six-dimensional arm commands, we compute forward differences of orders one to three, take their Euclidean norms, and average first within episodes and then across episodes. Delta, acceleration, and jerk denote these normalized-command statistics. Boundary differences cross query boundaries; within-chunk differences do not.

Table 1 compares each frozen backbone with its CAEM-BC adapter. Each suite contributes 500 episodes per method, and the four-suite average weights suites equally. Checkpoints use demonstration validation. Targeted component studies use 50 episodes; ten-task studies use 100.

Table 1: LIBERO success rate (%), with 500 episodes per method and suite. CAEM-BC adapts each frozen backbone using 40 training demonstrations per task and 3,000 optimization steps. Average weights the four suites equally.

Backbone	Configuration	Spatial	Object	Goal	Long	Average
SmolVLA	Base	84.8	91.8	87.6	72.2	84.10
	CAEM-BC	**85.8**	**93.6**	**89.6**	72.2	**85.30**
$\pi_{0.5}$	Base	98.0	100.0	97.8	95.0	97.70
	CAEM-BC	**99.4**	100.0	97.8	**95.4**	**98.15**
Adapter-Pro	Base	97.2	95.8	97.0	92.6	95.65
	CAEM-BC	**97.4**	**96.2**	97.0	92.6	**95.80**

4.2 MAIN RESULTS

CAEM-BC improves the four-suite average for all three frozen backbones (Table 1). SmolVLA rises from 84.10% to 85.30%, $\pi_{0.5}$ from 97.70% to 98.15%, and Adapter-Pro from 95.65% to 95.80%. Across 2,000 episodes per backbone, these changes correspond to 24, nine, and three additional successes. CAEM-BC improves seven of the twelve backbone–suite pairs and matches Base on the other five.

The gains span different baseline operating points: SmolVLA improves on three suites, while $\pi_{0.5}$ and Adapter-Pro improve from already high success rates. Every backbone uses 40 training demonstrations per task and 3,000 optimization steps (Section 3.4). The small adapter therefore improves both the lower-performing policy and the stronger baselines without changing their pretrained weights.

On Spatial, mean executed-arm third differences decrease by 37.54% for SmolVLA and 29.24% for $\pi_{0.5}$, with reductions on every task (Figure 2). Higher-order variation falls more than first differences. Task success is less uniform: SmolVLA gains on some tasks and loses on others, whereas $\pi_{0.5}$ improves six tasks and ties on four. Adapter-Pro Object adds two successes with 10.90% lower jerk.

The gains also extend to additional Spatial states and language instructions (Table 2). The SmolVLA checkpoint replication improves success by 3.2 percentage points while reducing jerk by 40.52%; Adapter-Pro gains one point on another initial-state bank. Under LIBERO-PRO language rewrites, Adapter-Pro retains 94.6% success with 25.96% lower jerk. The two guarded evaluations likewise preserve or improve success. Across all five settings, lower command variation accompanies maintained or improved completion, extending the evidence beyond the main evaluation bank.

To identify which task failures the adapter repairs, we return to the main Spatial bank and match episodes by initial-state identifier and state hash. Table 3 decomposes their outcomes. SmolVLA repairs 37 failures but introduces 32, so its net gain of five reflects substantial redistribution across

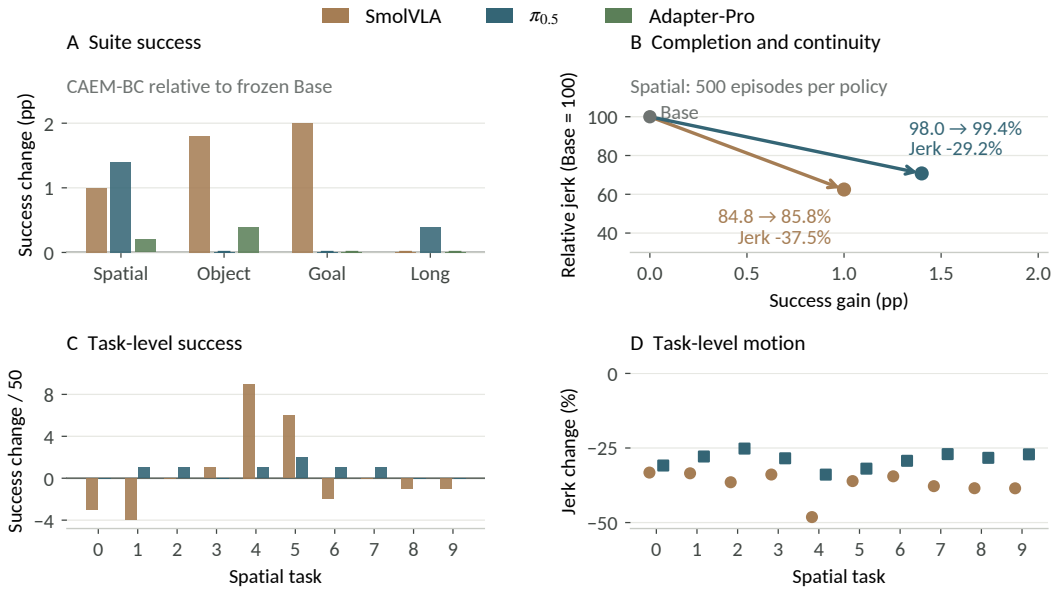

Figure 2: Task success and executed-motion analysis from the formal evaluations. (A) CAEM-BC success-rate gains relative to each frozen Base. (B) Spatial adaptation improves success while lowering mean executed-arm jerk; arrows join each Base to its adapted result. Jerk is normalized to each Base. (C) Spatial success-count changes, each task evaluated on 50 episodes. (D) Spatial third-difference changes by task. Spatial task panels show SmolVLA and $\pi_{0.5}$.

Table 2: LIBERO-Spatial performance across additional states and language instructions. Each evaluation covers ten tasks and 50 episodes per task per method. The first four rows use custom Spatial state banks; the last uses LIBERO-PRO language variations (Zhou et al., 2025). SR is success rate (%); Delta, Accel., and Jerk are relative changes in executed-arm command differences. Guarded rows include a verifier and causal fallback.

