STEERING VISION-LANGUAGE-ACTION MODELS WITH LATENT RISK-INFORMED CONCEPTORS FOR RELIABLE EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action models enable robots to generalize across diverse tasks, yet local execution deviations can accumulate into task failure, limiting deployment reliability. Inference-time steering with conceptors can mitigate such deviations by preserving success-related representations and suppressing failure-related components without updating policy parameters. However, these methods construct failure references using rollout-level labels, potentially mixing normal and failure-related representations and leading to harmful interventions. We therefore propose Risk-Informed Conceptors, a latent steering framework that uses local failure evidence to refine the representations from which steering matrices are constructed. We leverage weakly supervised failure detection to obtain step-level risk estimates from rollout-level labels, using multiple-instance learning (MIL) with task-wise mixture-risk correction. These estimates identify failure-relevant representations for conceptor construction and intervention-layer selection. At inference time, the detector provides calibrated risk signals to determine when and how steering is applied, while keeping the VLA frozen. Extensive experiments across multiple benchmarks validate the performance of Risk-Informed Conceptors, showing consistent improvements over frozen VLAs.

1 INTRODUCTION

Vision-language-action models (VLAs) have shown promising generalization ability across diverse manipulation tasks by mapping visual observations and language instructions to actions (Kim et al., 2024; Black et al., 2024; 2025; Kim et al., 2025; Shukor et al., 2025). However, distribution shifts and compounding execution errors can still cause task failure during deployment (Gu et al., 2026; Römer et al., 2026a; Pan et al., 2026). One promising approach is to combine early failure detection with online recovery, monitoring observations or internal policy states to identify deviations and adjust subsequent actions (Gu et al., 2026; Wu et al., 2025; Zeng et al., 2026; Pan et al., 2026). Despite their effectiveness, some existing methods rely on dedicated failure-recovery data and additional training of vision-language models (VLMs), increasing data collection and training costs (Zeng et al., 2026).

Recent studies have explored latent features for runtime failure monitoring and inference-time correction (Gu et al., 2026; Römer et al., 2026a; Pan et al., 2026). Beyond monitoring, latent feature steering has shown promise in controlling generation and reducing hallucinations in large language models (LLMs) and VLMs (Liu et al., 2023; Rimsky et al., 2024; Liu et al., 2025; Wu et al., 2026). These methods derive steering directions from reference activations and utilize them to adjust hidden states during inference, without updating model parameters. Reference-based steering has also been applied to robot policies (Miao et al., 2026). Activations from successful and failed rollouts are used to fit conceptor matrices, which act as soft filters on hidden features. The resulting conceptor matrices aim to preserve success-related structure while suppressing failure-related components. However, failure rollouts may still contain normal execution. Grouping activations by rollout label can therefore produce *mixed failure references*, which combine both normal and failure-related patterns (Figure 1). Conceptor matrices built from such references may suppress useful representations and interfere with normal behavior. This raises a key question: Which local activations should serve

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

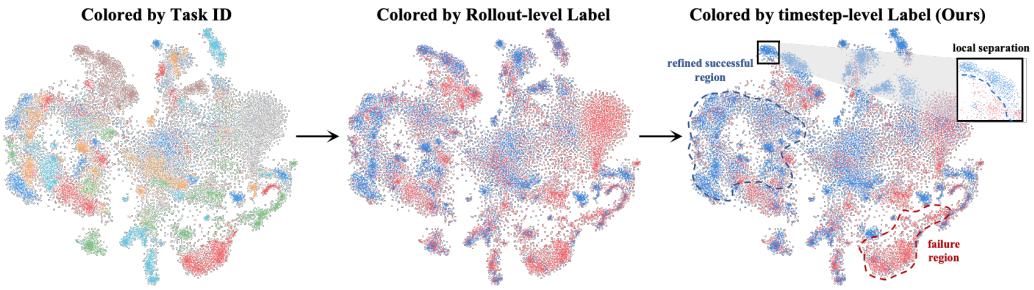


Figure 1: t-SNE plots of the VLA's internal features. From left to right, the same VLA features are colored by task ID, rollout-level labels, and refined timestep-level labels.

as failure references for effective corrective steering? Appendix A reviews related work on failure detection and latent steering.

To address this issue, we aim to identify local failure evidence within rollouts and refine steering references. Inspired by non-negative risk estimation in positive-unlabeled learning (nnPU) (Kiryo et al., 2017), we model local representations from failure rollouts as an unlabeled mixture of normal and failure-related components, using task-wise successful rollouts as a normal reference. This formulation allows reference selection to account for normal execution within failure rollouts.

Guided by these insights, we propose *Risk-Informed Conceptors*, a weakly supervised framework that uses local failure evidence to guide latent steering. We train a detector with negative–unlabeled multiple-instance learning (NU-MIL), combining rollout-level classification with task-wise mixture-risk correction. NU-MIL uses successful rollouts as normal references to account for normal execution within failed rollouts, allowing the detector to learn local failure scores without step-level annotations. The detector's predictions then refine failure references, which are combined with successful references to construct conceptor-based steering matrices. We further introduce an operator-aware risk criterion for layer selection. This criterion compares how candidate operators modify activations from successful and failed rollouts, linking the intervention location to the operator used for steering. At runtime, calibrated alarms from the same detector activate the selected operator, while the VLA parameters remain frozen. Detection thus goes beyond deciding when to intervene and helps determine the representations from which corrective interventions are built.

We conduct extensive experiments on LIBERO-10, LIBERO-Pro, LIBERO-Plus, and RoboCasa to evaluate our framework. Detection results show that incorporating NU-MIL into failure detection outperforms the other embedding distribution-based and classifier-based methods. Closed-loop evaluations further show that our steering framework improves task success rates over the frozen VLA baseline. Our contributions are summarized as follows:

- We propose *Risk-Informed Conceptors*, a framework that uses local failure evidence to refine steering references. Its NU-MIL detector combines rollout-level classification with task-wise mixture-risk correction to learn local failure scores without step-level annotations. The detector's predictions guide Conceptor construction as well as runtime intervention.
- We introduce an operator-aware risk criterion for layer selection. By comparing how candidate operators modify activations from successful and failed rollouts, this criterion links the intervention location to the operator used for steering.
- We evaluate failure detection across four benchmarks and closed-loop steering on LIBERO-Pro and LIBERO-Plus. NU-MIL outperforms the evaluated embedding distribution-based and classifier-based baselines, while steering improves task success rates over the frozen VLA baseline on the detector-seen subsets.

