arXiv:2610.25632v1 [cs.RO] 19 Oct 2026

STEERACT: UNLOCKING ACTION MODELS THROUGH FEW-SHOT TEMPORAL STEERING

Kaichen Zhou[1,2,*] **Zhiyan Li**[2,*] **Chongkai Gao**[3] **Yuhan Wang**[4]
Wenlong Huang[3] **Maggie Wang**[3] **Fangneng Zhan**[1] **Manling Li**[5]
Qianqian Wang[2] **Mac Schwager**[3] **Zhuang Liu**[6]
Li Fei-Fei[3] **Mengyu Wang**[2,†] **Paul Pu Liang**[1,†]

[1]Massachusetts Institute of Technology [2]Harvard University
[3]Stanford University [4]Imperial College London
[5]Northwestern University [6]Princeton University
[*]Equal contribution [†]Equal supervision

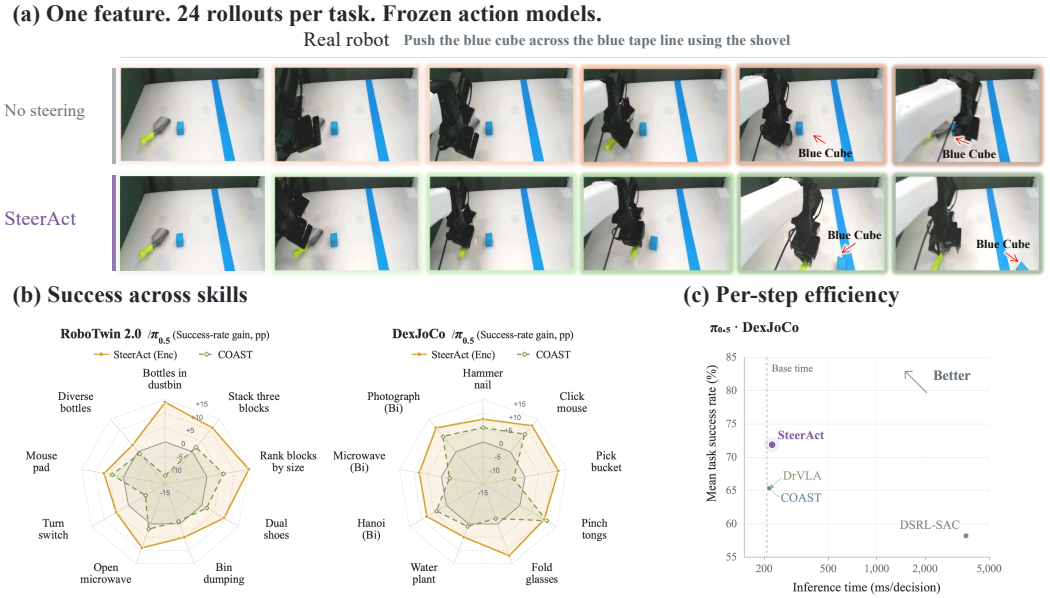


Figure 1: **Steering one feature at the right time.** Using just 24 rollouts per task, SteerAct improves robot behavior by steering a single feature activation, without retraining the action model. As the robot acts, SteerAct compares the patterns of how this feature activation changes with patterns from those existing rollouts to decide when to steer and how to steer. **(a)** Unsteered and steered real-robot executions illustrate how SteerAct improves task completion. **(b)** SteerAct improves task success across all 18 evaluated RoboTwin 2.0 and DexJoCo tasks with $\pi_{0.5}$. **(c)** On DexJoCo with $\pi_{0.5}$, SteerAct achieves the highest mean success among the evaluated methods.

ABSTRACT

Improving pretrained robot action models at test time from only a few rollouts remains challenging. Activation steering offers a lightweight approach by modifying feature activations while keeping model weights fixed. However, effective steering requires identifying which features to modify and determining the timing and strength of modification. We introduce SteerAct, a few-shot temporal steering framework that uses a shared temporal descriptor of sparse-feature dynamics for offline feature selection and online control. A pretrained sparse autoencoder provides candidate features, and our temporal descriptor summarizes the distribution of local activation-order patterns for each feature, enabling comparisons across rollouts without requiring timestep alignment. Before steering begins (Offline), we use successful and failed reference rollouts to select a single task-specific feature

whose temporal descriptors best distinguish the two outcomes and construct its reference banks. During each new execution (Online), we compare the feature's evolving temporal descriptor with these references to determine when to intervene and how strongly to increase or decrease its activation via a steering coefficient. Using only 24 outcome-labeled rollouts per task, SteerAct improves task success by approximately 10% relative to unsteered models in evaluations spanning $\pi_{0.5}$, Fast-WAM, RoboTwin 2.0, DexJoCo, and real-world manipulation, without updating action-model weights. Code and Demo can be found at: Link.
Keywords: Robot foundation models; Vision–language–action models (VLA); World-action models (WAM); Activation steering; Sparse autoencoders (SAEs); Few-shot adaptation.