Backbone	Evaluation	Base SR	CAEM SR	Delta	Accel.	Jerk
Adapter-Pro	Initial-state bank	95.4	96.4	−1.90%	−9.59%	−22.89%
SmolVLA	Checkpoint replication	82.6	85.8	−8.02%	−28.78%	−40.52%
$\pi_{0.5}$	New states, guarded	96.4	96.4	−3.23%	−10.93%	−18.09%
SmolVLA	New states, guarded	83.0	86.0	−3.21%	−13.36%	−18.42%
Adapter-Pro	LIBERO-PRO language	94.6	94.6	−3.47%	−12.88%	−25.96%

states. For $\pi_{0.5}$, nine repairs and two regressions produce a more asymmetric change. A task-level tie consequently need not mean the same episodes succeeded.

Table 3: Spatial outcomes on 500 matched initial states. Repairs succeed only with CAEM-BC; regressions only with Base. Motion changes use equally weighted episode means on the both-success subset.

	Episode outcomes				Both-success motion change	
Backbone	Both succeed	Repairs	Regressions	Both fail	Jerk	Boundary jerk
SmolVLA	392	37	32	39	−35.54%	−28.48%
$\pi_{0.5}$	488	9	2	1	−27.79%	−23.89%

On episodes both methods solve, jerk falls by 35.54% and 27.79%, respectively, with gains both within chunks and across query boundaries (Appendix D). These improvements persist on shared successful tasks, even when the paired policies follow different trajectories.

Table 4: Common $\pi_{0.5}$ LIBERO-Spatial evaluation: 500 episodes per method, native chunk horizon 10, and no artificially added inference delay. Command differences are measured on the executed arm sequence.

Method	Successes	SR (%) ↑	Delta ↓	Acceleration ↓	Jerk ↓
Base	490/500	98.0	0.09727	0.06129	0.08949
RTC	482/500	96.4	**0.09253**	**0.04441**	**0.05356**
CAEM-BC	**497/500**	**99.4**	0.09437	0.05107	0.06332

At the native execution cadence without added inference delay, we compare CAEM-BC with RTC to examine the relationship between command variation and task completion (Table 4). RTC yields the lowest differences, while CAEM-BC achieves the highest success rate and reduces all three motion statistics relative to Base. The results distinguish successful action correction from minimizing command variation, motivating the component analysis below.

4.3 Residual correction and execution continuity

To separate correction from realization, we evaluate five arm transforms using one frozen SmolVLA checkpoint and 50 fresh task-4 states (Table 5). Initial states, proposal seeds, Base gripper, and H10 execution are shared; selectors and guards are disabled. Task 4 was selected from prior results for its correction headroom.

Table 5: Same-bank mechanism decomposition on SmolVLA LIBERO-Spatial task 4 (50 paired states). Repairs and regressions are relative to Base. Motion values are equally weighted episode means; lower is better.

Arm transform	Success	Repair/regress	Delta	Accel.	Jerk	Boundary jerk
Base	35/50	–	0.09924	0.09213	0.15758	0.28909
Base + continuation	35/50	0/0	0.09570	0.06994	0.10265	0.20585
Base + fixed D3 smoother	34/50	0/1	0.09739	0.07860	0.12739	0.29839
Learned residual, no continuation	**45/50**	11/1	0.08778	0.07761	0.12931	0.20440
Full CAEM-BC	**45/50**	11/1	**0.08249**	**0.05600**	**0.07977**	**0.13688**

The learned residual accounts for the success change in this bank: residual-only and full CAEM-BC have the same episode outcomes, each repairing 11 Base failures and introducing one regression. Continuation alone preserves the Base outcome set while lowering jerk by 34.86%. Combining it with the residual preserves 45/50 success and lowers jerk by a further 38.31% relative to residual-only. The fixed within-chunk smoother lowers within-query jerk by 44.79% but increases boundary jerk by 3.22% relative to Base. This separates task correction from execution quality: the residual changes which episodes succeed, while continuation improves how the successful correction is executed. Ordinary independent-chunk smoothing does not reproduce that boundary benefit.

4.4 Elastic representation and adaptation cost

We first compare whole-action and residual prediction within each basis at unit trust, then examine elastic versus DCT modes and learned trust (Table 6). Complete SmolVLA and Adapter-Pro deployments include their recorded guard/selection settings; matched component rows isolate representation choices.

Whole-action prediction reconstructs the current chunk; residual prediction learns a correction to Base. At unit trust, residual prediction raises Adapter-Pro Object task-1 success from 76 to 98% with elastic modes and from 80 to 98% with DCT. Retaining Base outside the correction subspace (Eq. (6)) leaves capacity for necessary changes. Table 14 compares two action-realization designs across three backbones.

An independent $H = 16, K = 8$ diagnostic on 18,031 episode-disjoint rows reduces residual reconstruction error by 7.32% and first-action error by 17.25% relative to DCT (Appendix C). The following studies test the deployed $H = 10, K = 5$ basis.

Table 6: Representation ablations: success rate (%). Unit-trust pairs compare whole-action and residual targets within each basis. Columns specify settings and episode counts; targeted tasks follow earlier outcomes, and disjoint task 4 uses new states. Bold marks the complete deployment with its recorded guard or selection settings. Component rows share data and training protocols.