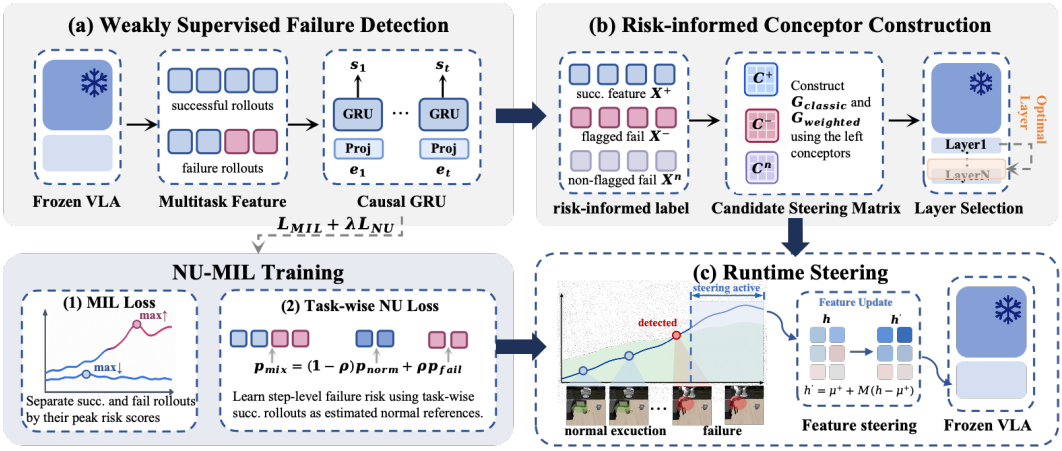


Figure 2: Overview of Risk-Informed Conceptors. (a) We train a failure detector using NU-MIL with rollout-level supervision to produce timestep-level failure scores. (b) The detector guides the refinement of failure references for constructing conceptor steering matrices, while relative modification risk guides layer selection. (c) At runtime, the first calibrated alarm triggers latent steering.

2 METHOD

Figure 2 provides an overview of Risk-Informed Conceptors, from offline reference refinement and steering operator selection to alarm-triggered intervention during deployment.

2.1 PROBLEM FORMULATION

We consider a frozen VLA policy for robotic task execution. At timestep t, the policy receives an observation o_t, consisting of RGB images, a natural language instruction, and the current robot state. Let H denote the prediction horizon and a_t an individual control action. The policy generates an action chunk $A_t = [a_t, \ldots, a_{t+H-1}]$, executes its first $H' \leq H$ actions, and replans at timestep $t + H'$. We denote the internal VLA embedding at each replanning timestep by e_t.

For each collected rollout τ_i, let k_i denote its task identity and $y_i \in \{0, 1\}$ its rollout-level label, where 0 indicates success and 1 indicates failure. Only rollout-level labels are available, without annotations of when execution errors occur. Our objective is to learn a multi-task detector that estimates a failure score s_t from the embedding history up to time t, and use the resulting local evidence to refine intervention references and guide latent steering without updating the VLA.

The key challenge is that failed trajectories may contain substantial normal execution, making their outcomes unreliable as labels for individual timesteps. We therefore model the embeddings observed at replanning timesteps in failed trajectories as a mixture of normal and failure-related execution. For task k, let $p_{\mathrm{mix},k}(e)$ denote this observed distribution, and let $p_{\mathrm{norm},k}(e)$ and $p_{\mathrm{fail},k}(e)$ denote its normal and failure-related components. With $\rho \in (0, 1)$ denoting the assumed proportion of failure-related instances, we write

$$p_{\mathrm{mix},k}(e) = (1 - \rho)\, p_{\mathrm{norm},k}(e) + \rho\, p_{\mathrm{fail},k}(e). \tag{1}$$

We use successful trajectories from the same task to estimate the normal component, assuming that they represent normal execution within failed trajectories. This yields a negative–unlabeled (NU) formulation in which embeddings from successful trajectories provide the negative reference, while those from failed trajectories form an unlabeled mixture whose local failure labels remain unobserved.

2.2 WEAKLY SUPERVISED FAILURE DETECTION

We develop NU-MIL to learn local failure evidence from rollout-level labels. NU-MIL combines rollout-level supervision with task-wise mixture-risk correction. A lightweight unidirectional gated

recurrent unit (GRU) processes the internal VLA embeddings sequentially and produces a risk score s_{it} for rollout τ_i at each replanning timestep t, using only the embedding history available up to that timestep. Architecture and training configurations are provided in Appendix C.

Let $\mathcal{S}$ and $\mathcal{F}$ denote the successful and failed training rollout sets, respectively, and let $\mathcal{L}_{\mathrm{BCE}}$ denote the standard binary cross-entropy loss. We supervise each rollout through its highest risk score $\bar{s}_i = \max_t s_{it}$, where the maximum is taken over its replanning timesteps. Expectations denote empirical rollout averages. The multiple-instance learning (MIL) objective is

$$\mathcal{L}_{\mathrm{MIL}} = \underbrace{\frac{1}{2}\mathbb{E}_{i\in\mathcal{S}}\mathcal{L}_{\mathrm{BCE}}(\bar{s}_i, 0)}_{\text{Successful-rollout constraint}} + \underbrace{\frac{1}{2}\mathbb{E}_{i\in\mathcal{F}}\mathcal{L}_{\mathrm{BCE}}(\bar{s}_i, 1)}_{\text{Failed-rollout constraint}}. \tag{2}$$

This objective suppresses risk peaks in successful rollouts and encourages at least one high-risk step in failed rollouts, without assigning a positive label to every step.

However, MIL alone provides limited supervision for distinguishing normal from failure-related timesteps within failed rollouts. Motivated by non-negative risk estimation (Kiryo et al., 2017), we use the mixture assumption in Equation 1 to account for normal execution within failed rollouts. Let $\mathcal{S}_k$ and $\mathcal{F}_k$ denote the successful and failed training rollouts for task k. For each rollout, we average the loss over replanning timesteps, with this average denoted by $\mathbb{E}_t$. We then average the resulting rollout-level losses separately over the successful set $\mathcal{S}_k$ and the failed set $\mathcal{F}_k$. Using $[a]_+ = \max(0, a)$, we define the task-wise correction loss for mixture proportion ρ as

$$\begin{aligned}\mathcal{L}_{\mathrm{NU},k} = {} & \underbrace{(1-\rho)\,\mathbb{E}_{i\in\mathcal{S}_k}\mathbb{E}_t\mathcal{L}_{\mathrm{BCE}}(s_{it}, 0)}_{\text{Normal-execution constraint}} \\ & + \underbrace{\left[\mathbb{E}_{i\in\mathcal{F}_k}\mathbb{E}_t\mathcal{L}_{\mathrm{BCE}}(s_{it}, 1) - (1-\rho)\,\mathbb{E}_{i\in\mathcal{S}_k}\mathbb{E}_t\mathcal{L}_{\mathrm{BCE}}(s_{it}, 1)\right]_+}_{\text{Corrected failure-related loss}}.\end{aligned} \tag{3}$$