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 rollouts 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 we use a few successful and failed rollouts to identify and steer a single internal feature, enabling more reliable execution by a frozen action model?*

Inference-time steering offers a way to modify action generation while keeping the pretrained model frozen. Existing approaches modify the initial noise of diffusion or flow policies (Wagenmaker et al., 2025; Tang et al., 2026), or intervene on internal activations and subspaces (Häon et al., 2025; Swann et al., 2026; Miao et al., 2026). For feature-based steering, two connected questions arise: which features should be selected from a few rollouts, and when and how should their activations be modified during a new execution? Addressing both requires a representation of feature activity that supports outcome-based feature selection and intervention decisions that adapt to the current rollout.

Our insight is that the temporal structure of feature activity can reveal outcome-related differences that activation magnitude alone misses. A feature can have similar average activation in successful and failed rollouts yet differ in how its activity rises, falls, or fluctuates (Figure 2). We refer to the relative ordering of activation values within short windows as *temporal activation patterns*. Their distributions characterize local feature dynamics without requiring timestep alignment across executions, providing a common basis for identifying useful steering features and interpreting their activity during deployment.

Building on this insight, we introduce SteerAct, a few-shot temporal steering framework that connects offline feature selection with online intervention through a shared temporal descriptor (Figure 2). 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. Before steering begins (offline), we construct success and failure reference banks from each candidate feature's temporal descriptors and select the feature with the strongest outcome separation. During execution (online), we compare the observed feature history with the selected banks to determine when correction is needed and how to adjust the feature's activation, without updating the action model's weights. Our contributions are summarized as follows:

- **Few-shot temporal steering of frozen action models.** We introduce SteerAct, a framework for improving pretrained robot action models from a few successful and failed rollouts through temporally guided interventions. The action model remains frozen, and no auxiliary reinforcement-learning controller is trained.
- **A shared temporal descriptor for feature selection and intervention.** We represent sparse-feature dynamics through distributions of temporal activation patterns. The same descriptor supports outcome-guided feature selection offline and reference-based control of intervention timing, and steering coefficient online.
- **Evaluation across action models and manipulation settings.** Experiments with $\pi_{0.5}$ and Fast-WAM on RoboTwin 2.0 and DexJoCo, together with real-world manipulation,

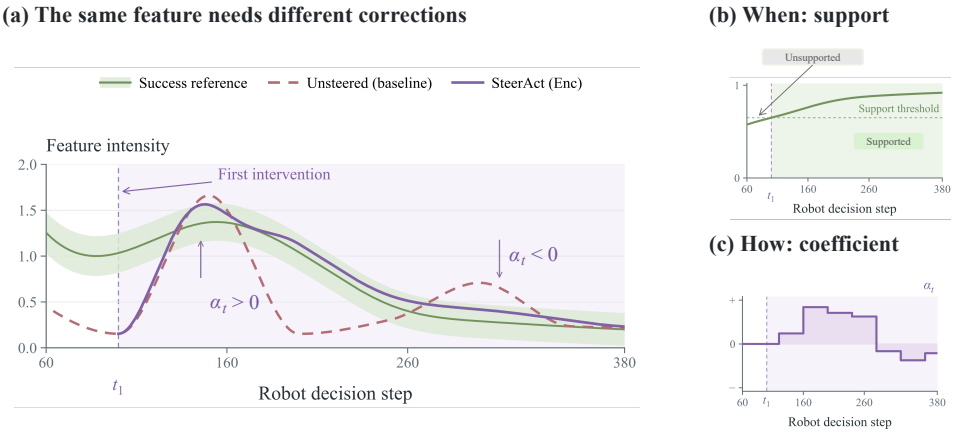


Figure 2: **Temporal evidence guides when and how to steer. (a)** The selected feature's activation traces compare a successful reference, an unsteered rollout, and rollouts with adaptive steering. **(b)** The temporal-support score measures whether the observed activation history of current rollout provides sufficient evidence for comparison with patterns from successful and failed reference rollouts. **(c)** The steering coefficient α_t determines the direction and strength of the activation edit; $\alpha_t = 0$ indicates no intervention at that decision.

demonstrate an approximately 10% relative improvement in task success over unsteered models using 24 outcome-labeled reference rollouts per task.