Basis	Target	Trust	SmolVLA Spatial T4 $n = 50$	$\pi_{0.5}$ Spatial T5 $n = 50$	Adapter-Pro Object T1 $n = 50$	SmolVLA 10 tasks $n = 100$	SmolVLA Disjoint T4 $n = 50$
Frozen Base			68	94	98	86	62
Complete CAEM-BC			**84**	**98**	**100**	**86**	**70**
Elastic	Whole action	Unit	82	90	76	81	66
Elastic	Residual	Unit	70	92	98	85	66
DCT	Whole action	Unit	66	96	80	83	56
DCT	Residual	Unit	60	94	98	79	74
Elastic	Residual	Learned	70	92	96	77	42
DCT	Residual	Learned	72	96	98	77	48

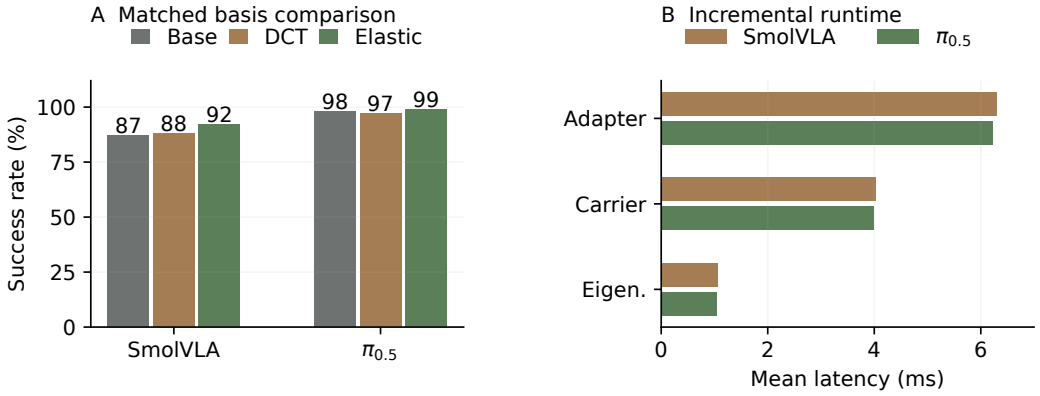

Figure 3: Basis choice and inference cost. (A) Matched basis interventions, 100 episodes per policy and family. (B) Synchronized batch-one measurements on RTX 5090; adapter totals exclude backbone inference. Carrier and eigensolve timings report separately benchmarked components of the complete adapter.

On separate 100-episode basis-intervention banks, elastic modes reach 99% versus DCT's 97% for $\pi_{0.5}$, with jerk 0.06330 versus 0.06680. For SmolVLA, elastic reaches 92% versus DCT's 88%, with jerk 0.08009 and 0.07657, respectively. Elastic modes improve completion for both backbones; jerk provides a complementary measure of execution quality. These matched basis interventions isolate the deployed representation at the same rank and execution horizon (Figure 3; Appendix E).

Across 100 held-out demonstrations per family, within-demonstration projector distances average 0.174/0.159 for $\pi_{0.5}$/SmolVLA, confirming adaptive subspaces (Appendix E). Separate trust interventions give 96 versus 95 successes for $\pi_{0.5}$ and 86 versus 83 for SmolVLA with learned versus constant-mean trust, each on 100 states.

The 4.90–5.99M adapters keep training and inference costs small. Recorded 3,000-step runs take 2.7–3.9 minutes for Adapter-Pro and 5.5–5.7 minutes for $\pi_{0.5}$, including validation and checkpoint writes but excluding frozen-feature preparation. The $\pi_{0.5}$/SmolVLA adapters add 6.23/6.30 ms per query on RTX 5090, excluding backbone inference (Figure 3); continuation dominates this cost.

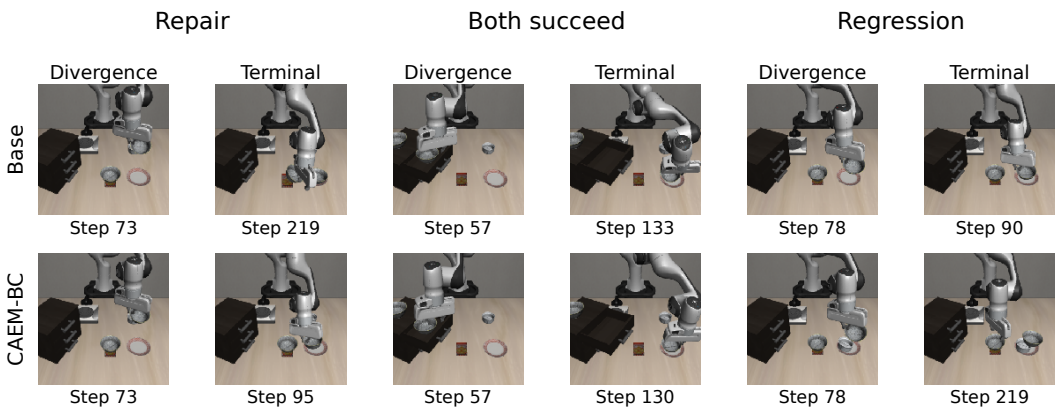

Figure 4: Divergence and placement outcomes from real $\pi_{0.5}$ Spatial observations. Each case shows the first material divergence and the terminal observation, with Base above CAEM-BC. Labels give executed steps. Repair: task 5/episode 27; both succeed: task 4/episode 10; regression: task 5/episode 49. Termination times differ. Full sequences appear in Appendix F.

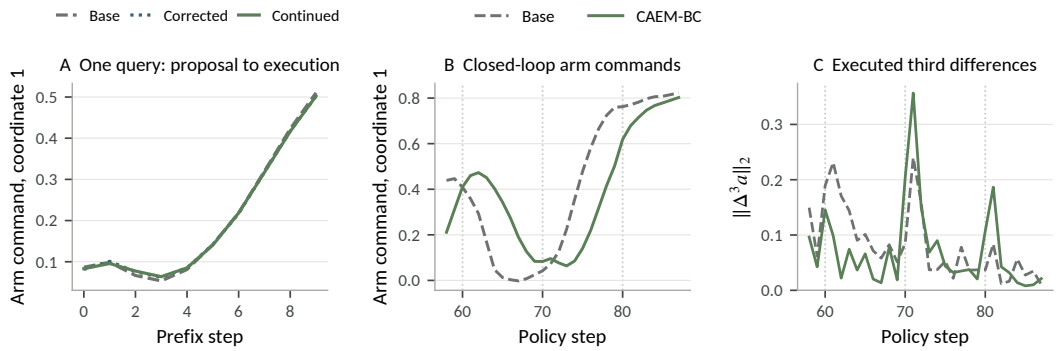

Figure 5: Formal $\pi_{0.5}$ Spatial rollout, task 4, episode 10, selected nearest the median relative jerk change among 488 both-success episodes (27.59% reduction). (A) Base, corrected, and continued commands at query step 70. (B) Executed commands and (C) third-difference norms on the same time window; dotted lines mark CAEM-BC queries. Coordinate 1 is selected by the largest proposal range at that query.

