STEERACT: TEMPORALLY GATED SINGLE-FEATURE STEERING OF FROZEN ACTION MODELS

Anonymous authors
Paper under double-blind review

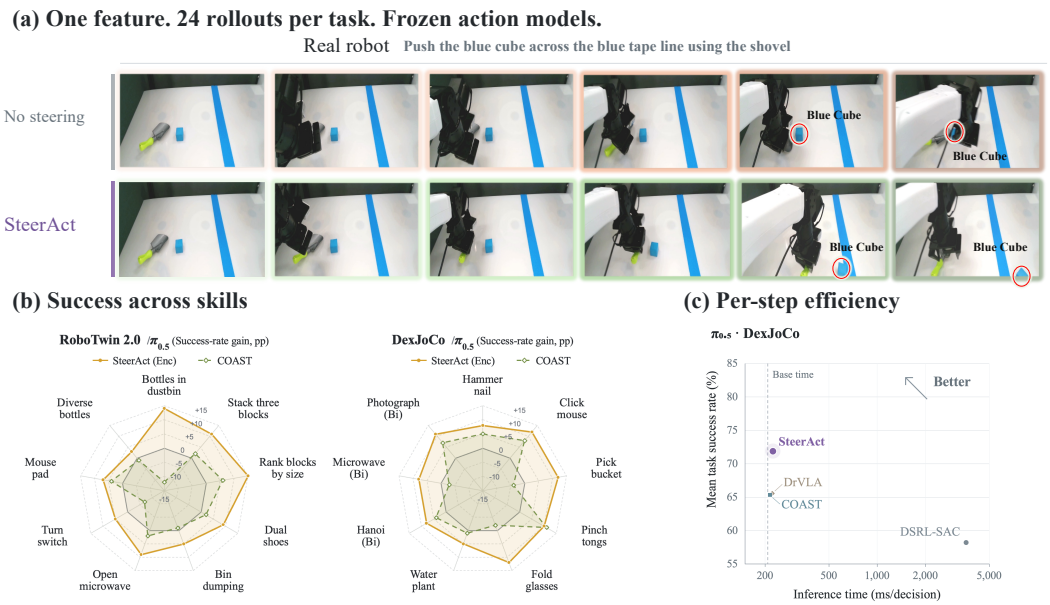


Figure 1: **Steering one feature at the right time.** Using 24 outcome-labeled rollouts per task, SteerAct steers a frozen action model through a single sparse feature, using temporal evidence to determine when to intervene and strength of the correction. **(a)** compares unsteered and steered executions of a real-robot manipulation task. **(b)** shows that SteerAct improves average task success across 18 tasks spanning RoboTwin 2.0 and DexJoCo. **(c)** compares mean task success rates against inference time per decision on DexJoCo with $\pi_{0.5}$.

ABSTRACT

Improving pretrained robot action models from a few rollouts requires identifying both what to steer and when to intervene and the sign of the correction. We find that successful and failed executions exhibit distinct temporal patterns in sparse latent space, and that steering one feature per task can improve closed-loop success without updating model weights. Based on this finding, we introduce SteerAct, a reference-guided steering method using a pretrained sparse autoencoder. Offline, SteerAct summarizes sparse-feature activation histories from successful and failed rollouts via temporal descriptors, constructs outcome-conditioned reference banks, and selects the feature with the strongest outcome separation. Online, a support gate assesses whether the observed activation history provides sufficient evidence for reference comparison, while a failure gate checks whether its descriptor is closer to failed than successful references. When both gates are open, a bounded search selects a signed correction predicted to move the current descriptor closer to the success bank. The selected feature direction and reference banks remain fixed; only the intervention coefficient adapts during execution. Using 24 outcome-labeled reference rollouts per task, SteerAct improves task success by approximately 10% relative to unsteered policies in evaluations spanning the vision–language–action model $\pi_{0.5}$, the world–action model Fast-WAM, RoboTwin 2.0, DexJoCo, and real-world manipulation. Code and Demo can be found at: Link.

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.

KEY WORDS: VLA, WAM, inference-time steering.

1 INTRODUCTION

Pretrained robot action models, including vision–language–action (VLA) and world–action models (WAMs), provide a reusable foundation for manipulation (Black et al., 2025; Yuan et al., 2026). Their capabilities, however, do not always translate into reliable task execution (Zhang et al., 2026; Fei et al., 2025). When additional robot trials are limited, a central challenge is to turn the available execution experience into effective behavioral corrections (Ball et al., 2023; Wagenmaker et al., 2025; Gu et al., 2026; Agia et al., 2024; Xu et al., 2025; Park et al., 2026; Mahato & Ren, 2026). We therefore ask: *can a few successful and failed rollouts identify a single internal feature through which a frozen action model can be guided toward more reliable execution?*

Inference-time steering offers a way to modify action generation without updating the pretrained policy. Existing approaches steer the initial noise of diffusion or flow policies (Wagenmaker et al., 2025; Tang et al., 2026), or intervene on internal feature activations and subspaces (Häon et al., 2025; Swann et al., 2026; Miao et al., 2026). Identifying a useful intervention target, however, does not by itself determine whether or how to intervene during a particular execution. Even for the same task, different rollouts may require different corrections, and a correction that helps at one decision may become unnecessary or disruptive later. Task-level feature selection must therefore be distinguished from rollout-level control of its intervention.

The history of feature activity provides information that a single activation value cannot capture. The same value can occur while activity is rising, falling, or fluctuating (Figure 2). We refer to the local ordering of feature activations across robot decisions as *temporal activation patterns*. These patterns allow us to compare how executions unfold without requiring their individual timesteps to align. They motivate conditioning intervention on the observed activation history, rather than prescribing a constant correction throughout a rollout.

We introduce SteerAct, a few-shot framework for temporal steering: adapting intervention decisions to the feature-activation history observed during the current execution (Figure 1). Its central idea is to use the same temporal descriptor for task-specific feature selection and rollout-dependent control. A shared pretrained sparse autoencoder (SAE) (Gao et al., 2025; Rajamanoharan et al., 2024; Lieberum et al., 2024; Marks et al., 2025) provides candidate features and their decoder directions. Offline, SteerAct summarizes each candidate feature's activation histories from successful and failed rollouts, constructs outcome-conditioned reference banks, and selects the feature whose banks exhibit the strongest outcome separation. This identifies a single steering direction from differences in activation dynamics, rather than from average activation alone. Online, a support gate assesses whether the observed history provides sufficient evidence for reference comparison, and a failure gate checks whether its descriptor is closer to failed than successful references. When both gates are open, a bounded search selects a signed correction based on its predicted effect on distance to the success bank. The selected feature, decoder direction, and reference banks remain fixed for the task; only the intervention coefficient adapts to the evolving rollout. The SAE is trained beforehand and shared across tasks. Neither its weights nor the action model's weights are updated during task adaptation or deployment, and no auxiliary reinforcement-learning controller is trained. Our contributions:

- **Few-shot single-feature selection.** We introduce an outcome-guided procedure that uses temporal descriptors of sparse-feature activity to select one steering feature per task from a few successful and failed rollouts. The selected feature's reference banks provide a common basis for both offline selection and online control.
- **Reference-guided temporal steering.** We use the selected feature's reference banks to adapt intervention to the observed activation history. A support gate and a failure gate determine when intervention is enabled, and a bounded search selects the signed correction. The action model and SAE remain frozen throughout this process.
- **Evaluation across models and manipulation settings.** Experiments with $\pi_{0.5}$ and Fast-WAM on RoboTwin 2.0 and DexJoCo, together with real-world manipulation, demonstrate an approximately 10% relative improvement in task success using 24 outcome-labeled reference rollouts per task. These results support the applicability of single-feature temporal steering across VLA and WAM (Figure 1).

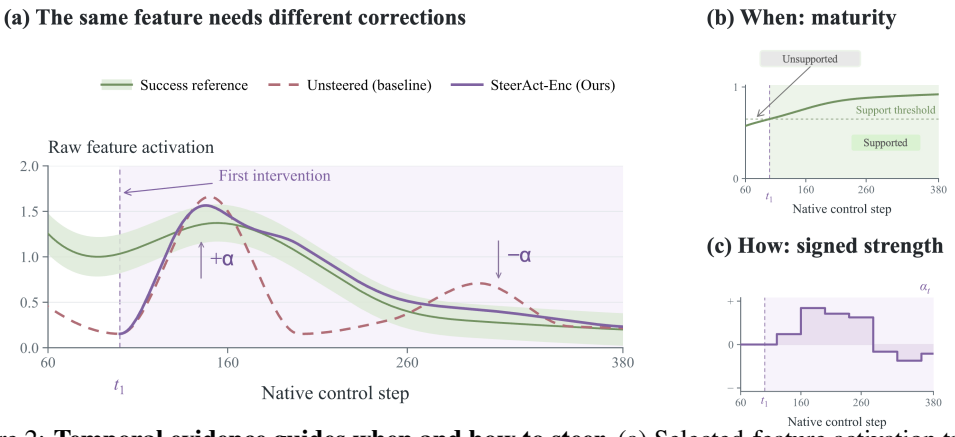


Figure 2: **Temporal evidence guides when and how to steer.** (a) Selected-feature activation traces compare a successful reference, an unsteered rollout, and rollouts with adaptive steering. (b) The temporal-support score indicates whether the observed rollout prefix has sufficient reference-bank support for diagnosis. (c) shows the intervention coefficient α_t, which determines both the sign and strength of the correction.

