CASH: CALIBRATED SELF-HEALING VISION-LANGUAGE-ACTION POLICIES FOR EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Action-chunking vision-language-action (VLA) policies predict multiple future actions and execute them without incorporating new observations, local execution errors can thus accumulate over the open-loop horizon. To address this issue, we present CASH (Calibrated Self-Healing), a lightweight failure-detection and self-healing framework that keeps the VLA frozen and consists of detection and self-healing. The detection module contains an execution anomaly detector (EAD) and a policy anomaly detector (PAD). EAD tests whether an executed action is consistent with the observed latent transition. Its score combines forward prediction error, directional disagreement, and inverse-dynamics reconstruction error; a conformalized quantile threshold with a one-sided CUSUM (cumulative sum) that converts the score into a state-conditioned trigger for persistent deviations. PAD tests whether a proposed action is plausible under the current observation and instruction, using successful actions and several families of counterfactual actions to train a value model. When EAD triggers, CASH computes the latent displacement that should have occurred but has not yet occurred, and adjusts the action chunk through a lightweight dynamics model. When PAD triggers, CASH shortens the action chunk and triggers earlier re-planning; repeated low scores activate a window reset for severe out-of-distribution (OOD) conditions. The detection modules are trained on cached latent features using successful trajectories only, without failure data or manual failure labels. The self-healing module does not require additional training. CASH can be integrated into different VLA models without retraining the VLA backbone, while preserving the efficiency of action chunking.

1 INTRODUCTION

Embodied agents can perceive and interact with the physical world, and are widely viewed as a cornerstone of artificial general intelligence (Duan et al., 2022; Ma et al., 2026). Powered by large-scale vision-language pre-training, vision-language-action (VLA) models have become a practical interface between pre-trained vision-language representations and robot control, and now serve as the dominant policy backbone for embodied agents (Brohan et al., 2023; Zitkovich et al., 2023; Kim et al., 2024). A natural deployment of such a policy is a closed loop, in which a new action is predicted from the latest observation at every control step. As noted in early imitation learning studies (Zhao et al., 2023; Chi et al., 2023), this requires a full network forward pass at each control step, making it difficult for large backbones to operate at the control frequency required by the task, and restarting prediction at every step tends to produce fragmented, jerky motion rather than coherent trajectories.

Action chunking mitigates this tension by predicting several future actions at once and executing them before requesting the next observation (Zhao et al., 2023; Chi et al., 2023; Black et al., 2025b;a), which reduces inference frequency and promotes smoother motion. However, two distinct failure sources remain unaddressed. Firstly, during chunk execution, the policy does not observe the consequences of its earlier actions until the entire chunk has been executed, allowing small execution errors to accumulate into task failure. Secondly, the proposed actions can be unreasonable under the current observation and instruction, while the policy may carry out wrong actions confidently and

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.

smoothly (Agia et al., 2024; Römer et al., 2025), which instead necessitates evaluating candidate actions before execution. Reliable deployment therefore requires checking at runtime whether the executed actions are consistent with the observed state changes, together with timely recovery, while preserving the efficiency that action chunking provides.

We propose CASH, a calibrated self-healing framework for action-chunking VLA policies. CASH keeps the VLA parameters frozen and consists of a detection module and a self-healing module. The detection module contains two complementary methods: (i) The Execution Anomaly Detector (EAD) checks whether the executed actions are consistent with the observed latent state transition, and (ii) the Policy Anomaly Detector (PAD) determines whether a candidate action is reasonable under the current observation and task instruction. Both detectors are trained on latent features from successful trajectories only and require neither failure data nor manual failure labels.

The self-healing module requires no training. When EAD triggers, CASH corrects the action window according to the gap between the latent displacement expected from the executed actions and the actual latent displacement, bringing the execution trajectory back to the expected path. When PAD triggers, CASH shortens the action chunk and re-plans earlier, and if low scores recur within a short period, it performs a window reset to handle a severe distribution shift.

The main contributions of this paper are as follows:

- We propose the Execution Anomaly Detector (EAD), which formulates execution monitoring for VLA policies based on action chunk as a joint consistency checking problem.
- We propose the Policy Anomaly Detector (PAD), which evaluates the reasonableness of candidate actions under the current observation and task instruction.
- We propose a complementary self-healing mechanism for EAD and PAD, where an EAD trigger induces a correction of the action window and a PAD trigger shortens the execution horizon or resets the execution window.