We start with the loss obtained by treating all timesteps in failed rollouts as failures, then subtract the estimated contribution from normal timesteps. Let $\mathcal{K}$ contain tasks with both successful and failed rollouts in the current training batch, and let $\lambda \geq 0$ weight the correction term. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{MIL}} + \frac{\lambda}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}\mathcal{L}_{\mathrm{NU},k}. \tag{4}$$

Together, the two terms enforce rollout-level consistency and account for mixed normal and failure-related execution, allowing the detector to learn step-level risk estimates without local annotations.

After training, we freeze the detector and use successful calibration rollouts to obtain a time-dependent threshold δ_t through functional conformal prediction (Gu et al., 2026). The step-wise alarm label is

$$\widehat{z}_{it} = \mathbf{1}[s_{it} \geq \delta_t]. \tag{5}$$

Under exchangeability of calibration and new successful rollouts, the conformal threshold band provides rollout-level false-alarm control. These alarms guide failure-reference refinement and trigger runtime steering, but are not treated as ground-truth annotations of failure onset.

2.3 Risk-Informed Conceptor Construction and Layer Selection

We use timestep-level labels from the frozen detector to organize reference activations within failed rollouts and combine them with successful references to construct steering matrices. The labels determine which samples enter each reference group without changing the activations. We then compare how these matrices modify successful and failed representations to select intervention layers, and use validation performance to determine the final configuration.

Detection-conditioned reference construction. We construct references separately for each task and omit the task index below for clarity. The superscripts $+$, $-$, and n denote successful references, detection-refined failure references, and non-alarmed references from failed rollouts, respectively. Let $v_{it}^{(\ell,j)}$ denote the mean hidden representation over action tokens in rollout τ_i at replanning

timestep t, candidate layer ℓ, and denoising step j. Using the rollout outcome y_i and timestep-level alarm label $\widehat{z}_{it}$, we form

$$X_\ell^+ = \left\{ v_{it}^{(\ell,j)} \mid y_i = 0 \right\},$$
$$X_\ell^- = \left\{ v_{it}^{(\ell,j)} \mid y_i = 1,\ \widehat{z}_{it} = 1 \right\}, \tag{6}$$
$$X_\ell^n = \left\{ v_{it}^{(\ell,j)} \mid y_i = 1,\ \widehat{z}_{it} = 0 \right\}.$$

Successful rollouts contribute all replanning timesteps. Failed rollouts are divided according to the alarm label at each timestep, rather than whether an alarm has occurred earlier. The same labels are shared across candidate layers and denoising steps, while each layer uses its own activations.

For each reference group $q \in \{+, -, n\}$, we pool activations across denoising steps and assign equal weight to each sample associated with a replanning timestep and a denoising step. Let R_ℓ^q denote the covariance matrix computed after subtracting the group's own mean, normalized by its sample count. With aperture parameter $\alpha_C > 0$ and identity matrix I, the reference conceptor has the closed-form solution

$$C_\ell^q = R_\ell^q \left(R_\ell^q + \alpha_C^{-2} I \right)^{-1}, \qquad q \in \{+, -, n\}. \tag{7}$$

This gives the successful conceptor C_ℓ^+, detection-refined failure conceptor C_ℓ^-, and non-alarmed failure conceptor C_ℓ^n. The three groups are centered separately without normalizing individual activation vectors. The resulting matrices are task-specific and shared across denoising steps.

Steering matrix construction. We first combine successful and detection-refined failure references to form the Classic steering matrix. Let $\wedge$ denote conceptor AND. We define

$$G_{\text{classic},\ell} = C_\ell^+ \wedge (I - C_\ell^-). \tag{8}$$

This matrix favors directions supported by successful references but less supported by failure references. Detection therefore affects the steering matrix through the covariance of the selected failure activations.

We further use differences between alarmed and non-alarmed features within failed rollouts to adjust the suppression defined by Classic. Let $[\cdot]_+$ denote the positive spectral part of a symmetric matrix, obtained by setting negative eigenvalues to zero. We introduce a local difference matrix D_ℓ and a Classic suppression matrix B_ℓ, and construct the Classic-weighted Local steering matrix as

$$D_\ell = [C_\ell^- - C_\ell^n]_+, \quad B_\ell = [I - G_{\text{classic},\ell}]_+, \quad G_{\text{weighted},\ell} = I - B_\ell^{1/2} D_\ell B_\ell^{1/2}. \tag{9}$$

Here, $B_\ell^{1/2}$ is the symmetric positive semidefinite square root of B_ℓ. The matrix D_ℓ captures directions with greater support in alarmed than non-alarmed failure references. Weighting this difference by $B_\ell^{1/2}$ incorporates the suppression structure of Classic into the local comparison. Thus, $G_{\text{weighted},\ell}$ combines Classic's suppression structure with local alarm evidence from failed rollouts to guide more selective steering.

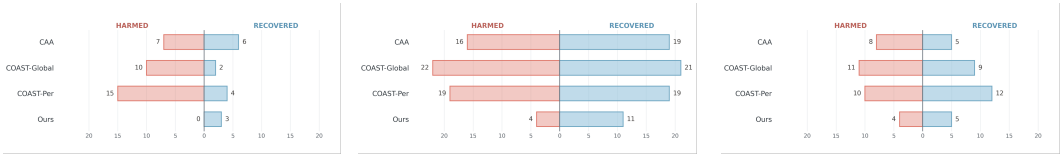
Layer selection We select an intervention layer separately for each steering matrix type and use G_ℓ to denote the matrix currently being evaluated. The reference aperture is fixed to 1 during layer selection. Let μ_ℓ^+ denote the successful-reference mean for the current task and layer, and let $\epsilon > 0$ be a numerical stabilizer. For representation v, we define the relative modification risk using the successful-reference mean

$$r_\ell(v) = \frac{\left\| (I - G_\ell)(v - \mu_\ell^+) \right\|_2^2}{\left\| v - \mu_\ell^+ \right\|_2^2 + \epsilon}. \tag{10}$$

This score measures the normalized suppression of the representation's deviation from the successful-reference mean. We evaluate it on complete rollouts held out from the corresponding reference fitting, using the same rollouts and evaluation positions across candidate layers. All valid replanning timesteps are included, without further selection based on alarm labels.