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 temporal 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 is a few-shot temporal steering framework that uses a shared representation of feature dynamics for offline feature selection and online intervention (Figure 3). A pretrained SAE decomposes the action model's hidden activations into sparse features, each associated with a decoder direction for intervention. A temporal descriptor summarizes each feature's activation history, placing complete reference histories, observed execution prefixes, and candidate-edited histories in the same representation space. Before steering begins (offline), differences between successful and failed references identify task-specific steering features. During execution (online), comparisons in this space determine whether intervention is warranted and guide the steering coefficient. The SAE is shared across tasks, and both its weights and the action model's weights remain frozen throughout. The complete procedure is summarized in Algorithm 1 in the appendix.

3.1 PROBLEM SETTING

We consider a frozen robot action model with access to its hidden activations at a fixed intervention site. A pretrained sparse autoencoder (SAE), shared across tasks, encodes a hidden-state vector $\mathbf{h} \in \mathbb{R}^d$ into a sparse code $\mathbf{z} \in \mathbb{R}^K$ and reconstructs it through 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. In the reconstruction, each scalar activation $\mathbf{z}[f]$ weights a decoder column $\mathbf{d}_f = W_{\text{dec}}[:, f] \in \mathbb{R}^d$. Thus, a feature f consists of an input-dependent activation and a fixed direction in the model's hidden space. We use its activation history to guide intervention and its decoder direction to modify the original hidden state.

Given a few successful and failed reference rollouts, we select one task-specific feature f^* offline and fix its steering direction $\mathbf{d} = \mathbf{d}_{f^*}$. During a new rollout, let $\mathbf{h}_{t,j}$ denote the hidden-state vector of targeted token j at robot decision t. Here, j indexes the tokens selected for intervention, while t indexes robot decisions, not iterations within the action sampler. Steering applies the same direction and steering coefficient to each targeted token:

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

Our online goal is to choose steering coefficient α_t using only the activation history observed so far. The selected direction remains fixed, and neither the action model nor the SAE is updated.

3.2 TEMPORAL DESCRIPTOR

Our representation captures the distribution of local temporal patterns in feature activity, rather than its average magnitude. For feature f in rollout e, we encode the measured hidden-state vectors with the SAE and define its intensity at decision t as

$$y_{f,e,t} = \frac{1}{N_{e,t}} \sum_{j=1}^{N_{e,t}} \max\left\{ [E(\mathbf{h}_{e,t,j})]_f , 0 \right\}. \tag{3}$$

Here, $N_{e,t}$ is the number of measured tokens, and $[E(\mathbf{h}_{e,t,j})]_f$ is the activation of feature f obtained by encoding token j's hidden-state vector. Thus, intensity measures feature activity through the SAE encoder, whereas steering modifies the hidden states from which this activity is measured. For the selected feature in the current rollout, we write $y_t = y_{f^*,e,t}$. We summarize the observed

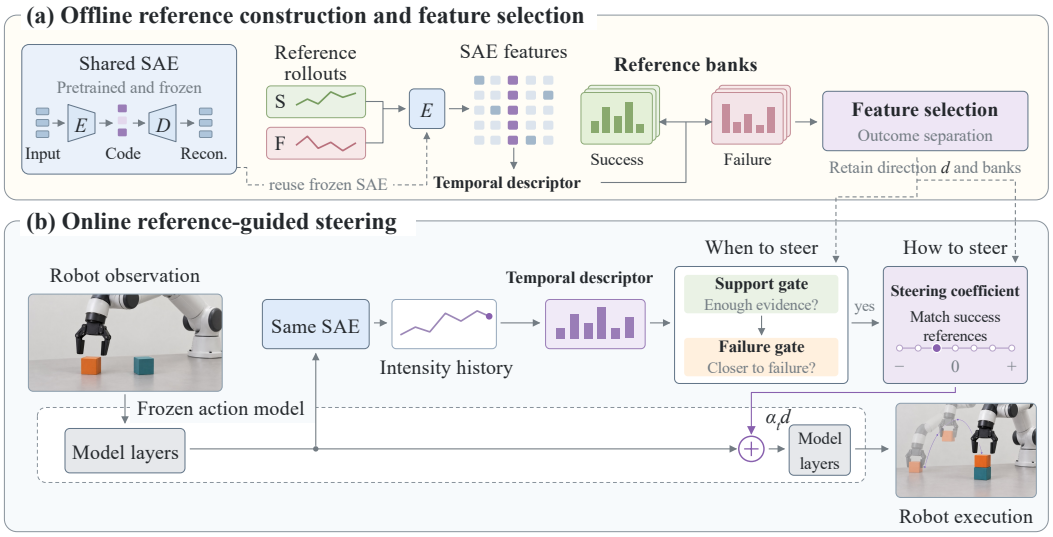


Figure 3: **SteerAct overview.** Offline, an SAE fitted to frozen-policy activations exposes candidate directions. Outcome-labeled rollouts determine a feature and its success/failure banks of ordinal dynamics. Online, supported failure-like prefixes enter a signed coefficient search; other prefixes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