2 Related work

2.1 Execution-time failure detection

Execution-time monitoring judges whether an observed transition is consistent with that implied by the commanded actions. Latent dynamics models have a long history in classical model-based control (Deisenroth & Rasmussen, 2011), underpin learned world models (Hafner et al., 2020), and are extended to action-conditioned prediction by large-scale video-pretrained world models (Assran et al., 2025). Along this line, VLA-Corrector (Pan et al., 2026) trains a latent dynamics model on successful demonstrations and uses the directional disagreement between the predicted and true latent displacement as its anomaly signal; CheckVLA (Liu et al., 2026) verifies action-conditioned outcomes with a world model and rewrites the suffix of the action chunk; Foresight (Zhang et al., 2026b) scores observed latents and action-conditioned predictions with a causal sequence model and relies only on trajectory-level success labels; foundation world model detectors (Ward et al., 2026) use prediction uncertainty in a compressed visual latent space. ProbeAct (Zhang et al., 2026a) builds no dynamics model and instead reads the policy's internal hidden states through a lightweight probe, at the cost of probe annotations and access to policy internals. FAIL-Detect (Chen et al., 2025) avoids failure labels by using the action likelihood of a conditional flow model as its anomaly score. Other work includes multitask failure detectors that transfer across tasks (Gu et al., 2025), direct use of vision-language models to detect and reason over failures (Duan et al., 2025; Lin et al., 2026), the adaptable error detector AED for few-shot imitation policies (Yeh et al., 2024), A2C2 (Sendai et al., 2025), which corrects action chunks in real time without discarding observations, and, on the recovery side, FLARE (Zhao et al., 2026), which distinguishes in-distribution retries from out-of-distribution resets and switches among multiple fine-tuned adapters.

2.2 Policy anomaly detection

Policy anomaly detection judges whether a candidate action is reasonable under the current observation and instruction. Recent runtime work evaluates the action itself rather than only its outcomes. Sentinel (Agia et al., 2024) divides policy failures into action temporal inconsistency and

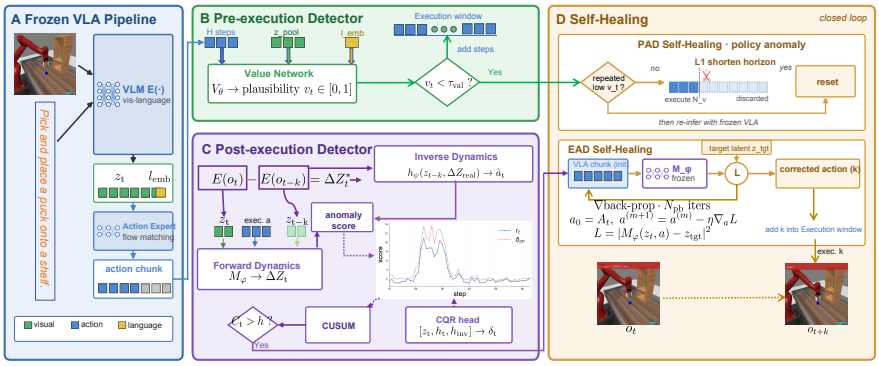


Figure 1: Overview of CASH. A frozen VLA proposes an action chunk, and the PAD evaluates that action chunk before execution. The EAD scores each completed k-step window and accumulates evidence above the conformal upper bound. Trigger events enter one of two self-healing paths according to their source: an EAD trigger induces a latent complement-displacement correction, while a PAD trigger induces foreshortening or a window reset.

task progress failure, and for the latter calls a vision-language model to judge progress when the policy keeps executing actions that cannot complete the task. FIPER (Römer et al., 2025) characterizes impending failures with the out-of-distribution score of the observation embedding and the entropy of the action chunk, and calibrates the two scores separately on successful trajectories. These methods motivate evaluating actions before execution but leave clear gaps. Sentinel's task progress check repeatedly calls a large vision-language model at runtime and offers no calibration guarantee, while its temporal consistency statistic misses smooth but wrong actions. FIPER's chunk entropy applies only to stochastic generative policies and cannot flag confident but wrong actions. Both methods stop at alerting without triggering bounded recovery.

3 METHOD

Figure 1 illustrates the overall workflow of CASH, which consists of a detection module and a self-healing module. The detection module contains two complementary methods, an execution anomaly detector and a policy anomaly detector. The self-healing module selects a different intervention according to the outputs of the two detectors.

3.1 EAD: EXECUTION ANOMALY DETECTION

We write $z_t = E(o_t) \in \mathbb{R}^{L\times D}$ for the latent token sequence produced by the frozen vision encoder and A_t for the action chunk of length C with $a_i \in \mathbb{R}^A$.