Let $\bar{r}_{i,\ell}$ denote the average risk within rollout τ_i. For task k, we compute the difference between empirical averages over failed and successful rollouts

$$J_{k,\ell} = \underbrace{\mathbb{E}[\bar{r}_{i,\ell} \mid k_i = k,\ y_i = 1]}_{\text{Failed-rollout modification risk}} - \underbrace{\mathbb{E}[\bar{r}_{i,\ell} \mid k_i = k,\ y_i = 0]}_{\text{Successful-rollout modification risk}}. \tag{11}$$



(a) LIBERO-Plus **(b) LIBERO-Pro** **(c) RoboCasa**

Figure 3: Failure recovery and success disruption across intervention methods. From left to right, results are shown for the evaluated task subsets of LIBERO-Plus, LIBERO-Pro, and RoboCasa. For rollouts with matched initial states, we count failures under the frozen base policy that each intervention turns into successes, and successes that it turns into failures.

Table 1: Steering performance comparison across benchmarks. We report the average success rate (%) and the gain Δ over the corresponding base model.

Model	**LIBERO-Plus** (π_0)		**LIBERO-Pro** ($\pi_{0.5}$)		**RoboCasa** ($\pi_{0.5}$)	
	SR ↑	Δ ↑	SR ↑	Δ ↑	SR ↑	Δ ↑
Base	48.89	–	62.67	–	41.67	–
+ CAA	47.78	−1.11	64.67	+2.00	39.17	−2.50
+ COAST-Global	40.00	−8.89	62.00	−0.67	40.00	−1.67
+ COAST-Per	36.67	−12.22	62.67	+0.00	**43.33**	**+1.67**
+ **Ours**	**52.22**	**+3.33**	**67.33**	**+4.67**	42.50	+0.83

A positive difference indicates greater relative modification of failed representations than successful ones. Let K denote the number of tasks used for layer selection. We average the differences equally across tasks and select a shared intervention layer for the current matrix type

$$\ell^\star = \arg\max_{\ell} \frac{1}{K} \sum_{k=1}^{K} J_{k,\ell}. \tag{12}$$

2.4 Runtime Steering

Following COAST (Miao et al., 2026), we construct the runtime steering matrix M by interpolating the selected steering matrix G with the identity matrix I using strength β. For an action-token hidden state h at the selected layer, we apply M to its deviation from the task-specific successful-reference mean μ^+ and restore the mean to obtain the modified state

$$M = (1-\beta)I + \beta G, \qquad h' = \mu^+ + M(h - \mu^+). \tag{13}$$

At runtime, no steering is applied before or during the first alarmed timestep. Intervention begins at the next replanning timestep. We use the first alarm to activate steering and keep it active for the remainder of the rollout. The same matrix and successful-reference mean are shared across denoising steps, while both the VLA and detector remain frozen.

3 Experiments

3.1 Experimental Setup

Benchmarks. We evaluate failure detection on LIBERO-10, LIBERO-Plus Layout, LIBERO-Pro Environment, and RoboCasa Atomic-Seen, covering layout changes, environmental variation, and everyday kitchen tasks. Detection experiments include both detector-seen and detector-unseen tasks. Given the relatively high baseline success rate on LIBERO-10, we evaluate steering on the task subsets from the other three benchmarks.

Detection baselines. We compare against nine methods in two groups. Embedding distribution-based methods include Cosine k-NN, Euclidean k-NN, Mahalanobis, PCA-KMeans, RND, and Log-

Table 2: Performance comparison on failure detection with π_0 on LIBERO-10 and LIBERO-Plus (Layout).

Method	LIBERO-10 (success rate 82.80%)						LIBERO-Plus (Layout) (success rate 77.60%)					
	Seen			Unseen			Seen			Unseen		
	AUROC	bACC	TWA	AUROC	bACC	TWA	AUROC	bACC	TWA	AUROC	bACC	TWA
Cosine k-NN	0.823	0.706	0.663	0.783	0.661	0.585	0.778	0.752	0.714	0.574	0.590	0.541
Euclidean k-NN	0.827	0.729	0.681	0.787	0.687	0.603	0.789	0.741	0.711	0.577	0.579	0.524
Mahalanobis	0.838	0.720	0.685	0.818	0.692	0.605	0.783	0.750	**0.723**	0.616	0.625	0.580
PCA KMeans	0.613	0.560	0.536	0.565	0.543	0.472	0.554	0.601	0.568	0.570	0.583	0.540
RND	0.787	0.702	0.651	0.729	0.674	0.588	0.769	0.666	0.619	0.640	0.598	0.533
LogpZO	0.852	0.747	0.706	0.801	**0.741**	0.651	0.844	0.738	0.688	0.774	0.663	0.603
SAFE-LSTM	0.681	0.704	0.664	0.800	0.632	0.585	0.801	0.669	0.657	0.794	0.683	0.632
SAFE-MLP	0.835	0.635	0.625	**0.883**	0.728	0.617	0.667	0.604	0.586	0.786	0.722	0.591
Hide-and-Seek	0.798	0.747	0.669	0.779	0.738	**0.659**	0.678	0.659	0.598	0.758	0.659	0.601
Ours	**0.857**	**0.810**	**0.719**	0.847	0.734	0.639	**0.846**	**0.756**	0.698	**0.850**	**0.770**	**0.706**

Table 3: Performance comparison on failure detection with $\pi_{0.5}$ on LIBERO-Pro (Environment) and RoboCasa (Atomic-Seen).