intensity history using ordinal patterns (Bandt & Pompe, 2002). Omitting feature and rollout subscripts for readability, we form all available triples within $y_{1:t}$ at two spacings: (y_i, y_{i+1}, y_{i+2}) and (y_i, y_{i+2}, y_{i+4}). Each triple is assigned to one of six patterns according to the relative ordering of its values. Ties are resolved by temporal order, with earlier measurements ranked first.

For each spacing, we count the triples assigned to each pattern. Dividing these six counts by the total number of triples gives the empirical pattern distribution. We combine its six frequencies with three additional statistics: entropy, the fraction of triples containing ties, and an indicator of whether any triple is available. These nine statistics per spacing form an 18-dimensional descriptor:

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

The descriptor is used to compare temporal behavior across rollouts. We apply the same mapping ϕ to complete reference rollouts, observed execution prefixes, and histories modified by candidate corrections. This connects offline feature selection with online steering without requiring timestep alignment across rollouts.

3.3 Offline Reference Construction and Feature Selection

Before steering begins (offline preparation), we use a few rollouts collected on the target task to identify a feature whose temporal patterns distinguish successful from failed executions. For each candidate $f \in \mathcal{C} \subseteq \{1, \ldots, K\}$, we compute one descriptor from each complete reference rollout and group the descriptors by outcome (Success and Failure). This produces a success bank $\mathcal{B}_f^S$ and a failure bank $\mathcal{B}_f^F$. For outcomes $A, B \in \{S, F\}$, let $\mathbf{v}_{f,i}^A$ denote the i-th descriptor in bank $\mathcal{B}_f^A$. We define $\bar{d}_f^{AB} = \text{mean}_{i,j} \left\| \mathbf{v}_{f,i}^A - \mathbf{v}_{f,j}^B \right\|_2$, where the mean includes all reference pairs, excluding $i = j$ when $A = B$. Thus, $\bar{d}_f^{SF}$ measures between-outcome separation, while $\bar{d}_f^{SS}$ and $\bar{d}_f^{FF}$ measure within-outcome variability. We select the feature with the greatest separation between these banks, accounting for variability within each outcome:

$$f^* = \arg\max_{f \in \mathcal{C}} \left(2\,\bar{d}_f^{SF} - \bar{d}_f^{SS} - \bar{d}_f^{FF} \right). \tag{5}$$

Success–failure distance · Selected feature · Within-failure distance · Within-success distance · Candidate feature · Candidate feature set

Selection therefore depends on differences in temporal patterns, not merely differences in average activation. We retain the selected feature's decoder direction $\mathbf{d}$ and its reference banks, $\mathcal{B}_S = \mathcal{B}^S_{f^*}$ and $\mathcal{B}_F = \mathcal{B}^F_{f^*}$. All three remain fixed during deployment. The same references used to select the feature now provide the basis for deciding when and how to steer it.

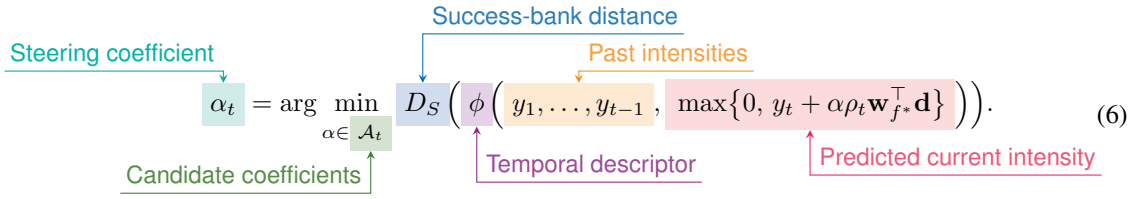
3.4 Online Reference-Guided Steering

During execution (Online), we monitor the selected feature and adapt only its steering coefficient α_t. At decision t, we append the current, unedited intensity y_t to the recorded history and compute $\mathbf{v}_t = \phi(y_{1:t})$. We first assess whether this history supports a meaningful reference comparison, then whether it resembles failure. Only when both gates are open do we search for a correction.