Action-modulated forward dynamics. The forward model predicts the latent displacement induced by an action window:

$$\hat{\Delta Z}_{t+k} = M_\phi(z_t, a_{t:t+k}). \tag{1}$$

The latent state is of a much larger dimension than the action window. Directly concatenating action vectors with latent tokens before a normalization layer can easily dilute the action signal. To this end, CASH injects the action using Feature-wise Linear Modulation (FiLM) (Perez et al., 2018):

$$\tilde{z}_t = (1 + \gamma(a_{t:t+k})) \odot z_t + \beta(a_{t:t+k}). \tag{2}$$

The residual form $1+\gamma$ ensures that with initialization $\gamma = 0$ and $\beta = 0$, the modulation degenerates to the identity mapping. A small MLP maps the normalized action window to the per-token vectors γ and β, and a residual network maps the modulated tokens to a displacement with the same shape as z_t.

The forward loss consists of a magnitude term and a direction term:

$$\mathcal{L}_{\text{fwd}} = \left\| \hat{\Delta Z}_{t+k} - \Delta Z_{t+k} \right\|_2^2 + \beta_{\text{dir}} \left(1 - \cos(\hat{\Delta Z}_{t+k}, \Delta Z_{t+k})\right). \tag{3}$$

The regression target is the true latent displacement over the window, $\Delta Z_{t+k} = z_{t+k} - z_t$. The squared term penalizes both magnitude and direction errors, while the cosine term focuses on directional consistency when the magnitudes are close.

Inverse dynamics and counterfactual cycle consistency. With only the forward model, the demonstration data impose too few constraints on the action–state mapping. In successful demonstrations, action windows are strongly correlated with the current state: given a state z_t, the training set usually contains only one corresponding action window $a_{t:t+k}$, and lacks contrastive samples of different actions under the same state. The forward model can then reduce the loss by predicting the state transition from z_t alone, that is, the optimal solution degenerates to $\hat{\Delta Z}_{t+k} \approx g(z_t)$. This prevents the forward model from accurately assessing the consequences of anomalous actions, which in turn weakens anomaly detection based on prediction error.

To address this issue, CASH introduces an inverse dynamics head h_ψ, and enforces the sensitivity of the forward model to the action input through cycle consistency. The inverse head reconstructs the action window from the current state and the visual feature displacement:

$$\hat{a}_{t:t+k} = h_\psi(z_t, \Delta Z_{t+k})\,. \tag{4}$$

The inverse head together with the forward model forms a bidirectional mapping of "action $\leftrightarrow$ state transition".

The inverse head is first trained on true transitions to ensure that it can reconstruct the correct action from a nominal state transition:

$$\mathcal{L}_{\text{inv}} = \|h_\psi(z_t, \Delta Z_{t+k}) - a_{t:t+k}\|_2^2\,. \tag{5}$$

Building on this supervision, cycle consistency closes the mapping in both directions and forces M_ϕ to respond to the action input. For a true window, the displacement predicted by the forward model should pass through the inverse head and return the original action,

$$\mathcal{L}_{\text{cyc}}^{+} = \|h_\psi(z_t, M_\phi(z_t, a_{t:t+k})) - a_{t:t+k}\|_2^2\,. \tag{6}$$

Cycle consistency over true action windows alone is still insufficient to address the following problem: if M_ϕ ignores the action input, then $M_\phi(z_t, a) = M_\phi(z_t, a')$, in which case $h_\psi(z_t, M_\phi(z_t, a))$ outputs the same value for all a, and Equation 6 can still be minimized by making h_ψ output the common action in the training set. To address this, we introduce a counterfactual branch: the true window is replaced by a counterfactual action window $a'_{t:t+k}$, and the forward–inverse loop is required to reconstruct the counterfactual action as well:

$$\mathcal{L}_{\text{cyc}}^{-} = \left\|h_\psi\big(z_t, M_\phi(z_t, a'_{t:t+k})\big) - a'_{t:t+k}\right\|_2^2\,. \tag{7}$$

The counterfactual window $a'_{t:t+k}$ is constructed using the same three types of replacement as in PAD (Section 3.2).

The counterfactual branch provides different action inputs $a \neq a'$ for the same state z_t. If M_ϕ ignores the action input, then $M_\phi(z_t, a) = M_\phi(z_t, a')$, so h_ψ cannot reconstruct both a and a' simultaneously, and the cycle consistency loss cannot decrease. This forces M_ϕ to respond to changes in the action input, fundamentally resolving the action-insensitivity problem.