4.5 Closed-loop execution analysis

The rollout traces show how these corrections enter the executed sequence. Spatial boundary/within-chunk jerk falls by 34.10/40.67% for SmolVLA and 26.76/31.94% for $\pi_{0.5}$. In a separate 500-state Adapter-Pro carrier-only comparison, continuation lowers query-seam jerk by 21.40% and removes 33 recorded clips, with success changing from 483 to 482.

Figure 4 shows the behavioral differences: CAEM-BC repairs a bowl-on-plate placement in task 5, follows a different successful path in task 4, and misses a placement that Base completes in the regression case. Appendix F gives the full sequences. Figure 5 aligns the step-70 proposal, continued commands, and executed third differences, linking the local history splice to closed-loop continuity.

5 Conclusion

CAEM-BC adapts frozen VLA policies through condition-dependent low-rank residuals and history-aware bounded continuation. Experiments separate their roles in task correction and execution continuity, demonstrating improvements with limited demonstrations, fewer than 6M trainable parameters, and a short training schedule. Additional Spatial evaluations extend these gains to new initial states and rewritten instructions, with task success maintained or improved.

AI USE STATEMENT

We used AI tools to assist with manuscript editing, figure refinement, code development, refinement of theoretical explanations, and interpretation of experimental results. The authors take responsibility for the final content of this paper.

REFERENCES

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, volume 38, pp. 33383–33407. Curran Associates, Inc., 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/300ccb2187dedd4edcc07f7e76d8e553-Abstract-Conference.html.

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

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 238–257. PMLR, 2025. URL https://proceedings.mlr.press/v305/jiang25a.html.

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9.

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, pp. 44776–44791, 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0d78dd998f7b9ac79604d47a2d79bb0d-Abstract-Conference.html.

Yufeng Liu, Hang Yu, Juntu Zhao, Bocheng Li, Di Zhang, Mingzhu Li, Wenxuan Wu, Yingdong Hu, Junyuan Xie, Junliang Guo, Dequan Wang, and Yang Gao. Learning native continuation for action chunking flow policies. In *Proceedings of Robotics: Science and Systems*, 2026. URL https://roboticsconference.org/program/papers/58/.

OpenHelix Team. VLA-Adapter: Official implementation and pro checkpoints. https://github.com/OpenHelix-Team/VLA-Adapter, 2025. Accessed: 2026-07-28.

Yi Pan, Miao Pan, Qi Lu, Jiaming Huang, Man Zhang, Siteng Huang, Xin Li, Jie Zhang, Yongliang Shen, Xuhong Zhang, and Wenqi Zhang. VLA-Corrector: Lightweight detect-and-correct inference for adaptive action horizon, 2026. URL https://arxiv.org/abs/2607.01804.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://roboticsproceedings.org/rss21/p012.html.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for VLA action chunks, 2025. URL https://arxiv.org/abs/2509.23224.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

Shahram Najam Syed, Yatharth Ahuja, Arthur Jakobsson, and Jeff Ichnowski. ExpReS-VLA: Specializing vision-language-action models through experience replay and retrieval. In *IEEE International Conference on Robotics and Automation*, 2026. URL https://arxiv.org/abs/2511.06202.

David Vigouroux, Joseba Dalmau, Louis Béthune, and Victor Boutin. Deep sturm–liouville: From sample-based to 1d regularization with learnable orthogonal basis functions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 61480–61497. PMLR, 2025. URL https://proceedings.mlr.press/v267/vigouroux25a.html.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Yuzhang Shang, Ramana Rao Kompella, and Gaowen Liu. Real-time robot execution with masked action chunking. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/c35834443b7881e782e52b3519fe27c7-Abstract-Conference.html.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18638–18646, 2026b. doi: 10.1609/aaai.v40i22.38931. URL https://ojs.aaai.org/index.php/AAAI/article/view/38931.

Zhendong Wang, Max Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 63399–63416. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25ba.html.

Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, and Dieter Fox. ManiFlow: A general robot manipulation policy via consistency flow training. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2268–2293. PMLR, 2025. URL https://proceedings.mlr.press/v305/yan25a.html.

Dijia Zhan, Xuemiao Xu, Jinyi Li, and Jie Tang. SEAM: Smooth execution of action-chunked motion for vision-language-action policies, 2026. URL https://arxiv.org/abs/2607.04609.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://roboticsproceedings.org/rss19/p016.html.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/8667f264f88c7938a73a53ab01eb1327-Abstract-Conference.html.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2025. URL https://arxiv.org/abs/2510.03827.

A Residual projection and bounded realization

The following properties characterize residual projection, the bounded continuation solution, and its finite-step implementation at a fixed policy query.

A.1 Projection error

For a fixed orthonormal frame $U \in \mathbb{R}^{H\times K}$, let $P = UU^\top$. The least-squares coefficient matrix for a demonstration residual is $Z^\star = U^\top(A^\star - B)$. Consequently,

$$B + UZ^\star = PA^\star + (I-P)B, \tag{11}$$

$$\|B + UZ^\star - A^\star\|_F^2 = \|(I-P)(A^\star - B)\|_F^2. \tag{12}$$

For arbitrary predicted coefficients $\widehat{Z}$, orthogonality gives the exact error decomposition

$$\|B + U\widehat{Z} - A^\star\|_F^2 = \|\widehat{Z} - U^\top(A^\star - B)\|_F^2 + \|(I-P)(A^\star - B)\|_F^2. \tag{13}$$

The decomposition separates coefficient prediction error from residual representation error. The matched-rank diagnostic measures the latter, while component rollouts evaluate how the learned representation and predicted coefficients affect closed-loop task completion. At $K = H$, $P = I$ and the projection error vanishes for either complete orthonormal chart, up to numerical precision.