2 Related Work

2.1 Inference-time Steering

Inference-time steering improves the behavior of a pretrained model during deployment without updating its backbone weights (Liu et al., 2026; Tan et al., 2026). Candidate-selection methods sample multiple action proposals and rank them using a learned value function (Nakamoto et al., 2025), an action verifier (Kwok et al., 2025), a vision-language model reasoning over predicted outcomes (Wu et al., 2025), or consistency with previous action chunks and likelihood of future plans (Liu et al., 2025b). Sampler methods instead modify the generative process. Building on classifier and classifier-free guidance for diffusion models (Dhariwal & Nichol, 2021; Ho & Salimans, 2021), these approaches bias sampling through user-provided guidance (Wang et al., 2025) or objectives evaluated through learned dynamics (Du & Song, 2025). Others operate through the sampler's latent noise. DSRL treats the frozen sampler as an action transformation and learns a state-conditioned policy over its initial noise (Wagenmaker et al., 2025), while flow reversal steering maps candidate actions back to the noise that produces them (Tang et al., 2026). These methods control action generation through noise selection or prediction-based guidance. SteerAct instead performs representation-level steering by directly modifying a sparse internal feature and using its activation history to determine when and how to intervene. This requires hidden-state access but avoids learning an auxiliary RL controller or predicting future visual observations.

2.2 Representation-level Steering and Sparse Autoencoders

Representation-level steering modifies a model's hidden activations along directions associated with desired behavior (Jin et al., 2026; Hong et al., 2026). Prior work has identified such directions through optimized latent vectors, contrastive activation differences, probing, and representation engineering (Subramani et al., 2022; Turner et al., 2023; Li et al., 2023; Zou et al., 2023). SAEs provide a structured interface for representation-level control by decomposing model activations into a dictionary of sparse features (Bricken et al., 2023; Cunningham et al., 2024), whose decoder directions can provide candidate steering directions (Templeton et al., 2024). In robotics, mechanistic studies identify internal VLA directions associated with low-level behaviors such as motion direction and speed (Häon et al., 2025). SAE-based VLA steering using contrastive action pairs has been used to construct latent steering directions for VLA control (Khan et al., 2025), while individual SAE features have also been shown to encode interpretable motion primitives and semantic properties and to causally steer robot behavior (Swann et al., 2026). COAST instead uses successful and failed rollouts to fit contrastive conceptors that steer hidden states toward success-associated subspaces (Miao et al., 2026). Its global and denoising-step-specific variants establish that outcome-guided activation steering need not rely on a single additive vector. SteerAct builds on prior representation-level steering methods, but uses the feature's activation history to control steering over the course

of a rollout. From successful and failed reference rollouts, it selects a single sparse feature whose temporal activation patterns best distinguish the two outcomes. During deployment, it compares the observed feature history with the references to decide when and how to steer that feature.

3 METHODOLOGY

SteerAct uses a few successful and failed rollouts to steer one sparse feature of a frozen action model (Figure 3). We first define a temporal descriptor that summarizes how each feature's activity evolves over an execution. Offline, we construct success and failure reference banks for each candidate feature, select the feature whose banks are most clearly separated, and retain its direction and references. Online, we compare the observed activation history with these banks to assess whether intervention is warranted. When intervention is enabled, a bounded search selects the sign and strength of a correction by estimating its effect on the descriptor's distance to successful references. The descriptor therefore supports both offline feature selection and online correction; all online decisions use only the observed execution prefix (Appendix Algorithm 1).

3.1 PROBLEM SETTING

For a frozen action model and a fixed intervention site, we use a pretrained sparse autoencoder (SAE) shared across tasks. It maps a hidden-state vector $\mathbf{h} \in \mathbb{R}^d$ to a sparse code $\mathbf{z} \in \mathbb{R}^K$ and reconstructs it with a linear decoder:

$$\mathbf{z} = E(\mathbf{h}), \qquad \widehat{\mathbf{h}} = W_{\text{dec}}\mathbf{z} + \mathbf{b}. \tag{1}$$

Here, K is the number of SAE features. Feature f has an input-dependent activation $\mathbf{z}[f]$ and a fixed decoder direction $\mathbf{d}_f = W_{\text{dec}}[:, f] \in \mathbb{R}^d$. Its activation history provides the signal for deciding how to steer; its decoder direction provides the means to edit the model. For each task, we select one feature f^* and fix $\mathbf{d} = \mathbf{d}_{f^*}$. At robot decision t, a signed coefficient α_t modifies each targeted token's hidden-state vector:

$$\mathbf{h}'_{t,j} = \mathbf{h}_{t,j} + \alpha_t \mathbf{d}. \tag{2}$$

The sign of α_t selects movement along or against $\mathbf{d}$, while $|\alpha_t|$ sets the edit magnitude; $\alpha_t = 0$ leaves the model unchanged. Thus, adaptation requires selecting f^* from reference rollouts and determining α_t from the current execution history. The selected feature and its references are task-specific; the SAE and action-model weights remain frozen. Throughout this section, t indexes robot action decisions, not iterations within the action sampler.

3.2 TEMPORAL DESCRIPTOR

A feature's current activation does not reveal whether its activity has been rising, falling, or fluctuating. We therefore compare executions through local ordering patterns in their activation histories, rather than through a single activation value or a trajectory-level average.

Measuring feature activity. At decision t of rollout e, encoding measured token j gives $\mathbf{z}_{e,t,j} = E(\mathbf{h}_{e,t,j}) \in \mathbb{R}^K$. Stacking these codes as rows yields $\mathbf{Z}_{e,t} \in \mathbb{R}^{N_{e,t} \times K}$, where $N_{e,t}$ is the number of measured tokens. The scalar $[\mathbf{Z}_{e,t}]_{j,f}$ is the activation of feature f in token j. We average its rectified activation across tokens to obtain the *feature intensity*:

$$y_{f,e,t} = \frac{1}{N_{e,t}} \sum_{j=1}^{N_{e,t}} \max\{[\mathbf{Z}_{e,t}]_{j,f}, 0\}\,. \tag{3}$$

Representing temporal patterns. We summarize intensity histories using ordinal patterns (Bandt & Pompe, 2002). For readability, we omit the feature and rollout subscripts. We extract triples at two spacings: (y_i, y_{i+1}, y_{i+2}) and (y_i, y_{i+2}, y_{i+4}). Each triple belongs to one of six orderings; equal values are ranked by their temporal positions, with earlier measurements first. For each spacing, we compute nine statistics: six relative frequencies describing how often each ordering occurs; one entropy of this pattern distribution, normalized by $\log 6$; one fraction of triples containing equal values; and one binary indicator of whether any triple is available. Concatenating the two nine-dimensional summaries gives

$$\mathbf{v}_{f,e,t} = \phi(y_{f,e,1:t}) \in \mathbb{R}^{18}. \tag{4}$$

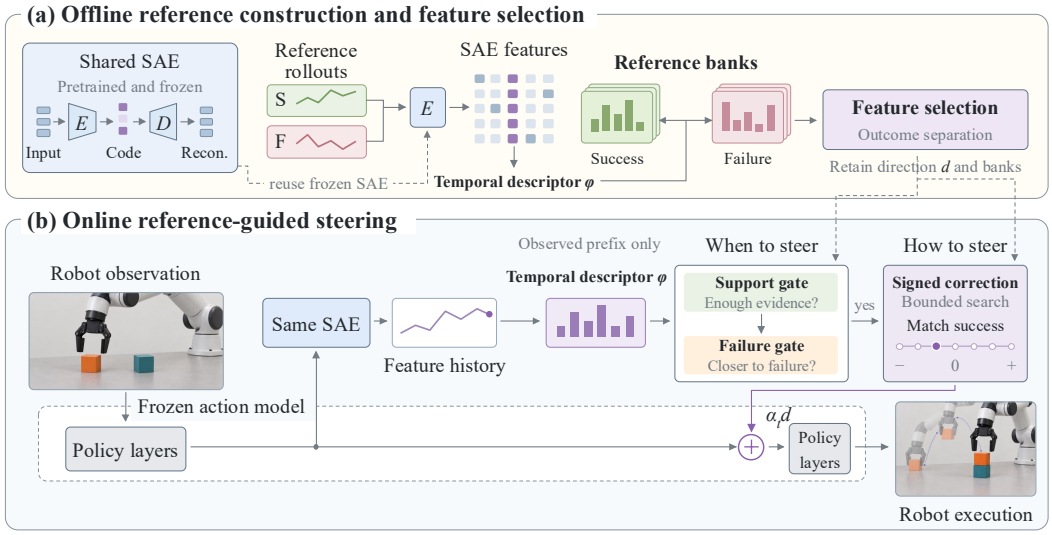


Figure 3: **SteerAct overview.** Offline, an SAE fitted to frozen-policy activations exposes candidate directions. Outcome-labeled rollouts determine a feature and its success/failure banks of ordinal dynamics. Online, supported failure-like prefixes enter a signed coefficient search; other prefixes retain $\alpha_t = 0$. The selected decoder direction modifies one configured residual stream, after which the frozen policy generates actions.