Method	LIBERO-Pro (Environment) (success rate 57.43%)						RoboCasa (Atomic-Seen) (success rate 50.50%)					
	Seen			Unseen			Seen			Unseen		
	AUROC	bACC	TWA	AUROC	bACC	TWA	AUROC	bACC	TWA	AUROC	bACC	TWA
Cosine k-NN	0.806	0.689	0.631	0.678	0.632	0.537	0.703	0.599	0.589	0.448	0.492	0.463
Euclidean k-NN	0.755	0.624	0.581	0.618	0.554	0.497	0.692	0.613	0.599	0.482	0.484	0.469
Mahalanobis	0.894	**0.766**	0.699	**0.845**	0.675	0.601	**0.749**	0.625	0.612	0.417	0.475	0.458
PCA KMeans	0.489	0.488	0.475	0.554	0.537	0.518	0.551	0.561	0.548	0.484	0.485	0.472
RND	0.770	0.634	0.587	0.744	0.623	0.557	0.696	0.580	0.561	0.499	0.538	0.510
LogpZO	0.804	0.682	0.625	0.704	0.647	0.606	0.685	0.566	0.558	0.445	0.484	0.466
SAFE-LSTM	0.694	0.582	0.572	0.712	0.696	0.669	0.701	0.629	0.614	**0.583**	0.525	**0.511**
SAFE-MLP	0.894	0.752	**0.700**	0.740	0.491	0.486	0.685	0.603	0.594	0.484	0.486	0.484
Hide-and-Seek	0.796	0.683	0.628	0.701	0.705	0.665	0.622	0.587	0.566	0.545	**0.546**	0.487
Ours	**0.906**	0.748	0.689	0.754	**0.732**	**0.701**	0.723	**0.643**	**0.630**	0.551	0.517	0.492

pZO. Classifier-based methods include SAFE-MLP, SAFE-LSTM (Gu et al., 2026), and Hide-and-Seek.

Steering baselines. We compare against the frozen VLA without intervention, additive steering with CAA (Rimsky et al., 2024), and two COAST variants (Miao et al., 2026). COAST-Global pools activations across denoising steps to construct one task-specific steering matrix, whereas COAST-Per constructs a separate matrix for each denoising step. CAA and both COAST variants apply steering throughout each rollout without detector gating. We reproduce them using the best configurations reported for their original settings and evaluate all methods on the same task subsets and initial states. Task selections and data splits are detailed in Appendix B; detector training and selected steering configurations are provided in Appendix C.

Evaluation Metrics. We evaluate failure detection using AUROC, the area under the receiver operating characteristic curve computed from each rollout's maximum risk score; bACC, balanced accuracy across successful and failed rollouts; and TWA, time-weighted accuracy that rewards early failure detection. Steering performance is measured by success rate (SR), the proportion of rollouts that successfully complete the task. Appendix B.3 provides the full metric definitions.

Table 4: **Ablation experiments on LIBERO-10.**

MIL	NU	AUROC	bACC
✓	×	0.802	0.747
×	✓	0.822	0.683
✓	✓	**0.857**	**0.810**

(a) Loss components

Backbone	AUROC	bACC
MLP	0.830	0.767
W-5	0.835	0.772
W-15	0.820	0.753
GRU	**0.857**	**0.810**

(b) Temporal modeling

Type	AUROC	bACC
Global	0.827	0.797
Task-wise	**0.857**	**0.810**

(c) NU conditioning

3.2 Evaluation on Steering and Detection Performance

Steering performance. As shown in Table 1, Risk-Informed Conceptors achieves relative success-rate improvements of 6.82% and 7.45% over the frozen VLA on the evaluated task subsets of LIBERO-Plus and LIBERO-Pro, respectively, outperforming all compared methods on both subsets. The aggregate across RoboCasa evaluation batches shows a relative improvement of 2.00%. The paired comparisons in Figure 3 further show that our method preserves originally successful patterns better. It preserves all originally successful rollouts on LIBERO-Plus and disrupts only 4 on each of LIBERO-Pro and RoboCasa, compared with 16–22 and 8–11 for the baselines, respectively. Some baselines may recover more failed rollouts but also disrupt more successful executions, partially or fully offsetting their recovery gains. Our method balances failure recovery and success preservation, achieving the largest gains on the two LIBERO subsets. These observations are consistent with our motivation to address mixed failure references. Rather than treating every activation in a failed rollout as failure evidence, we use local failure signals to refine the references, aiming to reduce the inclusion of normal execution components. The lower disruption of successful rollouts provides behavioral support for this design and is consistent with reducing the unintended suppression of representations needed for normal behavior.

Failure detection performance. Tables 2 and 3 compare NU-MIL with embedding distribution-based and classifier-based failure detection methods. On eval-seen tasks in LIBERO-10, NU-MIL achieves the best performance across all three metrics, improving bACC by 8.43% relative to the strongest baseline. It also achieves the highest bACC and TWA on eval-seen tasks in RoboCasa. These improvements extend to several settings with eval-unseen tasks. On LIBERO-Plus, NU-MIL improves AUROC, bACC, and TWA by 7.05%, 6.65%, and 11.71%, respectively, relative to the strongest baseline for each metric. On LIBERO-Pro, it achieves the highest bACC and TWA. Together, these results suggest that NU-MIL can learn useful failure signals from rollout-level supervision and apply them to new tasks with changes in layout or environment. However, the gains are not consistent across all benchmarks. Performance on eval-unseen RoboCasa tasks remains limited. Differences in rollout length and activation distributions, as well as reduced reliability of the calibrated thresholds on new tasks, may lead to this limitation.

3.3 Generalization to Unseen Tasks

We further evaluate our steering method on unseen tasks, distinguishing the transfer of local failure signals from the transfer of conceptors. We consider two settings, Task-specific and Transfer, to examine whether a frozen detector can support reference construction for new tasks and whether conceptors constructed from seen tasks can be applied directly to unseen ones. Seen and unseen tasks are defined relative to the detector's training tasks. The detector and conformal thresholds remain frozen in both settings.

Table 5: **Generalization of steering to detector-unseen tasks**

Model	**Task-specific**		**Transfer**	
	SR ↑	Δ ↑	SR ↑	Δ ↑
Base	35.00	–	35.00	–
+ Ours	**58.33**	**+23.33**	**61.67**	**+26.67**

Experimental setup. We use eval-seen tasks as source tasks and eval-unseen tasks as target tasks. In the Task-specific setting, each target task

has its own conceptor, fitted using reference rollouts from that task. Its steering parameters are selected on the same task's validation set. In the Transfer setting, we utilize reference rollouts from source tasks to fit a shared conceptor and compute the mean activation of successful rollouts. We use the target tasks' validation sets to choose which steering parameters to use. Target-task data are used only for this selection, not to fit the shared conceptor or compute its reference mean. In both settings, steering starts at the replanning timestep after the first alarm and remains active until the rollout ends. We evaluate on an independent set of target-task rollouts that is not used for fitting or parameter selection.

Results. Table 5 shows that both settings improve the average success rate on the evaluated target tasks. Task-specific raises it from 35.00% to 58.33%, a gain of 23.33 percentage points over the frozen base policy. Transfer achieves 61.67%, a gain of 26.67 percentage points, and outperforms Task-specific on average. On the two eval-unseen tasks, the shared conceptor achieves success rates of 66.67% and 56.67%, both above the corresponding base policy results. These results suggest that there might be a shared successful latent space across tasks. A conceptor fitted on references from eval-seen tasks can also improve performance on eval-unseen tasks.