A.2 Carrier uniqueness and input sensitivity

For fixed history, write Eq. (9) as $\frac{1}{2}\|X - Y\|_F^2 + g_h(X)$, where g_h contains the two convex penalties and the indicator of the action box. Its minimizer is the proximal map $X^\star(Y; h) = \mathrm{prox}_{g_h}(Y)$. To see its input sensitivity directly, the optimality conditions are $Y - X \in \partial g_h(X)$ and $Y' - X' \in \partial g_h(X')$. Monotonicity of the subgradient implies

$$\|X - X'\|_F^2 \le \langle X - X', Y - Y'\rangle, \qquad \|X - X'\|_F \le \|Y - Y'\|_F. \tag{14}$$

At fixed history, the exact carrier is nonexpansive with respect to its input proposal. Across policy queries, the affine term in Eq. (10) incorporates the updated executed history into each continuation problem.

A.3 Finite projected-gradient realization

Let $Q = I + \lambda D_3^\top D_3 + \mu E^\top E$ and $M = \lambda_{\max}(Q) \ge 1$. The projected-gradient map with step $1/M$ has contraction factor at most $1 - 1/M$: the affine gradient step has that Lipschitz bound because the eigenvalues of Q lie in $[1, M]$, and Euclidean box projection is nonexpansive. Hence

$$\|X^{(64)} - X^\star\|_F \le (1 - 1/M)^{64}\|X^{(0)} - X^\star\|_F. \tag{15}$$

Every iterate satisfies the action box, and the bound quantifies convergence toward the unique minimizer. Training-path evaluation and deployment both use the same 64-step realization.

B IMPLEMENTATION DETAILS

B.1 TRAINING CONFIGURATION

We regenerate frozen proposals and internal features through each family's native interface and align targets to the executed horizon. The SmolVLA and $\pi_{0.5}$ adapters use $H/K = 10/5$; the Adapter-Pro suite models use $8/5$. Conditions contain state, the internal backbone representation, the decoded proposal prefix, and, for the current models, three executed arm actions plus a validity mask. For the suite models, each task contributes 50 demonstrations: IDs 0–39 train the adapter and IDs 40–49 form a whole-demonstration validation split, so neighboring windows never cross splits. These adapters train for 3,000 optimizer steps with AdamW, batch size 64, learning rate 3×10^{-4}, weight decay 10^{-4}, task-and-demonstration-balanced sampling, and validation every 100 steps.

Checkpoint selection minimizes task-balanced carrier validation error among candidates passing the fidelity, gripper, trust, bounded-motion, and basis-quality gates. Representation ablations vary the basis, prediction target, or trust while fixing the backbone, gripper owner, and continuation runtime. Runtime settings for the component studies are specified in their table captions. The following subsections define the losses, execution procedure, and comparison settings.

For SmolVLA and $\pi_{0.5}$, the condition concatenates an 11-dimensional state, five temporal summaries of frozen action-generation features, the 10×7 proposal prefix, and three arm-history commands with validity masks. The summaries are the full-horizon mean, executed-prefix mean, first and last executed-prefix features, and a normalized prefix linear moment. Adapter-Pro uses its 896-dimensional pooled action-token feature and an 8×7 prefix, giving input dimensions 3,702, 5,222, and 984, respectively. Each coordinate is centered and scaled using training-row statistics, with population standard deviation clamped below at 10^{-5}. Training adds Gaussian noise of standard deviation 0.02 to the normalized condition.

B.2 ADAPTER ARCHITECTURE AND TRAINING COST

The condition encoder is LayerNorm followed by $D \to 256 \to 256$ linear layers with SiLU activations. Concatenating its output, 30 modal coordinates, and $(\tau, \sin \pi\tau, \cos \pi\tau)$ gives the predictor's 289-dimensional input. A $289 \to 512$ projection feeds four residual blocks, each LayerNorm–Linear$(512, 1024)$–SiLU–Linear$(1024, 512)$, and a linear output produces 30 coefficients. The elastic-field, trust, and auxiliary gripper heads each use width 256, with output sizes $2H - 1$, one, and $3H$, respectively. The three branches share the same encoded condition. Table 7 summarizes the architecture, total trainable parameters, and recorded training time; the frozen backbone supplies the cached proposals and features.

Table 7: Adapter architecture and recorded training cost. Times span nine complete 3,000-step runs per timed family (three seeds on Object, Goal, and Long), including validation and checkpoint writes on cached inputs. SmolVLA wall times were not recorded.

	SmolVLA	$\pi_{0.5}$	Adapter-Pro
Condition input D	3,702	5,222	984
Executed horizon H	10	10	8
Retained modes K	5	5	5
Shared encoder width	256	256	256
Basis output $2H - 1$	19	19	15
Coefficient hidden width	512	512	512
Coefficient residual blocks	4	4	4
Coefficient output $6K$	30	30	30
Complete adapter parameters	5,601,340	5,993,500	4,897,526
Training time (minutes/run)	—	5.50–5.69	2.72–3.86

B.3 TRAINING OBJECTIVE

The implemented joint objective is

$$\begin{aligned}\mathcal{L} = 0.25\mathcal{L}_{\mathrm{FM}} + \mathcal{L}_{\mathrm{mean}} + \mathcal{L}_{\mathrm{basis}} + 0.1\mathcal{L}_{\mathrm{grip}} + \mathcal{L}_{\mathrm{jerk}} + \mathcal{L}_{\mathrm{first}} \\ + \mathcal{L}_{\mathrm{field}} + \mathcal{L}_{\mathrm{potential}} + \mathcal{L}_{\mathrm{preserve}} + \mathcal{L}_{\mathrm{carrier}},\end{aligned} \tag{16}$$

Table 8 gives the coefficients in Eq. (16); the definitions below incorporate each grouped term's coefficients. $\mathcal{L}_{\mathrm{basis}}$ reconstructs the target residual after projection. The jerk term penalizes only predicted jerk above the frozen proposal; the first-action term matches the first residual. Field regularization comprises edge-field smoothness, potential magnitude, total variation and curvature, and basis localization. Preservation and carrier terms respectively limit residual magnitude and compare the history-conditioned output with both the demonstration and frozen proposal. Transition regret is implemented but has zero weight in these checkpoints. The gripper event loss remains auxiliary when the runtime gripper owner is the frozen backbone.