This descriptor summarizes local temporal patterns without requiring timestep alignment between rollouts. We retain the underlying pattern counts separately for online support assessment, since identical frequencies can arise from different amounts of evidence.

3.3 Offline Reference Construction and Feature Selection

Our offline goal is to identify one feature whose temporal behavior distinguishes successful from failed executions. We then retain examples of that feature's behavior to guide steering in new rollouts.

Reference banks. For each candidate SAE feature $f \in \mathcal{C}$, we apply the temporal descriptor defined above to its activation history in each complete reference rollout. We group the resulting descriptors by the rollout's final outcome: successful rollouts form $\mathcal{B}_f^S$, and failed rollouts form $\mathcal{B}_f^F$. Each bank is therefore a collection of examples summarizing how the feature behaves under one outcome.

Feature selection. A useful feature should separate outcomes: its success and failure descriptors should lie far apart, while descriptors within each outcome should remain tightly clustered. We therefore score each candidate by the between-outcome distance minus the within-outcome distances:

$$G_f = 2\bar{d}_f^{SF} - \bar{d}_f^{SS} - \bar{d}_f^{FF}, \tag{5}$$

where $\bar{d}_f^{SF}$ is the average Euclidean distance between success and failure descriptors. The other two terms measure average distances within the success and failure banks, excluding self-pairs. Computing this score requires at least two reference rollouts per outcome. We select the highest-scoring feature and use its SAE decoder column as the steering direction:

$$f^* = \arg\max_{f \in \mathcal{C}} G_f, \qquad \mathbf{d} = W_{\text{dec}}[:, f^*]. \tag{6}$$

We retain its two banks as $\mathcal{B}_S = \mathcal{B}_{f^*}^S$ and $\mathcal{B}_F = \mathcal{B}_{f^*}^F$. These references and the direction $\mathbf{d}$ remain fixed for the task. Selection identifies which feature to modify; the online procedure determines whether a modification is needed and which sign and strength to use.

3.4 Online Reference-Guided Steering

During a new rollout, we monitor the selected feature and adapt only its steering coefficient α_t. At each decision, we summarize its currently observed descriptor as $\mathbf{v}_t = \phi(y_{1:t})$. The two reference banks then help answer three questions: is the history sufficiently supported for comparison, does it resemble failure, and what correction would bring its descriptor closer to successful references?

Support gate. We first check whether the descriptor provides enough evidence for a meaningful comparison. With only a few measurements, a descriptor may appear close to a reference by chance

because its pattern frequencies are still uncertain. To account for this uncertainty, we draw M possible descriptors from a Dirichlet model of the observed pattern counts. A sampled descriptor is considered supported if at least k_{sup} reference descriptors, from either outcome, lie within an offline-calibrated radius ϵ_{sup}. If N_t^{sup} samples satisfy this test, the support score is

$$m_t = \frac{N_t^{\text{sup}}}{M}. \tag{7}$$

The support check passes when $m_t \geq \tau_m$ and enough triples have been observed at both spacings. Thus, this gate checks whether the available history supports comparison with the references; it does not predict task success. Sampling and neighborhood calibration are detailed in Appendix B.

Failure gate. Having sufficient support does not mean that a correction is necessary. We next compare the current descriptor with the successful and failed references. Let $D_S(\mathbf{v}_t)$ and $D_F(\mathbf{v}_t)$ denote its average distance to the nearest references in each bank; a smaller distance indicates more similar temporal patterns. Distances use coordinate scales fixed offline (Appendix B). The failure check passes when the descriptor is closer to failed than successful references:

$$D_F(\mathbf{v}_t) < D_S(\mathbf{v}_t). \tag{8}$$

To avoid rapid switching, each gate opens or closes only after a prescribed number of consecutive passing or failing checks. Steering is enabled only while both gates are open; otherwise, $\alpha_t = 0$. The persistence and reset rules are given in Appendix C.

Signed correction. When steering is enabled, we choose the coefficient α_t of the hidden-state edit in Eq. (2) from a candidate set $\mathcal{A}_t$: a fixed, bounded grid of positive and negative values. Positive and negative values edit the hidden state along opposite directions of $\mathbf{d}$, and 0 leaves it unchanged. For each candidate α, we predict the resulting activation intensity and recompute the temporal descriptor, replacing only the current entry:

$$\widehat{y}_t(\alpha) = \max\{0,\, y_t + \alpha g_t\}, \qquad \mathbf{v}_t(\alpha) = \phi(y_1, \ldots, y_{t-1}, \widehat{y}_t(\alpha)). \tag{9}$$

Here, g_t is a local linear sensitivity, the change in intensity per unit coefficient, obtained by re-encoding the edited hidden state (Appendix B). We score each candidate by how much closer its predicted descriptor moves toward the success bank:

$$\Delta_t(\alpha) = D_S(\mathbf{v}_t) - D_S(\mathbf{v}_t(\alpha)). \tag{10}$$

This measures a change in the current activation summary, not a prediction of future task success. We then apply the candidate whose predicted descriptor is closest to the success bank:

$$\alpha_t = \arg\max_{\alpha \in \mathcal{A}_t} \Delta_t(\alpha). \tag{11}$$

Since $0 \in \mathcal{A}_t$, the rule abstains whenever no candidate improves on the current descriptor; ties favor the smaller $|\alpha|$ (grid in Appendix C). After a nonzero correction, we re-measure the feature intensity.

4 Experiments

We evaluate whether outcome-conditioned temporal steering can improve closed-loop task success while keeping the underlying action policy frozen. Our experiments address five questions: (1) Does SteerAct improve performance across different tasks and backbones (Section 4.2)? (2) Does the method improve success in physical robot manipulation (Section 4.3)? (3) Which temporal representations are most effective for outcome matching (Section 4.4)? (4) Where should the intervention be applied (Section 4.4)? (5) How does performance depend on the amount of source experience and the availability of successful references (Section 4.4)?

4.1 Experimental Setup

Benchmarks and backbones. We evaluate on nine selected RoboTwin 2.0 tasks (Chen et al., 2025) and nine selected DexJoCo tasks (Wang et al., 2026), using both $\pi_{0.5}$ and Fast-WAM (Yuan et al., 2026) on each benchmark (Section 4.2). This gives 36 simulated task–backbone evaluation settings. We additionally evaluate three real-robot tasks with $\pi_{0.5}$ (Section 4.3). **Setting.** Unless otherwise stated, in simulation SteerAct uses 24 outcome-labeled source rollouts. Experiments

Table 1: Task success rates (%, ↑) on RoboTwin 2.0 and DexJoCo. Each Mean averages the nine tasks in its dataset. Time reports shared inference latency across datasets. Enc/Dec denote encoder/action-expert steering. Gray highlights SteerAct (Enc). **Bold**/underline indicate the best/second-best distinct success rates within each backbone, including ties.

	RoboTwin 2.0										DexJoCo										
Method	Bottles in Dustbin	Stack Three Blocks	Rank Blocks by Size	Dual Shoes	Dump Bin into Big Bin	Open Microwave	Turn Switch	Mouse Pad	Diverse Bottles	Mean	Hammer Nail	Click Mouse	Pick Bucket	Pinch Tongs	Fold Glasses	Water Plant	Hanoi Bi	Microwave Bi	Photo Bi	Mean	Time ↓ (ms/decision)
Backbone: $\pi_{0.5}$																					
Base	74	72	43	73	89	84	47	62	77	69.0	82	67	83	22	71	92	15	77	52	62.3	207.9
DSRL-SAC	80	78	40	57	49	**100**	**65**	**71**	**100**	71.1	80	60	70	18	74	96	10	69	47	58.2	3582.7
DrVLA	62	**84**	**60**	**86**	72	91	48	68	72	71.4	85	76	87	26	68	94	15	84	55	65.6	221.6
COAST	62	74	49	75	88	86	40	66	76	68.4	87	75	79	**33**	69	93	19	74	59	65.3	214.4
SteerAct (Dec)	79	83	39	81	92	87	44	67	66	70.9	79	74	75	21	**87**	96	17	**91**	59	66.6	278.1
SteerAct (Enc)	**88**	83	58	82	**94**	93	52	69	80	**77.7**	**90**	**79**	**95**	32	83	**97**	**23**	85	**63**	**71.9**	223.8
Backbone: Fast-WAM																					
Base	72	77	63	71	91	31	53	87	79	69.3	22	**7**	70	12	4	52	**0**	19	24	23.3	372.6
DrVLA	78	72	**76**	73	**94**	52	63	74	**84**	74.0	25	3	73	17	2	**63**	**0**	**25**	22	25.6	419.2
COAST	64	72	74	77	84	52	59	78	66	69.6	21	**7**	66	15	**5**	55	**0**	21	25	23.9	387.5
SteerAct (Dec)	**84**	**79**	75	81	91	**54**	60	90	80	77.1	24	5	63	**21**	**5**	57	**0**	**25**	17	24.1	423.9
SteerAct (Enc)	**84**	78	72	**85**	**94**	43	**73**	**95**	83	**78.6**	**27**	5	**76**	19	**5**	59	**0**	23	**27**	**26.8**	415.7