3.4 Ablation Study on Detection Method

We examine the effects of loss components, temporal modeling, and task-wise correction on LIBERO-10. All results are averaged over five random seeds, with the conformal calibration level and mixture proportion fixed at $\alpha_{\mathrm{CP}} = 0.15$ and $\rho = 0.5$.

Loss components. Table 4(a) evaluates the contributions of MIL and NU correction. Combining both terms achieves the best results in both metrics, with relative bACC improvements of 8.43% over MIL alone and 18.59% over NU alone. MIL uses rollout-level labels to constrain the strongest failure evidence, while NU correction accounts for normal execution within failed rollouts. Their combined gains support the complementary roles of rollout-level supervision and mixture-risk correction.

Temporal modeling. Table 4(b) compares a timestep-wise MLP, fixed-window GRUs, and a standard GRU. The standard GRU achieves relative bACC improvements of 5.61% over the MLP and 4.92% over W-5, the strongest fixed-window variant. Increasing the window from 5 to 15 timesteps does not improve performance. These results support maintaining a recurrent state throughout execution rather than processing timesteps independently or restricting the context to a fixed window.

Task-wise correction. Table 4(c) compares two ways to estimate the loss contributed by normal timesteps. Global correction pools successful rollouts from all tasks, while task-wise correction uses only successful rollouts from the same task. Task-wise correction improves AUROC and bACC by 3.63% and 1.63%, respectively, relative to global correction. This suggests that successful rollouts from the same task provide a better reference for this correction than successful rollouts pooled across tasks.

4 Conclusion

In this work, we propose Risk-Informed Conceptors, a framework that connects weakly supervised failure detection with inference-time latent steering through reference refinement. NU-MIL combines rollout-level supervision with task-wise mixture-risk correction to learn local failure signals. These signals guide the selection of reference activations for conceptor construction, aiming to reduce the inclusion of normal execution components in failure references. We further select intervention layers based on the relative modification risk of the steering matrices and activate the selected matrix after the first alarm, without updating VLA parameters. Experiments show improved failure detection across multiple benchmark settings and higher success rates on the evaluated closed-loop task subsets, with fewer disruptions of originally successful rollouts than the compared steering methods. These findings support using local failure detection not only to trigger intervention but also to refine the references from which steering matrices are constructed, providing a practical approach to improving the execution reliability of frozen VLAs.

AI USE STATEMENT

We used generative AI tools for assistance with language editing, literature retrieval, and parts of code generation and debugging. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide implementation details for training and deployment in Appendix C.

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.

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 Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024.

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.

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.

Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. *Advances in neural information processing systems*, 30, 2017.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023.

Sheng Liu, Haotian Ye, and James Y Zou. Reducing hallucinations in large vision-language models via latent space steering. In *International Conference on Learning Representations*, volume 2025, pp. 72402–72419, 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.

Yi Pan, Miao Pan, Qi Lu, Jiaming Huang, Man Zhang, Siteng Huang, Xin Li, Jie Zhang, Yongliang Shen, Xuhong Zhang, et al. Vla-corrector: Lightweight detect-and-correct inference for adaptive action horizon. *arXiv preprint arXiv:2607.01804*, 2026.

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.

Carl Qi, Xiaojie Wang, Silong Yong, Stephen Sheng, Huitan Mao, Manikantan Nambi, Amy Zhang, Yeshwant Dattatreya, et al. Self-refining vision language model for robotic failure detection and reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 113934–113956, 2026.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024.

Ralf Römer, Adrian Kobras, Luca Worbis, and Angela Schoellig. Failure prediction at runtime for generative robot policies. *Advances in Neural Information Processing Systems*, 38:7631–7670, 2026a.

Ralf Römer, Maximilian Seeliger, Saida Liu, Ben Sturgis, Marco Bagatella, Daniel Marta, Andreas Krause, and Angela P Schoellig. Uncertainty quantification for flow-based vision-language-action models. *arXiv preprint arXiv:2606.18043*, 2026b.

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.

Yanchuan Tang, Taowen Wang, Yuefei Chen, Boxuan Zhang, Qiang Guan, and Ruixiang Tang. Shifting uncertainty to critical moments: Towards reliable uncertainty quantification for vla model. *arXiv preprint arXiv:2603.18342*, 2026.

Jialin Wu, Wei Shi, Han Shen, Peigui Qi, Kunsheng Tang, Zhicong Huang, Binghao Wang, and Zhou Yang. Revis: Sparse latent steering to mitigate object hallucination in large vision-language models. *arXiv preprint arXiv:2602.11824*, 2026.

Yilin Wu, Anqi Li, Tucker Hermans, Fabio Ramos, Andrea Bajcsy, and Claudia Pérez-D'Arpino. Do what you say: Steering vision-language-action models via runtime reasoning-action alignment verification. *arXiv preprint arXiv:2510.16281*, 2025.

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.

Xianchao Zeng, Xinyu Zhou, Youcheng Li, Jiayou Shi, Tianle Li, Liangming Chen, Lei Ren, and Yong-Lu Li. Diagnose, correct, and learn from manipulation failures via visual symbols. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 42386–42395, 2026.

Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6919–6929. IEEE, 2025.

Shijie Zhou, Bin Zhu, Jiarui Yang, Xiangyu Zhao, Jingjing Chen, and Yu-Gang Jiang. Rc-nf: Robot-conditioned normalizing flow for real-time anomaly detection in robotic manipulation. *arXiv preprint arXiv:2603.11106*, 2026.

A Related Work

Runtime failure detection. Runtime failure detection supports reliable robot deployment by identifying execution deviations that may lead to task failure. Existing work estimates risk from observation shifts, action uncertainty, and the distribution of successful executions (Xu et al., 2025; Römer et al., 2026a; Zhou et al., 2026), with further studies incorporating temporal information and ensemble disagreement (Tang et al., 2026; Römer et al., 2026b). VLM-based approaches introduce task semantics to assess execution progress, check constraints, and recognize failures (Agia et al., 2024; Zhou et al., 2025; Duan et al., 2024; Qi et al., 2026; Zeng et al., 2026). For VLAs, SAFE uses internal features to predict the likelihood of task failure (Gu et al., 2026). Hide-and-Seek uses contrastive learning across and within rollouts to identify failure-related timesteps and estimate failure onset from rollout-level labels alone (Park et al., 2026). Our work combines MIL with task-wise mixture-risk correction to account for normal timesteps within failed rollouts, using successful rollouts from the same task as references.