Support Gate. A short history can produce uncertain pattern frequencies even when its descriptor $\mathbf{v}_t$ is close to a reference. In order to solve this problem, we propose a support score. We draw M plausible temporal descriptors from a Dirichlet model of the observed pattern counts to quantify uncertainty in the current activation history. For each sampled descriptor, we count the reference descriptors in $\mathcal{B}_S \cup \mathcal{B}_F$ whose unscaled Euclidean distance from that sample is at most the offline-calibrated radius ϵ_{sup}. A sample is supported when this count is at least k_{sup}. The support score m_t is the fraction of the M sampled descriptors that are supported. The support check passes when $m_t \geq \tau_m$. Sampling and calibration details are provided in Appendix B.1.

Failure Gate. Passing the support gate doesn't mean that a correction is needed. Let $D_S(\mathbf{v})$ and $D_F(\mathbf{v})$ denote the mean scaled Euclidean distances from descriptor $\mathbf{v}$ to its nearest k_{sup} references in the success and failure banks, respectively. The failure check passes when $D_F(\mathbf{v}_t) < D_S(\mathbf{v}_t)$; equal distances do not pass. To avoid rapid switching, each gate requires consecutive passing checks to open and consecutive failing checks to close. Steering is enabled only when both gates are open; otherwise, $\alpha_t = 0$. The required check counts and gate-reset rules are specified in Appendix B.2.

Steering Coefficient. When both gates are open, we evaluate a finite coefficient grid $\mathcal{A}_t$ within fixed bounds, containing positive and negative values as well as zero. Let $\mathbf{w}_{f^*}$ denote the selected feature's encoder weight vector, and let ρ_t be the fraction of measured tokens on which that feature has positive activation. Assuming a locally unchanged SAE active set, we select the coefficient whose predicted temporal descriptor is closest to the success bank:



(6)

We apply the selected correction using Eq. (2). Approximation details and numerical safeguards are provided in Appendix B.3.

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

Table 1: **SteerAct improves success across backbones and manipulation benchmarks.** Among the evaluated methods, SteerAct (Enc) achieves the highest mean success in all four benchmark–backbone settings with modest inference overhead. Success rates are reported in percent (↑); each Mean averages nine tasks. Time reports per-decision inference latency (ms, ↓), shared across datasets. Enc/Dec denote encoder-branch/action-expert steering. Gray highlights SteerAct (Enc); **bold**/underline mark 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

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 reference rollouts. Experiments 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 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. 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 hyperparameters are listed in Appendix C. Ablation studies supporting our 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

Table 2: **SteerAct improves success on all three real-robot tasks.** With $\pi_{0.5}$ frozen, SteerAct (Enc) achieves the highest success on every evaluated task, reaching 52.2% average success compared with 31.1% for Base and 40.0% for the strongest baseline, DrVLA. Values are success rates (%, ↑). Enc/Dec denote encoder-branch/action-expert steering. Gray highlights SteerAct (Enc); **bold**/underline mark the best/second-best distinct results in each row.