run on an NVIDIA RTX 6000 GPU. **Baselines.** *Base* is the unmodified pretrained policy. *DSRL-SAC* (Wagenmaker et al., 2025) trains a soft actor–critic controller over the initial noise supplied to a frozen generative action policy. For *COAST* (Miao et al., 2026), we evaluate its positive-only variant, which constructs contrastive conceptor operators from successful and failed rollouts and uses them to steer hidden activations toward success-associated subspaces. *DrVLA* (Swann et al., 2026) uses sparse autoencoders to identify interpretable VLA features and intervenes along their associated directions. We evaluate two variants of our method: *SteerAct (Enc)* intervenes in the backbone's encoder, whereas *SteerAct (Dec)* intervenes in its action expert, which applies steering at every denoising step whenever intervention is enabled. **Metrics.** Each method is evaluated with 200 held-out online rollouts per task. **Statistical significance.** Success rates are binomial proportions, so we test differences with two-proportion z-tests on counts pooled across tasks. All four aggregate gains over Base in Table 1 are significant (n=1800 per configuration; RoboTwin 2.0 with $\pi_{0.5}$: z=5.9, $p < 10^{-8}$; the others at $p < 0.05$). On the real robot (n=90 pooled), the 21.1 pp gain over Base is significant (z=2.9, p=0.004), whereas the 12.2 pp margin over DrVLA is not (z=1.6, p=0.10). Per-task differences at n=200 have a standard error of about 4 pp and are indicative only.

4.2 Evaluation Across Benchmarks, Embodiments, and Backbones

We evaluate SteerAct on gripper-based manipulation in RoboTwin 2.0 and dexterous-hand manipulation in DexJoCo, using $\pi_{0.5}$ and Fast-WAM as frozen backbones. Each task has its own selected feature and reference banks, while the steering procedure remains unchanged across settings. Detailed experimental protocol and hypermeters are listed in Appendix D. Ablation studies supporting out method are shown in 4.4.

Consistent aggregate gains. Table 1 shows that SteerAct (Enc) achieves the highest reported mean success in all four benchmark–backbone settings. Compared with Base, it gains 8.7 and 9.2 percentage points on RoboTwin 2.0, and 9.6 and 3.4 points on DexJoCo, with $\pi_{0.5}$ and Fast-WAM, respectively. Gains are computed before rounding the task macro-averages. These results support the applicability of the same procedure across the evaluated embodiments and architectures, although the magnitude of improvement varies. **Broad task-level improvements.** SteerAct (Enc) improves over Base on all 18 RoboTwin 2.0 task–backbone pairs and on 34 of 36 pairs overall. The exceptions are Click Mouse with Fast-WAM, which drops from 7% to 5%, and Hanoi Bi with Fast-WAM, which remains at 0%. Baseline effects are more mixed: DSRL-SAC improves aggregate success on RoboTwin 2.0 with $\pi_{0.5}$ but falls below Base on DexJoCo. The advantage is therefore broad improvement over the unmodified policy, not dominance on every task; individual baselines still achieve higher success on some tasks. **Success gains with modest inference overhead.** On RoboTwin 2.0, SteerAct (Enc) adds 15.9 ms per decision with $\pi_{0.5}$ and 43.1 ms with Fast-WAM, corresponding to 7.6% and 11.6% overhead. With $\pi_{0.5}$, its latency is close to DrVLA (223.8 versus 221.6 ms), while mean success is 6.2 points higher. With Fast-WAM, it achieves higher mean success at slightly lower latency than DrVLA. COAST is faster, but its mean success remains close to or below Base on these RoboTwin 2.0 evaluations. Thus, SteerAct offers a favorable success–latency trade-off without being the lowest-overhead method. **Remaining backbone dependence.** Aggregate improvements do not eliminate differences between the underlying models. On DexJoCo, Fast-WAM improves from 23.3% to 26.8%, but remains well below the 71.9% achieved with $\pi_{0.5}$. Hanoi Bi also remains at 0% across all reported Fast-WAM configurations. Targeted activation steering can therefore improve execution reliability without closing the backbone-dependent performance gap.

Table 2: Real-robot task success rate (%, ↑) with $\pi_{0.5}$. Base is the unmodified checkpoint; Enc and Dec identify the encoder branch and action expert of our method.

Task	Base	COAST	DrVLA	SteerAct Dec	SteerAct Enc
Folding Cloth	33.3%	26.7%	40.0%	36.7%	53.3%
Cup Ping-Pong	40.0%	36.7%	56.7%	43.3%	63.3%
Push Cube	20.0%	0%	23.3%	20.0%	40.0%
Average	31.1%	21.1%	40.0%	33.3%	52.2%

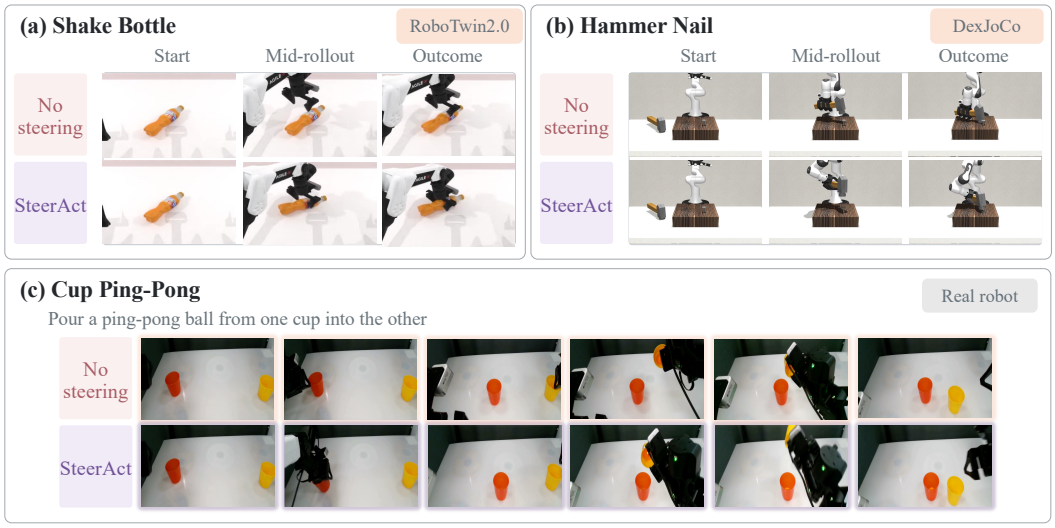


Figure 4: **Qualitative rollouts.** Tasks from RoboTwin 2.0, DexJoCo, and real-world manipulation.

4.3 Real-Robot Evaluation

Experimental setup. We evaluate $\pi_{0.5}$ as a frozen backbone with I2RT YAM (2 Realsense D405 camera, 1 Realsense D435i camera) on three physical manipulation tasks: Folding Cloth, Pingpong Transport, and Push Cube. Table 2 compares Base, COAST, DrVLA, and the Enc and Dec variants of SteerAct. We report task-level success rates and their unweighted mean. **Consistent improvements on physical tasks.** SteerAct (Enc) achieves the highest reported success on all three tasks, reaching 52.2% mean success compared with 31.1% for Base and 40.0% for DrVLA, the strongest evaluated baseline. These correspond to improvements of 21.1 and 12.2 percentage points, respectively. Gains over Base are 20.0 points on Folding Cloth, 23.3 on Pingpong Transport, and 20.0 on Push Cube. The benefit therefore extends across cloth manipulation, object transport, and pushing, rather than being driven by a single physical task. **Dependence on the intervention site.** The two steering variants differ substantially on real hardware. Enc outperforms Dec on every task, with mean success of 52.2% versus 33.3%. Dec remains close to Base overall and provides no improvement on Push Cube. These results favor encoder-branch over action-expert steering for the evaluated physical tasks, showing that a frozen policy's response to steering depends on where the intervention is applied.