The execution anomaly score r_t is a weighted combination of the forward prediction error, the directional inconsistency, and the inverse reconstruction error; its definition and scale normalization, the threshold calibration based on conformalized quantile regression (Romano et al., 2019), and the temporal aggregation by a one-sided CUSUM (Page, 1954) are all given in Appendix A.

3.2 PAD: POLICY ANOMALY DETECTION

Suppose the action proposed by the policy is unsuitable for the current state, but the execution process involves no perturbation. The forward model may accurately predict the resulting transition,

and the inverse model may faithfully reconstruct the action, and r_t remains low. The failure lies in the action, not in the execution. To address this, CASH introduces a value model

$$v_t = V_\theta(z_{\mathrm{pool},t}, a_{t:t+k}, l_{\mathrm{emb}}) \in [0,1], \tag{8}$$

where $z_{\mathrm{pool},t}$ pools z_t over the token dimension and l_{emb} is the embedded instruction.

Counterfactual construction. Positive samples are action windows from successful executions. Each positive sample is paired with three types of counterfactual windows: within-episode replacement, same-task cross-episode replacement, and cross-task replacement. Within-episode replacement keeps the task and instruction and changes only the temporal context. Same-task replacement changes the episode and the phase. Cross-task replacement changes both the action distribution and the instruction.

Ranking objective. Let the positive sample score be v^+ and the score of the j-th counterfactual sample be v_j^-, where $j \in \{1,2,3\}$. The loss is

$$\mathcal{L}_{\mathrm{rank}} = \frac{1}{3}\sum_{j=1}^{3} \mathrm{softplus}\big(\gamma_{\mathrm{rank}} - (v^+ - v_j^-)\big) + \beta_{\mathrm{anchor}}\left[-\log v^+\right], \tag{9}$$

where $\mathrm{softplus}(x) = \log(1+e^x)$ and γ_{rank} is the ranking margin. The first term requires the positive sample to obtain a higher score than each counterfactual sample. The second term is a positive-class anchor, which prevents the ranking term from pushing all scores toward zero. The anchor term also gives the threshold a stable interpretation: a low quantile of the positive sample score distribution can serve as the action-level alarm threshold.

Checkpoints are selected by the fraction of positive validation samples that score above all three counterfactual samples rather than by the training loss. The alarm threshold is then a low quantile of the positive validation scores:

$$\tau_{\mathrm{val}} = Q_{\alpha_{\mathrm{val}}}\big(\{v^+\}\big), \tag{10}$$

3.3 Self-healing: how EAD and PAD recover

Detection has practical value only when it is converted into a bounded intervention. CASH defines two self-healing paths that match the two detection signals. The EAD path corrects execution at the state level, and the PAD path adjusts the policy commitment or resets the window.

Complement displacement correction. When EAD triggers, the displacement produced by the executed window differs from what it should. The complement is the signed gap between the displacement that M_ϕ predicts for the executed window and the displacement actually observed, and it gives the correction that returns the latent trajectory to the expected path whether the deviation overshoots or falls short. The derivation and iteration budget are given in Appendix B.

Foreshortened lookahead and window reset. When the value model yields a score below τ_{val}, this usually indicates that the input of the policy is already OOD and that the action window proposed by the policy is not plausible under the current state and instruction. The lowest-cost response is not to modify the action, but to avoid committing to the entire prediction horizon. CASH executes only the first N_v actions, with $N_v < k$, and then obtains a new observation and a new proposal. This shortens the open-loop interval from k to N_v and increases the feedback frequency. This strategy preserves the action distribution of the policy and avoids introducing potentially unreliable corrections in unfamiliar states.

A single low score may come from a transient error or from score noise, but this is rare: it is usually accompanied by low scores recurring within a short period. The gripper state fed to the VLA is closed, but the object is not inside the gripper in the image. For the VLA, a closed gripper means the object has been grasped and it should move toward the goal point. The policy is then caught in a dilemma: either continue moving toward the goal point, or repeatedly adjust the gripper near the object, which keeps triggering PAD. CASH stores the indices of low-value decision points in a sliding window of length W. If the number of events in the window reaches C_{window}, the controller applies the reset operator $\mathrm{Reset}(s_t)$, which is defined in Appendix C.

Table 1: Comparison with VLA-Corrector on the Meta-World and LIBERO datasets. The upper panel reports the success rates on Meta-World tasks of different difficulty and the overall metric, and the lower panel reports those on the four LIBERO task categories and the overall metric; ↑ indicates higher is better.