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

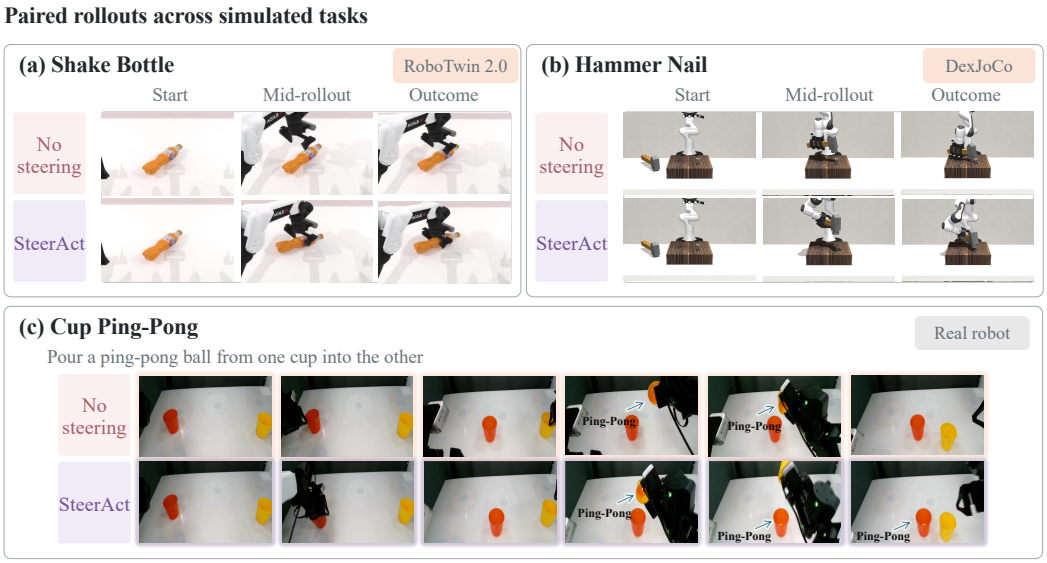


Figure 4: **Single-feature steering improves task completion across manipulation settings.** Representative unsteered and steered rollouts on RoboTwin 2.0, DexJoCo, and real robots illustrate how SteerAct corrects execution errors and enables successful task completion while keeping the underlying action model frozen.

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.

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.

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	<u>73.9</u>	72.2	71.7	70.9	**77.7**

(b) Total rollouts

Rollouts	Success
12	<u>73.6</u>
48	**77.3**
96	**77.3**

(c) Successful rollouts

Rollouts	Success
8	74.4
24	<u>77.6</u>
48	**77.9**

(d) Steering choices

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

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 the action model's 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 D).

More reference rollouts do not guarantee higher success. With 12 total reference 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, 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 reference 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 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.

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	<u>29.88</u>	22.38	**36.13**

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.

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/ICRA55743.2025.11127931. 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.

Kaichen Zhou, Yuzhen Chen, Fangneng Zhan, Hang Hua, Grace Chen, Xinhai Chang, Ao Qu, Yilun Du, Zhuang Liu, Paul Pu Liang, et al. Gem-4d: Geometry-enhanced video world models for robot manipulation. In *European Conference on Computer Vision*, pp. 130–147. Springer, 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 **16**

B Additional Details of Reference-Guided Steering **17**

B.1 Details of the Support Gate 17

B.2 Details of the Failure Gate 18

B.3 Details of the Steering Coefficient 18

C Experimental Protocol and Hyperparameters **19**

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

E Qualitative Paired Simulation Rollouts **21**

F Limitations. **21**

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

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	Drinking	Chemistry 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**

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; Zhou et al., 2026). 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.

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. (3) and Eq. (4).
Select f^* and its decoder direction $\mathbf{d}$ using Eq. (5).
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, $\alpha_0 \leftarrow 0$, 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 Sec. 3.4; update the temporal support gate using $m_t \geq \tau_m$.
    If support is open, update the failure gate using Sec. 3.4; otherwise reset it and its counters.
    $\alpha_t \leftarrow 0$.
    if both gates are open **then**
        $\alpha_t \leftarrow \textsc{BoundedSearch}(y_{1:t}, \mathbf{Z}_t, \alpha_{t-1})$, evaluating candidates with Eq. (6).
    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. (12).
    end if
    Retain the applied α_t for the next decision.
    Resume the model from the resulting hidden states and execute its action output.
end for

B Additional Details of Reference-Guided Steering

This appendix provides the implementation details of the Support Gate, Failure Gate, and Steering Coefficient introduced in Section 3. All three use the temporal descriptor defined in Eq. (4) and the reference banks retained after feature selection in Eq. (5). The banks contain descriptors of complete reference rollouts and remain fixed during deployment. Feature selection requires at least two reference rollouts per outcome because its within-bank distances exclude self-pairs.

B.1 Details of the Support Gate

The Support Gate checks whether the current history provides sufficient evidence for comparison with the reference banks. This subsection expands the Support Gate in Section 3. We explain how the Dirichlet model generates the M plausible temporal descriptors used to compute the support score m_t. We specify how ϵ_{sup} is determined offline to decide which reference descriptors count as nearby.

Generating the descriptors. We use the triples (y_i, y_{i+1}, y_{i+2}) and (y_i, y_{i+2}, y_{i+4}) underlying the temporal descriptor in Eq. (4). For each spacing, we count the occurrences of the six ordinal patterns within $y_{1:t}$ and add a pseudo-count of $1/2$ to each count to parameterize a Dirichlet distribution. To generate each of the M plausible descriptors, we draw one probability vector for each spacing. Each vector contains six nonnegative probabilities that sum to one. We compute its normalized entropy, $H(\mathbf{p}) = -\sum_{i=1}^{6} p_i \log p_i / \log 6$, and combine the six probabilities and entropy with the observed tie fraction and availability indicator. Concatenating the two nine-dimensional summaries gives an 18-dimensional descriptor. The tie fractions and availability indicators remain fixed across all M descriptors. Adjacent triples share activation values. Since the Dirichlet model does not account for this dependence, its uncertainty estimates are approximate.