4.4 Ablation Studies

We examine representation, intervention site, reference data, and steering choices on RoboTwin 2.0 with $\pi_{0.5}$. VLABench provides complementary comparisons of timing and gating. **Temporal descriptors improve outcome matching.** Table 3(a) compares different descriptors. The proposed descriptor achieves 77.7% mean success, exceeding autocorrelation, the strongest alternative descriptor, by 3.8 percentage points. Raw NN and DTW reach 72.2% and 71.7%, respectively: direct history matching and explicit temporal alignment do not recover the same aggregate performance. These results favor our descriptor under the shared gating scheme (Appendix E). **More source rollouts do not guarantee higher success.** With 12 total source rollouts per task, SteerAct reaches 73.6% mean success, 4.6 points above Base (Table 3(b)). Increasing the budget to 48 improves all nine tasks and raises the mean to 77.3%. However, the default 24-rollout configuration already achieves 77.7%, and increasing from 48 to 96 leaves the mean unchanged, with four task improvements,

Table 3: Ablations on RoboTwin 2.0 with $\pi_{0.5}$. Values are mean success rates (%) across nine tasks. Gray marks the default single-feature Enc variant (24 total source rollouts). Panels (b)–(d) use Enc; (c) counts successful source rollouts only. Full results are in Appendix Table 5.

(a) Representation and intervention site

Method	Base	Marginals	Increments	Events	Autocorr.	Raw NN	Raw DTW	Dec	**Enc**
Success	69.0	71.1	70.0	70.1	73.9	72.2	71.7	70.9	**77.7**

(b) Total rollouts

Rollouts	Success
12	73.6
48	**77.3**
96	**77.3**

(c) Successful rollouts

Rollouts	Success
8	74.4
24	77.6
48	**77.9**

(d) Steering choices

Setting	Success
Top-2 features	79.6
Top-4 features	**80.0**
Fixed $\alpha = 1$	75.8
Fixed $\alpha = 2$	75.7

Table 4: Mean success rates (%, ↑) across eight VLABench tasks, with 200 evaluation episodes per task and setting ($\pi_{0.5}$ with SteerAct(Enc)). Full results are in Appendix Table 6.

Setting	Frozen	Delay 10	Delay 20	Without support gate	Without failure gate	Without gates	Ours
Mean	23.63	24.63	24.88	23.13	29.88	22.38	**36.13**

four declines, and one tie. Thus, the reported results demonstrate effective few-shot adaptation, but not a monotonic benefit from enlarging the reference set. **Additional successful references yield diminishing aggregate gains.** Table 3(c) varies the number of successful source rollouts. Increasing this count from 8 to 24 raises mean success from 74.4% to 77.6%, a gain of 3.1 points. Increasing it further to 48 adds only 0.3 points, reaching 77.9%, with mixed task-level changes. These settings count successful episodes rather than all source episodes and are therefore not directly comparable to the default budget of 24 total rollouts. In particular, failed references must also be available for outcome-conditioned selection and steering. **A single feature is effective, but additional features can improve performance.** Table 3(d) shows that steering the top two and four features achieves 79.6% and 80.0% mean success, respectively, compared with 77.7% for the single-feature configuration. The corresponding gains are 1.9 and 2.3 points. These results establish that one feature is sufficient for substantial improvement over Base, not that one feature maximizes performance among the tested variants. The same group tests coefficient adaptation while retaining the intervention gates. Fixed coefficients of $\alpha = 1$ and $\alpha = 2$ achieve 75.8% and 75.7%, respectively, below the adaptive single-feature configuration by 1.9 and 2.0 points. Fixed corrections remain competitive on individual tasks, but neither matches the adaptive variant's aggregate success. This comparison supports choosing the correction during execution rather than prescribing either tested constant. **Both gating components contribute to aggregate success.** In the VLABench benchmark (Zhang et al., 2025) (Table 4) , the fully gated variant achieves 36.13% mean success and improves over the frozen policy on all eight tasks. Removing the support gate reduces the mean by 13.00 percentage points, while removing the failure gate reduces it by 6.25 points. Disabling both gates yields 22.38%, below the frozen policy's 23.63%. Neither remaining gate alone reproduces the full variant's aggregate benefit, supporting selective rather than unconditional intervention in this setting. Moreover, steering with our support gate achieves higher success than fixed-delay baselines that enable intervention only after 10 or 20 decision steps. This suggests that adapting intervention timing to the observed activation history is more effective than using these predetermined delays.

5 Conclusion

We introduced SteerAct, a few-shot framework for steering frozen action models through a single sparse feature per task. Temporal reference matching connects feature selection, intervention timing, and signed correction, while uncertainty-aware gating withholds unsupported interventions. Experiments on RoboTwin 2.0 and DexJoCo show mean success gains across $\pi_{0.5}$ and Fast-WAM, spanning gripper and dexterous-hand manipulation. Remaining regressions and backbone-dependent performance highlight the limits of steering alone. Together, these findings show that a single internal feature can provide a useful adaptation interface when its intervention is guided by the unfolding execution, without updating model weights.

AI USE STATEMENT

Generative AI assisted with manuscript editing, mathematical exposition, literature checking, table formatting, and figure export. No experimental measurements were generated. The authors are responsible for verifying the method implementation, citations, and empirical claims.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. *arXiv preprint arXiv:2410.04640*, 2024.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Christoph Bandt and Bernd Pompe. Permutation entropy: A natural complexity measure for time series. *Physical Review Letters*, 88(17):174102, 2002. doi: 10.1103/PhysRevLett.88.174102.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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, 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Maximilian Du and Shuran Song. DynaGuide: Steering diffusion policies with active dynamic guidance. In *Advances in Neural Information Processing Systems*, 2025. URL https://dynaguide.github.io/.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Ben D Fulcher and Nick S Jones. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, 2014.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Safe: Multitask failure detection for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40041–40076, 2026.

Bear Häon, Kaylene Caswell Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2743–2762. PMLR, 2025. URL https://proceedings.mlr.press/v305/haon25a.html.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jihoon Hong, Julian Skifstad, Qiyue Dai, Alice Chan, and Glen Chou. Steering robustness into world action models via mechanistic interpretability and optimal control. *arXiv preprint arXiv:2607.14943*, 2026.

Xinchen Jin, Aditya Chatterjee, Pranav Kumar, and Rohan Paleja. Event-grounded sparse autoencoders for vision-language-action policies. *arXiv preprint arXiv:2605.17204*, 2026.

Momin Ahmad Khan, Novak Boskov, Fatima Muhammad Anwar, and Manzoor Ahmad Khan. Controlling vision–language–action policies through sparse latent directions. In *NeurIPS 2025 Workshop on Mechanistic Interpretability*, 2025. URL https://openreview.net/forum?id=wtf3ww1EOL.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3200–3217. PMLR, 2025. URL https://proceedings.mlr.press/v305/kwok25a.html.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 41451–41530, 2023. doi: 10.52202/075280-1797. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/81b8390039b7302c909cb769f8b6cd93-Abstract-Conference.html.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 278–300, 2024.

Shuo Liu, Ishneet Sukhvinder Singh, Yiqing Xu, Jiafei Duan, and Ranjay Krishna. Vls: Steering pretrained robot policies via vision-language models. *arXiv preprint arXiv:2602.03973*, 2026.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025a.

Yuejiang Liu, Jubayer Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *International Conference on Learning Representations*, volume 2025, pp. 4594–4627, 2025b.

Dipesh Tharu Mahato and Rachel Ren. Early warning signals for openvla failure under visual distribution shift. *arXiv preprint arXiv:2606.29699*, 2026.

Samuel Marks, Can Rager, Eric Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations*, volume 2025, pp. 23888–23923, 2025.

Miranda Muqing Miao, Subin Kim, Brandon Yang, and Lyle Ungar. Contrastive conceptor activation steering (coast): Unlocking vision-language-action models through hidden states. *arXiv preprint arXiv:2605.17144*, 2026.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4996–5013. PMLR, 2025. URL https://proceedings.mlr.press/v270/nakamoto25a.html.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Seongheon Park, Wendi Li, Changdae Oh, Samuel Yeh, Zsolt Kira, Michael Hagenow, and Sharon Li. Hide-and-seek in trajectories: Discovering failure signals for vla runtime monitoring. *arXiv preprint arXiv:2605.30834*, 2026.

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. *arXiv preprint arXiv:2501.09747*, 2025.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Nishant Subramani, Nivedita Suresh, and Matthew E. Peters. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 566–581. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.48. URL https://aclanthology.org/2022.findings-acl.48/.

Aiden Swann, Lachlain McGranahan, Hugo Buurmeijer, Monroe Kennedy III, and Mac Schwager. Sparse autoencoders reveal interpretable and steerable features in vla models. *arXiv preprint arXiv:2603.19183*, 2026.

Derek Ming Siang Tan, Shailesh Shailesh, Srikrishna Iyer, William Wei Jie Teo, Yuanliang Ju, Qiao Gu, and Guillaume Sartoretti. RL^2-VLA: Adaptive rl latent compositional steering with test-time scaling for vision-language-action models. *arXiv preprint arXiv:2607.26991*, 2026.