(a) Meta-World

Method	Easy ↑	Medium ↑	Hard ↑	Very Hard ↑	Avg. ↑
SmolVLA	76.25%	47.27%	48.33%	58.00%	64.70%
VLA-Corrector	76.61%	49.55%	54.17%	59.00%	66.20%
CASH	79.29%	58.64%	56.67%	65.00%	70.60%
π_0	76.43%	48.18%	45.00%	26.00%	61.40%
VLA-Corrector	78.39%	53.18%	45.83%	38.00%	64.90%
CASH	81.07%	56.82%	50.83%	42.00%	68.20%
$\pi_{0.5}$	66.96%	37.27%	46.67%	30.00%	54.30%
VLA-Corrector	72.86%	39.55%	50.83%	36.00%	59.20%
CASH	73.93%	43.18%	55.00%	49.00%	62.40%

(b) LIBERO

Method	Object ↑	Spatial ↑	Goal ↑	Long ↑	Avg. ↑
$\pi_{0.5}$	98.20%	95.60%	98.00%	90.60%	95.60%
VLA-Corrector	98.60%	97.80%	99.20%	94.20%	97.45%
CASH	99.40%	99.20%	99.80%	95.40%	98.45%

Algorithm 1 gives the decision order, trigger budgets, and statistic resets for the foreshortening, window reset, and EAD complement-displacement correction above at deployment time.

4 EXPERIMENTS

We evaluate CASH on two manipulation benchmarks: Meta-World (Yu et al., 2020) and LIBERO (Liu et al., 2023), with SmolVLA (Shukor et al., 2025), π_0 (Black et al., 2025b), and $\pi_{0.5}$ (Black et al., 2025a) as the VLA backbones. The composition of the two benchmarks, the difficulty splits, the action-chunk lengths, the corresponding settings of each backbone, as well as the unified evaluation protocol, the random seeds, and the hyperparameters, are all given in Appendix E.

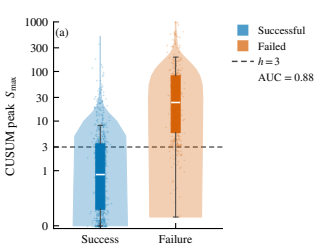
End-to-end success rate. An end-to-end comparison requires two conditions to hold at once: the intervention acts on the same frozen VLA, and the supervision the method needs is directly available in the benchmark. To our knowledge, VLA-Corrector (Pan et al., 2026) is the only system that satisfies both. The remaining methods fall into two groups. Some only detect and never recover. Others do report an end-to-end success rate but break at least one condition: for FLARE (Zhao et al., 2026), the policy under test is no longer the same frozen VLA, and it fundamentally requires failure data; ProbeAct (Zhang et al., 2026a)'s hidden-state probe is supervised by the 3D positions of task-relevant objects and additionally requires intermediate-layer features and gripper-internal signals.

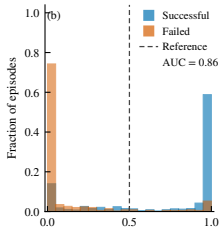
The two panels of Table 1 report the results on Meta-World and LIBERO respectively; both consistently show that CASH yields consistent gains across all difficulty levels: on Meta-World, with SmolVLA, π_0, and $\pi_{0.5}$ as the policy backbone respectively, CASH outperforms the corresponding VLA-Corrector on tasks of every difficulty and on the average success rate; on LIBERO, CASH with the $\pi_{0.5}$ backbone likewise exceeds VLA-Corrector and the original $\pi_{0.5}$ model across all four task categories as well as on the overall average, reaching an average success rate of 98.45%.

Detection method comparison. Table 2 compares CASH with four classes of detection methods (PCA+k-means, RND (Burda et al., 2019), CFM (Lipman et al., 2023), FAIL-Detect (Chen et al., 2025)), with the implementation of each method given in Appendix F. CASH leads on several metrics: it attains the highest AUC and AUPRC (0.981 / 0.986) and, at the operating point calibrated to a 5% per-point false positive rate, its *failure recall reaches* 100%, with no failed episode missed;

Table 2: Comparison with detection methods. Definitions of all column names are given in Appendix F.