Computing the support score. A sampled descriptor is supported if at least k_{sup} references in $\mathcal{B}_S \cup \mathcal{B}_F$ lie within raw Euclidean distance ϵ_{sup}. The support score m_t is the fraction of the M sampled descriptors that are supported. The support check passes when $m_t \geq \tau_m$ and both spacings meet the minimum triple-count requirement. The Support Gate starts closed. It opens after a fixed number of consecutive passing checks and closes after a fixed number of consecutive failing checks; otherwise, its state is retained.

Setting the comparison radius. We determine ϵ_{sup} from the reference banks before evaluation. For each reference, we exclude the reference itself and measure its raw Euclidean distance to its k_{sup}-th

nearest neighbor in the pooled bank. We set the radius to three times a prespecified quantile of these distances. The quantile level lies strictly between zero and one. The positive neighbor count k_{sup} cannot exceed the number of other references in the pooled bank. The quantile level, neighbor count, and multiplier are fixed settings; the resulting radius depends on the reference data. It is computed once and is not updated using evaluation rollouts.

B.2 Details of the Failure Gate

The Failure Gate compares the observed descriptor $\mathbf{v}_t$ with the success and failure banks through $D_S(\mathbf{v}_t)$ and $D_F(\mathbf{v}_t)$. The same success-bank distance also scores candidate corrections in Eq. (6). We explain its calculation here so that both uses follow exactly the same comparison.

Scaling descriptor coordinates. Unlike the Support Gate, the Failure Gate and coefficient selection in Eq. (6) use scaled Euclidean distances. We compute a scale for each of the 18 coordinates of the temporal descriptor in Eq. (4). Each reference rollout contributes one descriptor to its outcome bank. For coordinate q, we compute the mean $\mu_{S,q}$ and population variance $\sigma^2_{S,q}$ of its values across descriptors in the success bank. We obtain $\mu_{F,q}$ and $\sigma^2_{F,q}$ in the same way from the failure bank. These statistics therefore describe variation across reference rollouts, not across timesteps within a rollout. Population variances use the corresponding bank size as the denominator. To give success and failure equal weight when computing the scale, regardless of bank size, we set

$$s_q = \max\left\{10^{-8}, \sqrt{\frac{\sigma^2_{S,q} + \sigma^2_{F,q}}{2} + \frac{(\mu_{S,q} - \mu_{F,q})^2}{4}}\right\}. \tag{7}$$

This scale accounts for both within-bank variation and differences between the bank means. The floor prevents division by zero for constant coordinates. All scales are computed offline and remain fixed during deployment. For a query descriptor $\mathbf{v}$ and a reference descriptor $\mathbf{u}$, let v_q and u_q denote their values at coordinate q. Their scaled Euclidean distance is

$$\text{dist}(\mathbf{v}, \mathbf{u}) = \left[\sum_{q=1}^{18} \left(\frac{v_q - u_q}{s_q}\right)^2\right]^{1/2}. \tag{8}$$

The query may describe the observed execution prefix or a history modified by a candidate correction; both use the same fixed scales.

Computing D_S and D_F. Let k be the requested number of neighbors for outcome comparison. This setting is separate from k_{sup}, which counts nearby references for the Support Gate. We use $k_{\text{eff}} = \min\{k, |\mathcal{B}_S|, |\mathcal{B}_F|\}$ to ensure that both outcome comparisons average over the same number of references. To compute $D_S(\mathbf{v})$, we find the k_{eff} nearest success references under Eq. (8) and average their distances to $\mathbf{v}$. We compute $D_F(\mathbf{v})$ identically using the failure bank. The Failure Gate's instantaneous check passes when $D_F(\mathbf{v}_t) < D_S(\mathbf{v}_t)$; equal distances do not pass.

Using the same distance for steering. In Eq. (6), each candidate coefficient produces a modified temporal descriptor. We compute its distance to the success bank using the same scales and neighbor count as above. Nearest neighbors are recomputed for every candidate; the search does not hold a single success reference fixed. Only the Support Gate uses the descriptors sampled from the Dirichlet distribution. The Failure Gate uses the observed descriptor, and coefficient selection uses the candidate-edited descriptors. Support assessment and offline feature selection continue to use raw Euclidean distances; the coordinate scaling described here applies only to outcome comparison and candidate scoring.