Table 8: Loss coefficients in the current suite checkpoints. Terms shown on one row remain separate in code.

Term	Weight	Term	Weight
Flow matching	0.25	Deterministic mean	1.0
Basis reconstruction	1.0	Gripper event CE	0.1
Jerk excess	1.0	First action	1.0
Field smoothness	10^{-3}	Potential magnitude	10^{-4}
Potential TV	0.01	Potential curvature	0.01
Basis localization	0.1	Proposal preservation	0.5
Carrier target	1.0	Carrier preservation	0.5
Carrier smoothness excess	0.1	Transition regret	0.0

B.4 EIGENDECOMPOSITION

The small symmetric eigensystem is solved in float64 for float32 inputs, then each eigenvector sign is canonicalized before casting back. Autograd remains enabled. The original Adapter-Pro integration used a 960-dimensional history-free condition, unit residual trust, and an auxiliary event head for runtime gripper control; its continuation still received executed history.

B.5 LOSS DEFINITIONS AND NORMALIZATION

For a tensor T, write $\mathrm{ms}(T) = \sum_i T_i^2/|T|$, including the batch dimension. Thus $\mathcal{L}_{\mathrm{FM}} = \mathbb{E}[\|u_\theta(z_\tau, \tau, c) - (z_1 - \varepsilon)\|_2^2/(6K)]$ and likewise for $\mathcal{L}_{\mathrm{mean}}$. The Frobenius norms in the carrier optimization and projection identities remain sums, not means. Let $R = \alpha U\widehat{Z}$, $Y = B + R$, and X denote the implemented 64-step carrier output. For each batch member b, define $q_j(T_b) = (H-j)^{-1}\sum_{i=1}^{H-j}\|(D_jT_b)_i\|_2$. Then

$$\mathcal{L}_{\mathrm{basis}} = \mathrm{ms}(UU^\top R^\star - R^\star), \tag{17}$$

$$\mathcal{L}_{\mathrm{first}} = \mathrm{ms}(R_{:,1,:} - R^\star_{:,1,:}), \tag{18}$$

$$\mathcal{L}_{\mathrm{jerk}} = \mathbb{E}_b[\max(0, q_3(Y_b) - q_3(B_b))^2], \tag{19}$$

$$\mathcal{L}_{\mathrm{preserve}} = 0.5\,\mathrm{ms}(R), \tag{20}$$

$$\mathcal{L}_{\mathrm{field}} = 10^{-3}\,\mathrm{ms}(D_1k), \tag{21}$$

$$\mathcal{L}_{\mathrm{potential}} = 10^{-4}\,\mathrm{mean}(v) + 0.01\sum_{j=1}^{2}\mathrm{ms}(D_jv) + 0.1\,\mathrm{ms}(a), \tag{22}$$

$$\mathcal{L}_{\mathrm{carrier}} = \mathrm{ms}(X - A^\star) + 0.5\,\mathrm{ms}(X - B) + 0.1\,\mathrm{ms}(e). \tag{23}$$

The localization excess $a \in \mathbb{R}^{N\times K}$ and normalized motion excess $e \in \mathbb{R}^{N\times 3}$ are

$$a_{br} = \Big[\sum_i U_{b,i,r}^4 - 0.3\Big]_+, \qquad e_{bj} = \frac{[q_j(X_b) - q_j(B_b)]_+}{\max(q_j(B_b), 10^{-3})},$$

where N is batch size, $[x]_+ = \max(0, x)$, $r = 1, \ldots, K$, and $j = 1, 2, 3$. Thus $\mathrm{ms}(a)$ averages over samples and retained modes, and $\mathrm{ms}(e)$ over samples and difference orders. D_j acts along time. Potential regularization uses mean minimum-shifted v and squared first- and second-difference means.

The auxiliary gripper loss sums target-class-weighted cross entropy over batch and horizon, divided by the sum of target-class weights. Weights are inverse training-event frequencies, scaled to mean one over observed classes; unobserved classes receive zero weight. Current suite configurations use $v_{\max} = 1$, localization threshold 0.3, and $\lambda = \mu = 0.03$. All coefficients appear in Table 8.

B.6 Training and execution algorithm

1. Prepare aligned frozen proposals B, demonstration targets $A^\star$, conditions, and valid arm histories. Split by whole demonstration and fit condition normalization on training demonstrations only.
2. At each optimizer step, sample using the study's shared protocol, normalize the condition, and apply its configured condition noise. Construct k, v, L, compute the lowest K modes, and form $z_1 = \mathrm{vec}(U^\top (A^\star - B))$.
3. Sample ε, τ and evaluate both $u_\theta(z_\tau, \tau, c)$ and $u_\theta(0, 0, c)$. Decode the latter, apply trust, add B, and run 64 projected-gradient continuation steps with valid history.
4. Evaluate Eq. (16), backpropagate through the modal projection and eigendecomposition, clip gradient norm at 5, and update with AdamW. Validate at the configured interval and retain the minimum eligible carrier score.
5. At execution, obtain a native frozen proposal and condition; normalize without augmentation. Construct modes, evaluate $u_\theta(0, 0, c)$ once, decode and apply trust, and form $Y = B + R$.
6. Initialize $X^{(0)} = \mathrm{clip}(Y)$ and repeat $X^{(r+1)} = \mathrm{clip}[X^{(r)} - (QX^{(r)} - Y + \mu E^\top b)/\lambda_{\max}(Q)]$ for $r = 0, \ldots, 63$. Merge the runtime-owned gripper, execute the prefix, and update the history with commands actually executed.