Andy Tang, William Chen, Andrew Wagenmaker, Chelsea Finn, and Sergey Levine. Improving robotic generalist policies via flow reversal steering. *arXiv preprint arXiv:2606.13675*, 2026.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023. URL https://arxiv.org/abs/2308.10248.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

Hanwen Wang, Weizhi Zhao, Xiangyu Wang, Siyuan Huang, He Lin, Boyuan Zheng, Rongtao Xu, Gang Wang, Yao Mu, He Wang, Lue Fan, Hongsheng Li, Zhaoxiang Zhang, and Tieniu Tan. DexJoCo: A benchmark and toolkit for task-oriented dexterous manipulation on MuJoCo. *arXiv preprint arXiv:2605.16257*, 2026. URL https://arxiv.org/abs/2605.16257.

Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Pérez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15626–15633, 2025. doi: 10.1109/ICRA 55743.2025.11127931. URL https://doi.org/10.1109/ICRA55743.2025.11127931.

Yilin Wu, Thomas Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.076. URL https://doi.org/10.15607/RSS.2025.XXI.076.

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies. *arXiv preprint arXiv:2503.08558*, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11142–11152. IEEE, 2025.

Zhanguang Zhang, Zhiyuan Li, Behnam Rahmati, Rui Heng Yang, Yintao Ma, Amir Rasouli, Sajjad Pakdamansavoji, Yangzheng Wu, Lingfeng Zhang, Tongtong Cao, et al. Do world action models generalize better than vlas? a robustness study. *arXiv preprint arXiv:2603.22078*, 2026.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023. URL https://arxiv.org/abs/2310.01405.

APPENDIX

A Foundation Action Models **15**

B Reference Geometry and Local Response **15**

C Gate Updates, Bounded Search, and Feedback **17**

D Experimental Protocol and Hyperparameters **19**

E Representation and Intervention-Site Ablations **20**

F Qualitative Paired Simulation Rollouts **21**

G Limitations. **21**

Algorithm 1 SteerAct: Offline Selection and Online Steering

Require: Frozen action model, pretrained SAE, outcome-labeled rollouts, candidate features $\mathcal{C}$, and fixed control settings.

Offline: select a feature and its references

For each $f \in \mathcal{C}$, construct banks $\mathcal{B}_f^S, \mathcal{B}_f^F$ from descriptors in Eq. (4), and compute G_f using Eq. (5).
Select f^* and its decoder direction $\mathbf{d}$ using Eq. (6).
Retain $\mathcal{B}_S, \mathcal{B}_F$ for f^*; calibrate the support radius and coordinate scales (Appendix B).

Online: steer a new rollout

Initialize empty history, $\alpha_0 \leftarrow 0$, $u \leftarrow 0$, and closed gates with zeroed counters.
for each robot decision t until termination **do**
    Run the model to the intervention site; obtain $\mathbf{Z}_t$ and the unedited intensity y_t using Eqs. (1) and (3).
    Append y_t; compute $\mathbf{v}_t = \phi(y_{1:t})$ using Eq. (4), retaining the pattern counts.
    Compute m_t using Eq. (7); update the temporal support gate using $m_t \geq \tau_m$ and the minimum-count checks.
    If support is open, update the failure gate using Eq. (8); otherwise reset it and its counters.
    $\alpha_t \leftarrow 0$.
    if both gates are open **then**
        $\alpha_t \leftarrow \textsc{BoundedSearch}(y_{1:t}, \mathbf{Z}_t, \alpha_{t-1}, u)$, evaluating candidates with Eqs. (9) and (10).
    end if
    if $\alpha_t \neq 0$ **then**
        Edit each targeted token using Eq. (2).
        Re-encode the edited states and replace y_t with the measured intensity from Eq. (3).
    end if
    $u \leftarrow u + |\alpha_t|$; retain the applied α_t for the next decision.
    Resume the model from the resulting hidden states and execute its action output.
end for

A Foundation Action Models

Vision-language-action (VLA) and world-action models (WAMs) leverage large-scale pretraining and diverse robot experience to support general-purpose manipulation (Brohan et al., 2023; Kim et al., 2024; Black et al., 2025; Ye et al., 2026; Yuan et al., 2026; Black et al., 2024; Pertsch et al., 2025; Kim et al., 2025; Team et al., 2024; Liu et al., 2025a; Bjorck et al., 2025; Team et al., 2025; Shukor et al., 2025; O'Neill et al., 2024). While these models provide generalizable pretrained policies, their performance can remain unreliable on target tasks, motivating the need to improve their performance from limited additional experience. Our work addresses adaptation after such representations have been learned. Rather than introducing a new action-model architecture or updating its parameters, SteerAct uses outcome-labeled rollouts to guide interventions in the hidden activations of a frozen model to improve its performance on target tasks.

B Reference Geometry and Local Response

We specify the support sampling, reference calibration, distance normalization, and local-response approximation used in Section 3. These details complement the temporal descriptor and feature-selection criterion defined in the main text.

Count-based support sampling. For spacing $\ell \in \{1, 2\}$, let $n_{t,\ell,i}$ be the number of observed triples assigned to ordinal pattern $i \in \{1, \ldots, 6\}$. To represent uncertainty in the pattern probabilities, we draw

$$\widetilde{\mathbf{p}}_{t,\ell}^{(r)} \sim \text{Dirichlet}\left(n_{t,\ell,1} + \tfrac{1}{2}, \ldots, n_{t,\ell,6} + \tfrac{1}{2}\right), \qquad r = 1, \ldots, M, \tag{12}$$

where M is a fixed positive integer. For each draw, we replace the six empirical pattern frequencies with the sampled probabilities and recompute their normalized entropy:

$$H\left(\widetilde{\mathbf{p}}_{t,\ell}^{(r)}\right) = -\frac{1}{\log 6} \sum_{i=1}^{6} \widetilde{p}_{t,\ell,i}^{(r)} \log \widetilde{p}_{t,\ell,i}^{(r)}. \tag{13}$$

The observed tie fraction and validity indicator remain unchanged. Concatenating the resulting nine statistics at each spacing gives a sampled descriptor $\widetilde{\mathbf{v}}_t^{(r)} \in \mathbb{R}^{18}$.

Table 5: Ablations on RoboTwin 2.0 with $\pi_{0.5}$: (a) temporal descriptors, matching rules, feature counts, steering strengths, and intervention sites; (b) total source rollouts; (c) successful source rollouts. (d) feature count and coefficient adaptation. Entries are success rates (%, ↑); Mean averages over the nine displayed tasks. Matching controls use Enc and retain the ordinal-based support gate. Gray highlights the full method. **Bold**/underline mark the best/second-best distinct results per column within each group, including ties.

Method / Setting	Put Bottles in Dustbin	Stack Three Blocks	Rank Blocks by Size	Place Dual Shoes	Dump Bin into Big Bin	Open Microwave	Turn Switch	Place Mouse Pad	Pick Diverse Bottles	Mean
(a) Representation and steering ablations										
Base	74	72	43	73	89	84	47	62	77	69.0
Marginals	76	76	48	77	88	77	49	67	82	71.1
Increments	78	76	39	78	89	83	43	61	83	70.0
Events	76	78	45	77	89	82	50	54	80	70.1
Autocorrelation	82	83	49	**83**	93	86	49	56	**84**	73.9
Raw NN	75	81	52	76	87	89	**52**	66	72	72.2
Raw DTW	78	**84**	49	62	92	**96**	50	66	68	71.7
SteerAct (Dec)	79	83	39	81	92	87	44	67	66	70.9
SteerAct (Enc)	**88**	83	**58**	82	**94**	93	**52**	**69**	80	**77.7**
(b) Total source rollouts — SteerAct (Enc)										
12 rollouts	77	79	53	79	93	95	44	64	78	73.6
48 rollouts	**79**	84	**63**	**84**	**94**	**96**	47	70	79	**77.3**
96 rollouts	74	**87**	57	79	**94**	94	**51**	**74**	**86**	**77.3**
(c) Successful source rollouts — SteerAct (Enc)										
8 successful rollouts	77	79	60	75	90	88	**54**	66	81	74.4
24 successful rollouts	**79**	86	**64**	77	**94**	94	49	**71**	84	77.6
48 successful rollouts	76	**87**	59	**80**	**94**	**96**	52	70	**87**	**77.9**
(d) Feature count and coefficient adaptation — SteerAct (Enc)										
Top-2 features	**90**	87	59	**84**	94	96	51	**73**	82	79.6
Top-4 features	**90**	**89**	62	81	**95**	96	**54**	71	82	**80.0**
Fixed $\alpha = 1$	86	81	**66**	77	90	93	44	66	79	75.8
Fixed $\alpha = 2$	86	88	63	68	86	**97**	49	59	**85**	75.7

We evaluate these samples using the raw-distance support test in Eq. (7). The instantaneous support check passes only when $m_t \geq \tau_m$ and both spacings satisfy the fixed minimum triple-count requirement. These samples are used only for support assessment: outcome comparison and candidate evaluation use the empirical descriptor, not a sampled one. The Dirichlet model is a count-based uncertainty approximation; it does not explicitly model dependence among overlapping triples.