Method	AUC	AUPRC	AUC_{pt}	AUC_{H4}	TPR_{pt}	FPR_{pt}	FailRec	SucTrig	Dens	FirstAl	Early50
PCA+ k-means	0.855	0.916	0.758	0.486	0.378	0.074	0.850	0.376	33.97	0.27[0.23]	0.680
RND	0.809	0.827	0.693	0.520	0.094	0.059	0.459	0.203	8.93	0.33[0.37]	0.316
CFM	0.839	0.913	0.621	0.530	0.196	0.069	0.797	0.293	18.00	0.23[0.20]	0.733
FAIL-Detect (logpZO)	0.977	0.979	0.828	0.562	0.587	0.046	0.977	0.143	51.92	0.27[0.17]	0.872
FAIL-Detect (logpO)	0.850	0.888	0.522	0.536	0.062	0.044	0.838	0.286	5.94	0.30[0.40]	0.624
CASH	**0.981**	**0.986**	**0.862**	**0.780**	**0.612**	0.061	**1.000**	0.316	54.33	**0.17**[0.23]	**0.977**





Failed episodes (CASH)

Difficulty	C_t	V_θ	Both
easy	86.2%	99.1%	100.0%
medium	83.3%	90.4%	100.0%
hard	82.0%	98.4%	100.0%
very_hard	84.3%	96.1%	100.0%

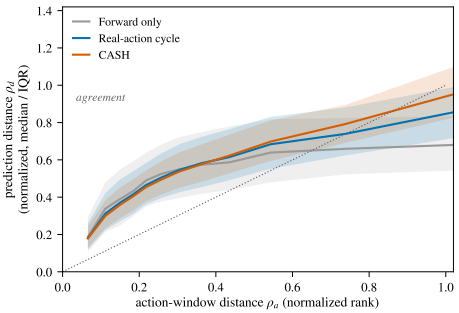
Figure 2: Discrimination of the two detection signals between successful and failed episodes. Left: distribution of the one-sided CUSUM peaks of EAD, with the dashed line indicating the deployment trigger threshold; the vertical axis is symlog and the displayed upper limit is truncated at 10^3 (about 2.3% of failed episodes have peaks beyond this limit). Right: distribution of the 5% quantile of the PAD score v_t within an episode, with the dashed line indicating the reference level 0.5; bar heights are the episode proportions normalized by the sample count of each group. The table on the right gives the proportion of failed episodes triggered by C_t, by v_t, or by *at least one* of the two.

Table 3: Recovery mechanism (Meta-World `medium`). EAD triggers and PAD triggers are the average number of triggers per episode; VLA calls and inference latency are per-episode averages.

Configuration	Success rate ↑	EAD triggers	PAD triggers	VLA calls	Inference latency (s)
SmolVLA	47.3%	–	–	22.4	5.5
EAD	54.1%	1.7	–	22.6	5.9
PAD	56.8%	–	3.6	24.1	6.1
CASH	58.6%	1.7	3.6	24.2	6.4

the strongest baseline logpZO reaches 97.7%, whereas RND recalls only 45.9%, with a per-point TPR of only 0.094—it separates the ranking mainly by relying on a few extreme decision points rather than continuously responding to the failure process. The gap in trigger timing is even more pronounced: the normalized median time-to-first-alarm of CASH is 0.17, earlier than that of all baselines (0.23–0.33), with Early50 reaching 0.977.

Discrimination of the detection mechanisms. Figure 2 shows that both detection signals can effectively distinguish successful from failed task trajectories at the episode level. On the left, the EAD detector accumulates markedly more evidence on failed trajectories: the median of its CUSUM peaks is 23.71, whereas that of successful trajectories is only 0.93, with an average AUC of 0.88. On the right, PAD separates the two families more steeply: the proportion of episodes whose 5% quantile of v_t lies below the reference level 0.5 is 28.9% for successful episodes and 90.1% for failed episodes, with the two distributions barely overlapping, corresponding to an AUC of 0.86. More important is the complementarity of the two in failure recall. In runs that record both signals, the proportion of episodes in which C_t triggers at least once is 84.2% and that for v_t is 95.6%. Taking the union of the two pathways raises the recall on failed episodes to 100%, with no failed episode missed.



Variant	D_{act}	R_{rank}
Forward only	0.379	0.595
Real-action cycle	0.432	0.770
CASH	0.496	0.856

Figure 3: Effect of the inverse head on the action sensitivity of the forward model. Left: when the state z_t is fixed and the action window is varied, the prediction distance ρ_d as a function of the action-window distance ρ_a (the solid line is the median and the shaded region is the interquartile range); the two axes are normalized by their own 95th percentiles, and a curve closer to the gray dashed line (agreement) indicates that the prediction follows the action more closely. Right: summary of the action dispersion D_{act} and the action–output rank correlation R_{rank}. Metric definitions are given in Appendix G.