B.7 Fixed smoother and five-way intervention

All five modes use the same frozen SmolVLA checkpoint, 50 task-4 initial states, proposal seeds, Base gripper, and $H = 10$ execution, without selectors or guards. Base returns B; residual-only returns Y. Carrier-only and Full apply Eq. (9) to B and Y, respectively, with $\lambda = \mu = 0.03$. The fixed D3 smoother applies the same 64-step box-constrained solver to B, with $\lambda = 0.03$, $\mu = 0$, and no history. Its strength is not fitted separately. Comparing smoother to carrier-only therefore isolates the history term while holding the input, within-chunk strength, action bounds, and solver fixed; comparing residual-only to Full isolates continuation on the learned correction.

Table 9: Current family integration and checkpoint-selection settings. AP denotes Adapter-Pro.

Family	H/K	Condition dim.	$v_{\max}$	Runtime gripper	Checkpoint rule
SmolVLA	10/5	3702	1	Frozen Base	Min. eligible carrier score
$\pi_{0.5}$	10/5	5222	1	Frozen Base	Min. eligible carrier score
AP (current suites)	8/5	984	1	Frozen Base	Min. eligible carrier score

The suite models in Table 9 use AdamW for 3,000 steps, batch size 64, learning rate 3×10^{-4}, weight decay 10^{-4}, gradient-norm clipping at 5, and validation every 100 steps. For these models, demonstrations 0–39 train the model and 40–49 validate it independently within every task. Eligibility requires lower validation arm error than Base, first-action error no more than 1.01 times Base, gripper-event accuracy at least 0.8, mean trust at most 0.55, improved bounded delta and jerk without acceleration exceeding 1.01 times Base, a learned-to-DCT reconstruction MSE ratio at most 1.05, and maximum basis localization at most 0.3. Both residual-corrected and continued outputs must have a worst-task MSE ratio at most 1.01 relative to Base. The final ranking uses the task-balanced carrier MSE ratio.

B.8 Native trust and auxiliary runtime settings

Native trust is a scalar sigmoid head inside CAEM-BC, initialized to 0.25 and learned end-to-end from Eq. (16); it continuously scales every query's residual. In the separate component studies, the correction-benefit selector is a two-layer SiLU classifier trained from non-official paired closed-loop packets. Its inputs concatenate condition, frozen chunk, native residual, and carrier-minus-base features. An episode is positive only when CAEM repairs a Base failure; stable successes, stable failures, and regressions are negative. A deterministic per-class episode split holds out every fifth sorted episode; query rows are sampled with class balance and trained by binary cross entropy. Validation aggregates each episode by its 20th-percentile query score and selects a threshold maximizing repair recall minus twice the false-intervention rate, with lower false intervention and then higher threshold as tie breakers. At runtime the threshold makes a hard CAEM-versus-Base choice. The guard deterministically checks intervention magnitude relative to the Base action, with optional episode latching or a query budget; native trust continuously scales the residual, while the selector makes a discrete deployment choice.

B.9 Data usage and comparison protocol

Each comparison uses a shared data protocol: baseline and ablation methods use the same demonstration selection, data preparation, training/validation split, sampling procedure, and evaluation-state usage as the corresponding CAEM-BC model. All trainable methods complete the training schedule specified for that study. Each study specifies its shared numerical settings and evaluation bank. Representation ablations fix the backbone, action coordinates, gripper owner, continuation, state bank, and episode keys. Elastic versus DCT changes the basis generator; residual versus whole changes the prediction target and Base addition; learned versus unit trust changes residual scaling and its head. The complete-deployment rows in Table 6 include the recorded auxiliary runtime settings. The five-arm decomposition reuses one trained checkpoint to vary residual application and continuation.

C Independent representation diagnostic

We evaluate residual projection on 18,031 episode-disjoint validation rows with horizon $H = 16$. Table 10 compares the two bases at equal retained rank. At $K = 8$, elastic modes reduce reconstruction MSE by 7.32% and first-action error by 17.25%. At $K = 4$, reconstruction MSE is essentially unchanged; at $K = 12$, the reductions are 4.32% and 28.19%, respectively. At full rank ($K = H = 16$), both orthonormal bases span the complete temporal space and reconstruct the residual to numerical precision (MSE below 4×10^{-14}). This diagnostic measures low-rank representation fidelity; deployed basis interventions and condition-dependent subspace measurements appear in Appendix E.

Table 10: Residual projection at matched rank on held-out episodes ($H = 16$). MSE is reported in units of 10^{-5}; first-action error is the mean ℓ_2 error.

	Reconstruction MSE ↓		First-action error ↓	
Rank K	DCT	Elastic	DCT	Elastic
4	36.2699	36.2897	0.05028	0.04842
8	6.1692	5.7173	0.01195	0.00988
12	1.9984	1.9120	0.00320	0.00230

Residual prediction lowers validation error with both elastic and DCT bases under both training budgets (Table 11). The five-demonstration setting validates on two held-out demonstrations per task with temporal stride 20; the twenty-demonstration setting uses ten held-out demonstrations per task and all recorded windows. Each setting covers all ten Spatial tasks. The comparison connects the residual parameterization to action fidelity: retaining the frozen proposal allows the learned subspace to represent the required correction.

Table 11: Whole-action and residual prediction on SmolVLA LIBERO-Spatial. Values are task-balanced validation MSE of the continued arm commands. Within each budget, all variants use the same data, training schedule, final-checkpoint selection, and unit trust.

Training demos per task	Steps	Basis	Whole MSE ↓	Residual MSE ↓
5	1,000	Elastic	0.013430	0.011635
5	1,000	DCT	0.012518	0.011489
20	1,500	Elastic	0.012733	0.010852
20	1,500	DCT	0.012040	0.010969

D Outcome-conditioned motion statistics

Table 12: Relative mean command-difference change (%) on formal Spatial episodes successful under both methods. Both methods complete every episode in each reported subset.