Inference-time latent steering. Inference-time latent steering guides model prediction by modifying hidden representations without updating pretrained model parameters. In LLMs, reference examples provide steering signals for task adaptation and behavior control (Liu et al., 2023; Rimsky et al., 2024). This idea has also been extended to VLMs to improve the use of visual information and reduce hallucinations (Liu et al., 2025; Wu et al., 2026). In robotics, COAST applies reference-based latent steering to VLAs, constructing conceptor steering matrices from successful and failed rollout activations to improve action generation (Miao et al., 2026). Our focus is on selecting the activations used to construct these steering matrices. Grouping activations solely by rollout outcome can place representations of normal behavior in the failure reference set, which may cause steering to suppress useful representations. We use the timestep-level failure signals learned by NU-MIL to refine this set before constructing the conceptors. This connects failure detection with steering: the detector helps determine both when to intervene and which activations to use as failure references, while the VLA remains frozen.

B Evaluation Benchmarks and Data Protocols

B.1 Failure Detection Benchmarks and Data Splits

We evaluate failure detection on LIBERO-10, LIBERO-Plus Layout, LIBERO-Pro Environment, and RoboCasa Atomic-Seen. We use official checkpoints for all benchmarks: π_0 for LIBERO-10 and LIBERO-Plus, and $\pi_{0.5}$ for LIBERO-Pro and RoboCasa.[1] All base policies remain frozen throughout our experiments. Each task contains 50 collected rollouts with rollout-level success or failure labels. Table 6(a) summarizes the datasets and task splits.

Environment settings and task coverage. We evaluate changes in object layout using LIBERO-Plus Layout and changes in the environment using LIBERO-Pro Environment. We exclude LIBERO-Pro tasks 4 and 6 because the table replacement in this setting leaves their initial object positions fixed rather than randomly sampled. We also exclude task 8, since it has only two successful rollouts in evaluation and is not available for calibration.

For RoboCasa, we select eight tasks from Atomic-Seen, including CloseFridge (0), OpenDrawer (1), OpenStandMixerHead (2), CoffeeSetupMug (3), PickPlaceCounterToCabinet (5), TurnOnElectricKettle (6), SlideDishwasherRack (7), and TurnOnSinkFaucet (8). These tasks cover opening and closing fixtures, placing objects, and operating kitchen appliances.

Data splits and evaluation scope. Seen and unseen tasks are defined relative to the detector's training tasks. The 50 rollouts of each seen task are divided into 30 training rollouts, 5 successful calibration rollouts, and 15 eval-seen rollouts. Detector hyperparameters are selected using eval-seen tasks. All 50 rollouts from each unseen task are assigned to eval-unseen and are excluded from detector training and conformal threshold estimation. The detector and thresholds remain frozen during evaluation. We use five random seeds, varying both training randomness and data splits.

[1] RoboCasa checkpoint: `https://huggingface.co/robocasa/robocasa365_checkpoints/tree/main/pi05_pretrain_human300/multitask_learning/75000`.

Table 6: **Benchmark coverage and data protocols** for failure detection and conceptor-based steering. Task IDs refer to the corresponding benchmark. Seen and unseen tasks are defined relative to detector training.

(a) Failure detection benchmarks and data sizes

Benchmark setting	VLA	Task IDs	Rollouts	Seen / Unseen tasks per split
LIBERO-10	π_0	0–9	500	7 / 3
LIBERO-Plus Layout	π_0	0–9	500	7 / 3
LIBERO-Pro Environment	$\pi_{0.5}$	0, 1, 2, 3, 5, 7, 9	350	5 / 2
RoboCasa Atomic-Seen	$\pi_{0.5}$	0, 1, 2, 3, 5, 6, 7, 8	400	6 / 2

(b) Conceptor evaluation tasks

Main comparison uses eval-seen tasks from failure detection.

Setting	Benchmark	Task IDs
Main comparison	LIBERO-Plus Layout	2, 7, 8
Main comparison	LIBERO-Pro Environment	0, 1, 2, 3, 9
Main comparison	RoboCasa Atomic-Seen	0, 1, 2, 3
Task-specific	LIBERO-Pro Environment	5, 7
Transfer	LIBERO-Pro Environment	5, 7

B.2 Conceptor Evaluation Benchmarks and Task Settings

We evaluate closed-loop steering on selected task subsets from LIBERO-Plus, LIBERO-Pro, and RoboCasa. Task selection prioritizes tasks with frozen-policy success rates below 70%, allowing us to examine correction when execution failures are frequent. We also retain several tasks with success rates above 80% to assess whether steering disrupts originally successful execution. The evaluation therefore considers both failure recovery and success preservation.

Tasks in the main comparison. The main steering comparison uses tasks seen by the frozen detector employed in the corresponding closed-loop experiments. This setting evaluates detection-conditioned reference construction and online steering on detector-seen tasks. We use tasks 2, 7, and 8 from LIBERO-Plus, tasks 0, 1, 2, 3, and 9 from LIBERO-Pro, and CloseFridge, OpenDrawer, OpenStandMixerHead, and CoffeeSetupMug from RoboCasa. Adaptation and transfer to detector-unseen tasks are evaluated separately in Table 5. Table 6(b) summarizes the evaluation tasks.

Task-specific conceptors. By default, we fit a separate conceptor for each task using only that task's reference rollouts. Only the Shared and Transfer settings combine reference rollouts from multiple tasks. In the Task-specific setting of Table 5, the detector has not been trained on the target tasks (eval-unseen). We keep the detector frozen and use its predictions to select reference activations from each target task's own rollouts, then fit a conceptor for that task. This tests whether the detector can guide conceptor construction and steering on new tasks without retraining.

Shared conceptor transfer. In the Transfer setting of Table 5, we fit one shared conceptor using reference rollouts from eval-seen source tasks. We also compute the mean activation of successful rollouts from these source tasks. We then use this conceptor and reference mean on eval-unseen target tasks. The target tasks' validation sets are used to choose the source tasks and steering parameters, but not to fit the conceptor or compute the reference mean. Thus, Task-specific fits a conceptor using the target task's own data, while Transfer fits it using data from other tasks.

In both settings, the detector and alarm thresholds remain fixed. Steering starts at the replanning timestep after the first alarm and continues until the rollout ends. The used task subsets are listed in Table 6(b).