B.3 Details of the Steering Coefficient

Equation (6) compares candidate coefficients by estimating how each hidden-state edit changes the current feature intensity. Here, we explain the approximation used in that equation and the numerical checks applied before using it.

Predicting Edited Feature Intensity. For the selected f^*, we define the positive-activation fraction

$$\rho_t = \frac{1}{N_t} \sum_{j=1}^{N_t} \mathbf{1}\Big\{[E(\mathbf{h}_{t,j})]_{f^*} > 0\Big\}, \tag{9}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. Thus, ρ_t is the fraction of measured tokens whose activation contributes positively to the rectified intensity in Eq. (3). Output of the frozen encoder E in Eq. (1) corresponding to feature f^* is denoted by $[E(\mathbf{h})]_{f^*}$, which can be written as

$$[E(\mathbf{h})]_{f^*} = \mathbf{w}_{f^*}^\top \mathbf{h} + b_{f^*}, \tag{10}$$

where b_{f^*} is the corresponding scalar bias. For a candidate coefficient α, as illustrated in Eq. (2), the corresponding edit of each output is

$$\left[E\big(\mathbf{h}'_{t,j}(\alpha)\big)\right]_{f^*} \approx [E(\mathbf{h}_{t,j})]_{f^*} + \alpha \mathbf{w}_{f^*}^\top \mathbf{d}. \tag{11}$$

The feature intensity in Eq. (3) averages the rectified activations across the encoder's outputs. Therefore, the predicted current intensity for candidate coefficient α is

$$\widehat{y}_t(\alpha) \approx y_t + \alpha \rho_t \mathbf{w}_{f^*}^\top \mathbf{d}. \tag{12}$$

Estimating the change in feature intensity. Under an affine encoder, moving a hidden-state vector by $\alpha\mathbf{d}$ changes the selected feature's pre-activation by $\alpha \mathbf{w}_{f^*}^\top \mathbf{d}$. If the SAE active set remains unchanged locally, tokens on which the feature is positive contribute this response, while inactive tokens contribute zero. The intensity in Eq. (3) averages rectified activations across measured tokens. When the measured and edited token sets are identical, averaging therefore gives the estimated intensity change $\alpha\rho_t \mathbf{w}_{f^*}^\top \mathbf{d}$, where ρ_t is the fraction of measured tokens on which the feature is positive. This explains the predicted current intensity $\max\{0, y_t + \alpha\rho_t \mathbf{w}_{f^*}^\top \mathbf{d}\}$ inside Eq. (6). The feature-dependent response is fixed offline; under the assumptions above, it is $\mathbf{w}_{f^*}^\top \mathbf{d}$. Only the active-token fraction ρ_t changes with the current hidden activations. The dot-product expression is not an identity for arbitrary SAE encoders or edits that change which features are active.

Evaluating candidate corrections. Each candidate replaces only the current intensity in the history used by ϕ; earlier entries remain unchanged during candidate evaluation. The resulting descriptor is compared with the success bank using D_S, as defined above. This calculation evaluates a local change in feature statistics rather than predicting future actions or task success. We abstain if the estimated intensity change per unit coefficient is nonfinite or its absolute value is below the configured minimum. Otherwise, coefficient selection follows Eq. (6), and the chosen coefficient modifies the targeted hidden states through Eq. (2). As specified in the main text, steering is disabled whenever either gate is closed.

Preprocessing feature intensities. The implementation retains raw intensities and applies $\widetilde{y}_t = \log(1 + y_t/s_f)$ before constructing the temporal descriptor, where $s_f > 0$ is a feature-specific scale fixed offline. In exact arithmetic, this strictly increasing transformation preserves ordering and ties. It therefore leaves the ordinal descriptor in Eq. (4) unchanged.

The predicted intensity change is computed in the raw intensity space, before this transformation. This keeps the response approximation consistent with the intensity measured in Eq. (3).

C 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. The TopK sparsity setting specifies the number of retained code entries; task adaptation selects a decoder direction for steering.

Shared Control Settings. Table 7 lists the control settings. The same values are used across tasks.

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

D 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.

E Qualitative Paired Simulation Rollouts

Figures 5 and 6 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.

F 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 5: Simulated rollouts on RoboTwin 2.0. Each panel compares the unsteered frozen backbone (top) with SteerAct (bottom) from a matched initial scene; time advances from left to right.



Figure 6: Paired simulated rollouts on RoboTwin 2.0 and DexJoCo (continued). The unsteered and steered rows begin from matched initial scenes, and frames advance from left to right.