Backbone	Episodes	Delta	Accel.	Jerk	Boundary	Within
SmolVLA	392	−5.06	−23.41	−35.54	−28.48	−40.98
$\pi_{0.5}$	488	−2.54	−15.36	−27.79	−23.89	−31.81

Each arm trajectory is differenced along executed policy steps. For order r, we compute the Euclidean norm of each valid six-coordinate difference and average over that episode before averaging over the selected episode set. Boundary and within columns partition third-difference windows by whether their support crosses a query boundary. The relative change is $100(\overline{m}_{\mathrm{CAEM}}/\overline{m}_{\mathrm{Base}} - 1)$; it is not the average of episode-relative changes. The trace-selection criterion instead uses episode-relative jerk change to choose the episode nearest its median, as stated in Figure 5. Paired policies start from the same states and may follow different subsequent trajectories; these statistics compare their executed motion on shared completed tasks.

E Deployed basis, trust, and runtime studies

Table 13: Basis and trust interventions, each evaluated on its own 100-state bank with the corresponding Base policy. The trust study compares native query-dependent scaling with constant-mean scaling.

Study	Backbone	Variant	Successes / 100	Mean jerk
Basis	$\pi_{0.5}$	Base	98	0.08707
		Fixed DCT	97	0.06680
		Learned elastic	99	0.06330
	SmolVLA	Base	87	0.12534
		Fixed DCT	88	0.07657
		Learned elastic	92	0.08009
Trust	$\pi_{0.5}$	Base	95	0.09384
		Constant mean	95	0.06468
		Native learned	96	0.06568
	SmolVLA	Base	85	0.13456
		Constant mean	83	0.08672
		Native learned	86	0.08368

Across 100 held-out demonstrations per family, mean normalized projector distance to DCT is 0.100 for $\pi_{0.5}$ and 0.122 for SmolVLA; distance from each demonstration's initial subspace averages 0.174 and 0.159.

Table 14: Action realization on 100 matched LIBERO-Spatial states per backbone. Full-action equilibrium uses the complete elastic spectrum with degree-two-nullspace KKT correction; the proposal-relative executor retains the frozen proposal as its action reference. Success rate is in percent; gain is in percentage points.

Backbone	Full-action equilibrium	Proposal-relative executor	Gain
Adapter-Pro	94	96	+2
$\pi_{0.5}$	96	96	0
SmolVLA	89	94	+5

Retaining the proposal as the action reference maintains or improves completion in Table 14; full-action equilibrium yields lower boundary variation.

For equal-rank frames U, V, normalized projector distance is $\|UU^\top - VV^\top\|_F/\sqrt{2K}$. The geometry summaries first aggregate within demonstrations and then give equal weight to tasks. The 100 demonstrations per family are independent sampling units; overlapping validation windows are repeated measurements within them. The two distances measure departure from DCT and adaptation within each demonstration.

Synchronized RTX 5090 batch-one measurements use 30 warm-ups and 200 repetitions. The $\pi_{0.5}$/SmolVLA adapters take 6.23/6.30 ms on average (95th percentiles: 6.72/6.97 ms). Carrier cost is 3.98/4.04 ms versus 1.04/1.06 ms for eigensolving. These are incremental timings after condition construction, excluding backbone inference, rendering, and environment stepping. Continuation, rather than eigensolving, dominates the measured adapter cost.

Runtime measurements use the complete adapter implementation, including its instantiated gripper-event head, even where deployment preserves the base gripper. Components were benchmarked separately with synchronization around each sample; their sum need not equal the separately measured full-adapter latency.

In an additional unguarded $\pi_{0.5}$ replication on a separate 500-state bank, Base and CAEM-BC complete 486 and 482 episodes, respectively, with mean jerk 0.09140 and 0.06536. This bank is distinct from the guarded and official evaluations.

F TRACE-ALIGNED VISUAL CASES

Cases were selected nearest each category's median jerk change before RGB capture. Sequential replay matches all archived non-latency trace arrays. Frames show grasp, divergence, and termination; images are vertically flipped for display.

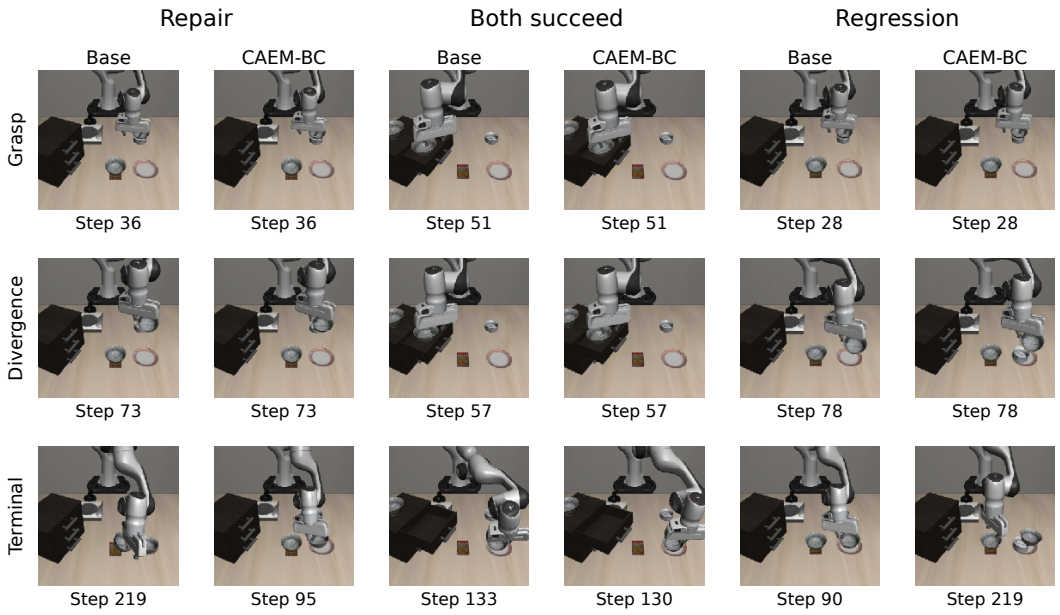

Figure 6: Three moments from each paired visual case: grasp command, first material divergence, and terminal observation. Each case has Base on the left and CAEM-BC on the right. Repair: task 5/episode 27; shared success: task 4/episode 10; regression: task 5/episode 49. Step labels refer to each policy's executed sequence.