Recovery mechanism. The recovery mechanism experiments conducted on the Meta-World medium-difficulty tasks (Table 3) show that the full CASH method, which integrates both the PAD and EAD recovery pathways, achieves the best success rate, outperforming the SmolVLA baseline and the configurations in which either pathway is enabled alone, confirming the complementary roles of the two in covering the two failure modes: out-of-distribution inputs to the policy and inconsistent action–state transitions; among them, EAD triggers far less frequently on average than PAD yet brings a considerable standalone performance gain, indicating that the improvement stems mainly from correction quality rather than intervention density. On the cost side, CASH incurs only a small increase in the number of VLA calls and in inference latency, and its computational overhead is within an acceptable range relative to the success-rate gain.

Action sensitivity mechanism. To examine the role of the inverse head, we compare three variants. In Figure 3, the three curves on the left all increase monotonically with ρ_a, but at clearly different rates: when the discrepancy between the action and the true action reaches its maximum level, the prediction distance of the forward-loss-only model rises only to 0.69 of its own 95th percentile and then saturates, whereas CASH rises to 0.99 and is still changing as the action continues to vary. The right panel gives the same conclusion: D_{act} rises from 0.379 to 0.432 and then to 0.496, and R_{rank} from 0.596 to 0.770 and then to 0.856. By incremental attribution, true-action cycle consistency contributes about $2/3$ ($+0.174$) of the total improvement in R_{rank}, and the counterfactual branch about $1/3$ ($+0.086$). This result shows that the inverse head makes the forward model more sensitive to actions and more faithful to the ranking of action differences.

PAD triggering and state-space coverage. Figure 2 shows that v_t can distinguish success from failure at the episode level, but it does not yet show from what kind of state its low scores arise. Figure 4 reveals the source of this discriminative ability. Taking as the boundary the upper bound of the gripper–object distance in the expert demonstration set when the gripper is closed, one can define a region that a demonstration trajectory never enters: it requires the gripper to be *both already closed and sufficiently far from the object*. In the expert demonstrations this region occurs 0 times; once the gripper is closed, its distance to the object is never greater than 0.081 m. This corresponds to a clear physical constraint: the expert closes the gripper only at the object, and the state "gripper closed but not grasping the object" does not exist in the demonstration distribution. Rollouts, by contrast, cross this boundary extensively: 37.6% fall in this region, and among the decision points at which the gripper is closed, 87.8% fall on states not covered by the demonstration set. v_t triggering is strongly correlated with this region: the trigger rate is 48.3% for decision points in the uncovered region and only 12.6% for those in the covered region, with a risk ratio of 3.84 and non-overlapping intervals. The converse also holds: 69.8% of the decision points at which v_t triggers fall in the uncovered region, whereas only 26.3% of the decision points at which it does not trigger do. This shows that

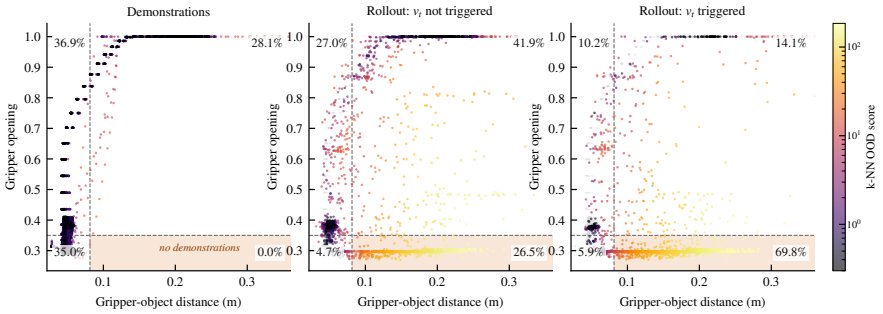


Figure 4: State distribution of self-healing trigger frames relative to the distribution of the expert demonstration set. The horizontal axis is the distance between the gripper end-effector and the task object, and the vertical axis is the gripper opening, which varies within $[0, 1]$, with 0 denoting a closed gripper; the vertical dashed line marks the upper bound of the gripper–object distance in the demonstration set when the gripper is closed (0.081 m), and the horizontal dashed line marks the gripper-opening threshold (0.35). From left to right, the three panels show the expert demonstrations, the decision points at which v_t does not trigger, and the decision points at which v_t triggers; the percentage inside each panel is the proportion of points of that group falling in each of the four regions. The scatter color indicates the 5-nearest-neighbor distance from the state to the demonstration set, a normalized OOD score that is independent of point density.