Support-radius construction. Let $N = |\mathcal{B}_S| + |\mathcal{B}_F|$ be the number of pooled reference descriptors. For each reference i, let q_i be its raw Euclidean distance to its k_{sup}-th nearest other reference, excluding itself. For a fixed quantile level ρ, we set

$$\epsilon_{\text{sup}} = 3\, Q_\rho(q_1, \ldots, q_N), \tag{14}$$

as offline-calibrated radius, where Q_ρ denotes the empirical ρ-quantile. The settings satisfy $0 < \rho < 1$ and $1 \leq k_{\text{sup}} \leq N - 1$. The radius is computed once from the retained reference banks and is not updated using evaluation rollouts. Thus, the radius is data-dependent, while the quantile level, neighbor count, and multiplier are fixed settings.

Outcome-distance normalization. For descriptor coordinate $q \in \{1, \ldots, 18\}$, let $\mu_{S,q}, \sigma^2_{S,q}$ and $\mu_{F,q}, \sigma^2_{F,q}$ denote the empirical means and population variances in the success and failure banks. The standard-deviation scale for their equally weighted mixture is

$$s_q = \max\left\{10^{-8},\ \sqrt{\frac{\sigma^2_{S,q} + \sigma^2_{F,q}}{2} + \frac{(\mu_{S,q} - \mu_{F,q})^2}{4}}\right\}. \tag{15}$$

Population variances use the bank size as their denominator. Each outcome receives equal weight regardless of its number of references. The between-outcome mean term is included, and the floor handles constant coordinates, including the validity indicators in full-rollout references. For

Table 6: Timing and gating comparisons on VLABench. Success rates (%, ↑) use 200 evaluation episodes per task and reported setting; Mean is the task macro-average. Gray highlights maturity-based steering. **Bold**/underline indicate the best/second-best distinct reported results per column, including ties.

Method	Painting	Book	Chemistry Drinking	Tube	Poker	Mahjong	Toy	Fruit	Mean
Frozen	17	46	15	47	18	6	7	33	23.63
Delay 10	17	**51**	22	47	13	13	5	29	24.63
Delay 20	17	44	19	52	16	11	12	28	24.88
Without support gate	15	49	23	49	12	8	5	24	23.13
Without failure gate	13	48	27	53	22	18	25	33	29.88
Without gates	17	43	24	42	22	9	3	19	22.38
SteerAct	**34**	50	**39**	**60**	**22**	**14**	**26**	**44**	**36.13**

descriptors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{18}$, the scaled distance is

$$\mathrm{dist}(\mathbf{u}, \mathbf{v}) = \left[\sum_{q=1}^{18} \left(\frac{u_q - v_q}{s_q} \right)^2 \right]^{1/2} . \tag{16}$$

For a fixed positive integer k, both outcome comparisons use

$$k_{\text{eff}} = \min\{k, |\mathcal{B}_S|, |\mathcal{B}_F|\} . \tag{17}$$

Let $\mathcal{N}_C(\mathbf{v})$ contain the k_{eff} nearest references to $\mathbf{v}$ in bank $\mathcal{B}_C$, for $C \in \{S, F\}$. The outcome distances are

$$D_C(\mathbf{v}) = \frac{1}{k_{\text{eff}}} \sum_{\mathbf{u} \in \mathcal{N}_C(\mathbf{v})} \mathrm{dist}(\mathbf{v}, \mathbf{u}), \qquad C \in \{S, F\}. \tag{18}$$

Using the same neighbor count makes the two comparisons consistent even when the banks have different sizes. The outcome distances use the scaled metric, whereas support-radius calibration and support assessment use raw Euclidean distance. All queries use the same full-rollout banks and fixed scales; nearest neighbors are recomputed for each query, including each candidate correction.

Local SAE response. Let r_t be the fraction of measured tokens on which the selected feature has positive activation. The local-response approximation uses an offline feature self-response coefficient c_{f^*} and sets

$$g_t = r_t c_{f^*}. \tag{19}$$

For an affine encoder before sparse feature selection, with a locally unchanged active set and identical measured and edited token sets, $c_{f^*} = \mathbf{w}_{f^*}^\top \mathbf{d}$, where $\mathbf{w}_{f^*}$ is the selected feature's encoder weight vector. Under these assumptions, each active token contributes the same first-order response, and averaging over all measured tokens yields the factor r_t. This approximation is not an identity for arbitrary SAE encoders or edits that change the active set. We abstain when g_t is nonfinite or its magnitude falls below the configured minimum effective gain. The prediction is used only to evaluate candidates in Eq. (9).

Intensity preprocessing. The implementation retains raw intensities in the history and applies $\widetilde{y}_t = \log(1 + y_t/s_f)$ before descriptor construction, using a positive feature-level scale s_f fixed offline. In exact arithmetic, this strictly increasing transformation preserves both ordering and ties, so it leaves the ordinal descriptor unchanged. The local-response approximation operates on raw intensities, before this transformation.

C Gate Updates, Bounded Search, and Feedback

Visualizing support-gate maturity. Figure 5 illustrates the support calculation at the beginning and end of one successful rollout. The archived visualization indexes this trajectory at native environment-step resolution, denoted by t_{env} to distinguish it from the controller's decision index t. At either cadence, the observed intensity prefix of the selected feature supplies ordinal-pattern counts at each

temporal spacing. Conditioned on these counts, the Dirichlet posterior represents uncertainty over the underlying probabilities of the possible ordinal patterns. Each draw can be read as one plausible mixture of local evolution patterns given the feature's behavior so far and, under the local-stationarity approximation, its near-term continuation. Replacing the empirical pattern frequencies by a draw and recomputing the associated entropy yields one possible current-prefix descriptor. Repeating this procedure produces the purple cloud. Thus, the posterior is conditioned on the pattern counts underlying the current descriptor.

For every purple point, we compute its raw descriptor distance to the k_{sup}-th nearest entry in the pooled bank $\mathcal{B}_S \cup \mathcal{B}_F$. These bank entries are descriptors computed from complete successful and failed trajectories. A point is supported when this distance is at most the calibrated radius ϵ_{sup} in Eq. (14); the supported fraction of the M purple points is the score m_t in Eq. (7). Once $m_t \geq \tau_m$ and the evidence and persistence checks also pass, the support gate treats the descriptor as sufficiently established for outcome comparison. Passing this gate alone never triggers an edit: steering is applied only if the subsequent failure gate also opens by indicating that the current dynamics are closer to failed than successful references.

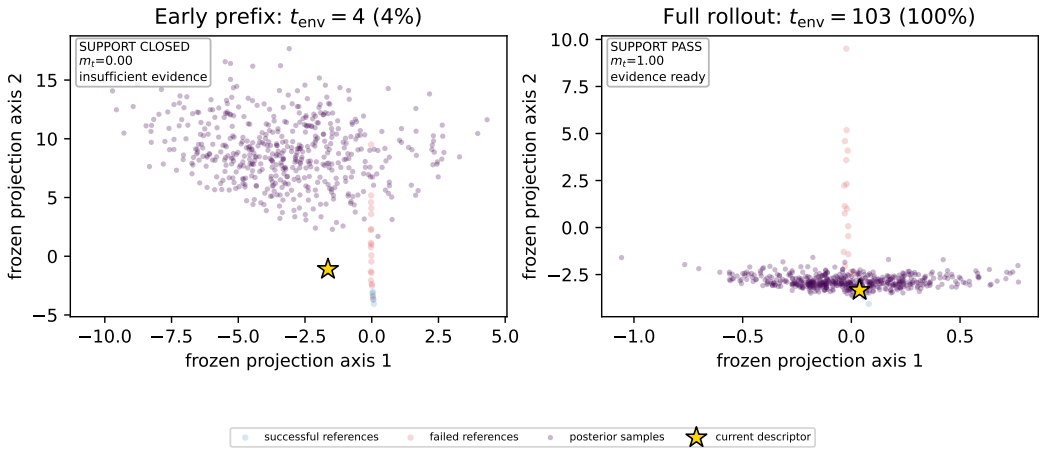


Figure 5: **Evolution of support for the selected feature.** Two endpoints from a successful task's rollout using descriptor, PaliGemma layer 14, and selected SAE feature 1118. Purple points are descriptors sampled from the Dirichlet posterior conditioned on the observed prefix; light blue and red points are full-rollout descriptors in the success and failure banks. At the earliest prefix ($t_{\text{env}} = 4$), the cloud is broad, none of the posterior samples passes the support test ($m_t = 0.00$), and the support gate remains closed. At the full rollout ($t_{\text{env}} = 103$), the cloud has contracted around the reference support, all posterior samples pass ($m_t = 1.00$), and the support gate opens. Here t_{env} retains the native-step indexing of the archived visualization.