B.3 Evaluation Metrics

We treat failed rollouts as positives and successful rollouts as negatives, denoted by $\mathcal{F}$ and $\mathcal{S}$, respectively. Here, these sets refer to evaluation rollouts rather than the training sets used in the detection objective. Let T_i denote the total number of replanning timesteps in rollout i. For each task, we use the shortest length among all loaded successful and failed rollouts as the common evaluation cutoff. The resulting window length for rollout i is denoted by H_i, with timesteps indexed from zero. A rollout is flagged as failed if $s_{it} \geq \delta_t$ at any timestep within this window, where s_{it} is the detector score and δ_t is the time-varying conformal threshold.

AUROC. Following SAFE (Gu et al., 2026), we use max-so-far (MSF) scores, which retain the highest detector score observed up to each timestep. The rollout-level score is the MSF score at the end of the evaluation window

$$\bar{s}_{it} = \max_{0 \leq u \leq t} s_{iu}, \qquad q_i = \bar{s}_{i,H_i-1}. \tag{14}$$

We compute AUROC from the continuous scores q_i and rollout-level labels. It measures the ranking of failed and successful rollouts and does not depend on the conformal threshold.

bACC. Let $\mathcal{TP}$ denote failed rollouts correctly flagged within the evaluation window, and $\mathcal{TN}$ denote successful rollouts with no alarms in that window. bACC equally weights the true-positive rate (TPR) and true-negative rate (TNR)

$$\mathrm{bACC} = \frac{1}{2}(\mathrm{TPR} + \mathrm{TNR}) = \frac{1}{2}\left(\frac{|\mathcal{TP}|}{|\mathcal{F}|} + \frac{|\mathcal{TN}|}{|\mathcal{S}|}\right). \tag{15}$$

This metric evaluates classification within the window without distinguishing earlier from later alarms.

TWA. TWA accounts for both detection correctness and alarm timing. Let t_i denote the first alarm timestep within the evaluation window for a correctly detected failed rollout. We compute

$$\mathrm{TWA} = \frac{1}{2}\left[\frac{1}{|\mathcal{F}|}\sum_{i \in \mathcal{TP}}\left(1 - \frac{t_i}{T_i}\right) + \frac{|\mathcal{TN}|}{|\mathcal{S}|}\right]. \tag{16}$$

Earlier correct alarms receive higher scores, while missed failures and false alarms receive no credit. Time weights use the full rollout length T_i, rather than the evaluation window length H_i. Normalized detection time (DT) averages t_i/T_i over failed rollouts, assigning a value of 1 to missed failures. Lower DT indicates earlier detection.

Definition of Recovered and Harmed Rollout. We compare the base policy with runtime steering using paired rollouts with the same initial states. Recovered counts rollouts that fail under Base but succeed with steering, whereas Harmed counts rollouts that succeed under Base but fail with steering.

For each seed, detection metrics are computed separately on the Seen and Unseen sets by pooling all rollouts within each set, rather than averaging task-level metrics. We use five seeds, with the conformal calibration level fixed at $\alpha_{\mathrm{CP}} = 0.15$.

C Implementation Details and Hyperparameters

Failure detection. We use the same unidirectional GRU architecture across all four benchmarks, with benchmark-specific training configurations listed in Table 7. The coefficient λ weights the NU correction term, and ρ denotes the assumed proportion of failure-related instances within failed rollouts.

Conceptor construction and parameter selection. For each reference group, we subtract the group's mean from its activation samples before fitting the conceptor. We combine samples from all replanning timesteps and denoising steps, giving each sample equal weight. Pooling across

Table 7: **NU-MIL architecture and training configurations across the four benchmarks.**

Parameter	LIBERO-10	LIBERO-Plus	LIBERO-Pro	RoboCasa
Denoising step index	9	5	5	3
Input dimension	1024	1024	1024	1024
GRU hidden dimension	256	256	256	256
Projection hidden dimension	512	512	512	512
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning rate	10^{-4}	10^{-4}	10^{-5}	10^{-5}
Weight decay	10^{-4}	10^{-3}	10^{-4}	10^{-4}
Training epochs	300	800	800	800
Configured batch size	32	32	32	32
NU weight λ	0.3	0.6	0.6	0.3
Mixture proportion ρ	0.5	0.75	0.75	0.75

Table 8: **Selected Conceptor configurations for closed-loop evaluation.** C denotes Classic and W denotes Classic-weighted Local. † indicates the Task-specific setting on detector-unseen tasks. ‡ indicates the Transfer setting, where target tasks 5 and 7 use the same shared conceptor fitted on source tasks 0 and 1, together with the same successful-reference mean and steering parameters.

(a) LIBERO-Plus

Task	Type	Layer	α_C	β
2	C	17	1	0.05
7	W	17	1	0.05
8	C	17	1	0.05

(b) LIBERO-Pro

Task	Type	Layer	α_C	β
0	C	11	0.1	0.05
1	C	11	5	0.1
2	C	11	1	0.05
3	C	11	2	0.05
9	C	11	0.25	0.05
$5^{\dagger}$	C	11	0.25	0.1
$7^{\dagger}$	C	11	1	0.3
$\{5,7\}^{\ddagger}$	C	11	1	0.3

(c) RoboCasa

Task	Type	Layer	α_C	β
0	W	11	5	0.05
1	C	11	0.5	0.05
2	C	11	5	0.05
3	W	11	10	0.3

denoising steps produces Global Conceptors, which are shared across denoising steps. Each task has its own conceptors, except in the Transfer setting, where the target tasks use the same conceptor fitted on source-task data.

We choose the intervention layer from $\{0, 5, 11, 17\}$, keeping the reference aperture fixed at 1 during layer selection. We then use validation rollouts to select the conceptor aperture $\alpha_C \in \{0.1, 0.25, 0.5, 1, 2, 5, 10\}$ and steering strength $\beta \in \{0.05, 0.1, 0.3\}$. Table 8 reports the selected settings; C denotes Classic and W denotes Classic-weighted Local.

These settings remain fixed during execution. Steering starts at the replanning timestep after the first alarm and continues until the rollout ends.

D LIMITATIONS

We evaluate our method on selected tasks in simulation. Further experiments are needed to assess its performance on a wider range of tasks and on real robots. Our detection metrics measure whether failures are detected and how early alarms are raised, but do not directly test whether high failure scores occur when failures actually begin. NU correction assumes that successful rollouts from a task can represent the normal timesteps within failed rollouts of the same task. This assumption needs to be tested across different tasks and execution conditions. We select intervention layers based on how steering changes hidden representations, without directly measuring the resulting changes in actions or task progress during layer selection. We also keep steering active from the first intervention until the rollout ends. Future work could use action changes and task progress to choose where to intervene and when to stop steering.