Gate initialization and persistence. At the start of each rollout, both gates are closed and their consecutive-pass and consecutive-fail counters are zero. The intensity history is empty, and $\alpha_0 = 0$.

Each gate has fixed positive integer thresholds for the numbers of consecutive passing and failing checks required to open and close it, respectively. A passing check increments the consecutive-pass counter and clears the consecutive-fail counter; a failing check does the converse. A closed gate opens once its opening threshold is reached, and an open gate closes once its closing threshold is reached. Otherwise, its state is retained. An open gate can therefore tolerate an isolated failed check when its closing threshold exceeds one.

Ordered gate updates. At each decision, we first update the support gate using $m_t \geq \tau_m$ together with the minimum triple-count checks. If the updated support gate is closed, we close the failure gate and reset both of its counters. Otherwise, we update the failure gate using the strict comparison in Eq. (8): $D_F(\mathbf{v}_t) < D_S(\mathbf{v}_t)$. Equal distances count as a failed check.

We search for a correction only when both updated gates are open; otherwise, $\alpha_t = 0$. The search thus depends on the persistent gate states, not solely on the current checks. In particular, an isolated

failed support check does not reset the failure gate unless it closes the support gate. Gate thresholds remain fixed throughout evaluation.

Candidate set. We evaluate coefficients on a uniform signed grid:

$$\mathcal{A} = \{-A_{\max} + j\delta_\alpha : j = 0, \ldots, 2A_{\max}/\delta_\alpha\}, \qquad A_{\max} = 10, \quad \delta_\alpha = 0.5. \tag{20}$$

Thus, $\mathcal{A} = \{-10, -9.5, \ldots, 0, \ldots, 9.5, 10\}$ contains 41 candidates, including the zero edit.

Improvement target and maximum-progress selection. For each candidate $\alpha \in \mathcal{A}$, we evaluate its predicted effect using Eq. (9) and compute its progress score $\Delta_t(\alpha)$ using Eq. (10). When the gates permit intervention, we select a maximizing candidate:

$$\alpha_t \in \arg\max_{\alpha \in \mathcal{A}} \Delta_t(\alpha). \tag{21}$$

Observed feedback. After applying the selected nonzero correction using Eq. (2), we re-encode the edited hidden states and replace the current history entry with the measured intensity from Eq. (3).

D Experimental Protocol and Hyperparameters

Simulated Fine-tuning and Evaluations. Both $\pi_{0.5}$ and Fast-WAM are fine-tuned on 500 successful demonstrations per task collected by scripted policies. All compared methods use the same fine-tuned checkpoint for a given backbone and evaluation setting. The action-model parameters remain frozen during steering and evaluation.

Real-Robot Fine-tuning and Evaluations. The real-robot $\pi_{0.5}$ policy is fine-tuned using 30 demonstrations per task, and each method is evaluated over 30 trials per task. These counts are separate from the simulation budgets above.

Online reference and evaluation rollouts. Each task uses 24 outcome-labeled online reference rollouts, with at least two successful and two failed episodes. These policy rollouts are distinct from the scripted demonstrations used for fine-tuning. Evaluation uses 200 held-out online rollouts per task and method, separate from the reference set. SAE training uses hidden activations from the online rollout data collected at a budget of 24 episodes per task.

Shared SAE training. We train TopK sparse autoencoders on hidden activations pooled across multiple tasks, using 24 online rollouts per task. Each SAE uses an expansion factor of 8 and retains the 100 largest encoder activations per token. Writing d for the hidden-state dimension and K for the dictionary size as in Section 3, these settings correspond to $K = 8d$ and $k_{\mathrm{SAE}} = 100$. The SAE is shared across tasks and remains frozen during feature selection and steering. The TopK sparsity setting specifies the number of retained code entries; task adaptation selects one decoder direction for steering.

Shared Control Settings. Table 7 lists the control settings. The same values are used across tasks. The coefficient grid and maximum-progress selection are specified in Appendix C.

Table 7: Shared SteerAct control hyperparameters.

Parameter	Symbol	Value
Support samples	M	500
Support-gate threshold	τ_m	0.65
Maximum coefficient magnitude	$A_{\max}$	10
Coefficient grid spacing	δ_α	0.5

E REPRESENTATION AND INTERVENTION-SITE ABLATIONS

We describe the variants in Table 5(a). The alternative descriptors summarize different properties of feature-intensity histories, including their distributions, increments, event structure, and temporal dependence. These are standard families of time-series statistics (Fulcher & Jones, 2014), but the specific combinations below are constructed for our ablations rather than reproductions of a single published method.

Base. The original frozen action model executes without activation steering. This reference measures whether each intervention variant improves upon the unmodified policy.

Marginals. This variant primarily summarizes the distribution of activation intensities. Given the transformed intensity history $x_i = \log(1 + y_i/s_f)$, it concatenates histogram-bin frequencies, quantiles, the mean, standard deviation, and median absolute deviation. It also includes the fractions of measurements below a low-intensity threshold and above a high-intensity threshold, together with the fraction of near-zero one-step increments. The representation therefore emphasizes activation levels and dispersion, while retaining a coarse measure of local stability. It does not encode the distribution of ordinal patterns.

Increments. This variant represents changes in activation intensity at multiple fixed lags. For each lag ℓ, it forms $\delta_i^{(\ell)} = x_{i+\ell} - x_i$ and summarizes these increments using histogram-bin frequencies, quantiles, the fractions of positive, negative, and near-zero changes, and the mean absolute change. A validity indicator records whether the prefix contains enough measurements for that lag. The lag-specific summaries are concatenated. Unlike ordinal patterns, this representation explicitly retains the magnitudes of activation changes.

Events. This variant summarizes threshold-defined activation events. Using fixed low and high intensity thresholds, it measures the fraction of time above each threshold, activation-onset frequency, and the mean and maximum lengths of consecutive above-threshold runs. Run lengths and event counts are normalized by prefix length. Additional statistics describe near-zero increments, local peak frequency and amplitudes, and the maximum and mean intensity. These events are defined from the activation signal itself, not from manually annotated behavioral phases. The representation emphasizes activation bursts, persistence, and transitions rather than local ordinal-pattern frequencies.

Autocorrelation. The variant labeled *Autocorr.* combines autocorrelation and spectral statistics. It computes autocorrelations of the mean-centered history at fixed lags, with indicators for valid estimates. It also applies a Hann window and summarizes the resulting Fourier power spectrum through normalized spectral entropy and relative power in fixed frequency bands. Additional statistics describe increment magnitude and variability, turning-point frequency, and the mean and standard deviation of intensity. Thus, this variant captures repeated fluctuations and longer-lag dependence; it is not an autocorrelation-only representation.

Raw NN. This control performs nearest-reference matching directly on feature-intensity histories instead of comparing fixed-dimensional statistical descriptors. It tests whether sequence-level matching provides a sufficient reference signal without summarizing the history through ordinal patterns or other handcrafted statistics.

Raw DTW. This control matches feature-intensity histories using dynamic time warping (DTW) (Sakoe & Chiba, 1978). DTW computes a sequence-matching cost while allowing monotonic, nonlinear temporal alignment between histories. This comparison tests whether explicitly accommodating differences in execution timing can replace matching temporal-pattern summaries.

Enc and Dec. These variants use the ordinal-pattern representation underlying SteerAct, based on local orderings and permutation entropy (Bandt & Pompe, 2002). Enc intervenes in the action model's context branch, whereas Dec intervenes in its action expert. Both use the reference-guided steering formulation; the comparison evaluates the effect of the intervention site. Enc is the default single-feature configuration reported in the main experiments.

F Qualitative Paired Simulation Rollouts

Figures 6 and 7 jointly present a single collection of paired qualitative comparisons on RoboTwin 2.0 and DexJoCo. For each task, the upper row executes the frozen backbone without intervention (*No steering*), while the lower row applies SteerAct from a matched initial scene. Frames progress from left to right and summarize the resulting closed-loop executions. These pairs are selected failure-to-success examples intended to expose behavioral differences.

G Limitations.

Our evidence that intervention timing matters comes from gate and fixed-delay ablations, not from a controlled experiment that isolates behavioral phase from instantaneous state. Gains do not close backbone-dependent gaps (e.g., Hanoi Bi remains at 0%), and the method requires hidden-state access and a pretrained SAE whose quality bounds the available steering directions.



Figure 6: Paired simulated rollouts on RoboTwin 2.0 and DexJoCo. Each panel compares the unsteered frozen backbone (top) with SteerAct (bottom) from a matched initial scene; time advances from left to right.



Figure 7: Paired simulated rollouts on RoboTwin 2.0 and DexJoCo.(Cond.) The unsteered and steered rows begin from matched initial scenes, and frames advance from left to right.