Table 4: Breakdown of the computational overhead of the modules added by CASH. The first row gives each component's share of the total time per decision (summing to 100%), and the second row gives each component's additional overhead relative to pure VLA inference.

	VLA inference	Visual encoding	EAD scoring	PAD scoring	EAD correction	Added modules
Share of time (%)	86.1	7.7	3.4	0.5	2.3	13.9
Overhead (%)	Baseline	+8.9	+4.0	+0.6	+2.7	**+16.2**

the low v_t scores are not random noise but are concentrated on a set of states that never appear in the demonstration set and are physically self-contradictory, that is, out-of-distribution states for the VLA.

Computational overhead. Whether the above gains are obtained at a significant computational cost directly determines whether the methods can be deployed. We measure the time consumed by each module one by one. Table 4 gives each component's share of the total time per decision and its additional overhead relative to pure VLA inference. VLA action-chunk inference accounts for 86.1% of the time per decision, and the added modules together account for 13.9%. Within the modules, visual encoding (+8.9%) and EAD scoring (+4.0%) are the main sources, together accounting for about 80% of the module overhead; EAD correction costs +2.7% and occurs only 1.7 times per episode on average, and thus its amortized share is very low; PAD scoring, although executed at every decision point, is extremely cheap (+0.6%).

5 Conclusion

CASH treats failure detection and recovery in action-chunking VLA policies as two distinct problems. The execution anomaly detector determines whether an observed latent transition is consistent with the executed action. The value model judges whether a candidate action window is reasonable, covering failures that the consistency check cannot detect. The recovery strategies divide the labor accordingly: complement-displacement correction supplies the latent displacement that the executed actions failed to produce, foreshortening reduces the open-loop interval when the policy is itself uncertain, and window reset restores a familiar state when the policy persistently deviates from the training distribution.

AI USE STATEMENT

AI tools were used to aid and polish the writing of this manuscript. All technical claims, citations, mathematical statements, and the final text were fully reviewed and verified by the authors.

ETHICS STATEMENT

This work does not involve human participants, personal identifiable data, or living subjects. Therefore, no ethics approval is required.

REPRODUCIBILITY STATEMENT

We make efforts to ensure full reproducibility of our experimental results. The detailed algorithm descriptions, model architecture, and theoretical analysis are presented in the main paper. The full hyper-parameter settings, additional ablation studies, and supplementary results are provided in the appendix. The source code for reproducing all experiments is available at the anonymous repository: `https://anonymous.4open.science/r/CASH2-AD11`

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. In *Proceedings of The 8th Conference on Robot Learning*, 2024. URL `https://arxiv.org/abs/2410.04640`.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, David Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL `https://arxiv.org/abs/2506.09985`.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025a. URL `https://proceedings.mlr.press/v305/black25a.html`.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohao Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems XXI*, 2025b. URL `https://doi.org/10.15607/RSS.2025.XXI.010`.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems XIX*, 2023. URL `https://doi.org/10.15607/RSS.2023.XIX.025`.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1810.12894.

Xu Chen, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Andrei Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies. In *Robotics: Science and Systems XXI*, 2025. URL https://doi.org/10.15607/RSS.2025.XXI.073.

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

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 465–472, 2011. URL https://dl.acm.org/doi/10.5555/3104482.3104541.

Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied AI: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244, 2022. URL https://doi.org/10.1109/TETCI.2022.3141105.

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. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.00371.

Isaac Gibbs and Emmanuel J. Candès. Adaptive conformal inference under distribution shift. In *Advances in Neural Information Processing Systems*, volume 34, pp. 1660–1672, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/0d441de75945e5acbc865406fc9a2559-Abstract.html.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. SAFE: Multitask failure detection for vision-language-action models. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.09937.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.01603.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2024. URL https://proceedings.mlr.press/v270/kim25c.html.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. FailSafe: Reasoning and recovery from failures in vision-language-action models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2026. URL https://arxiv.org/abs/2510.01642.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791, 2023. URL https://doi.org/10.52202/075280-1939.

Yushan Liu, Peibo Sun, Xintao Chao, Zhenyang Yang, Yifan Xie, Lingfeng Zhang, Shoujie Li, Chenyu Tang, Fang Chen, Xiao-Ping Zhang, and Wenbo Ding. CheckVLA: Execution-time verification with action-conditioned world model for long-horizon mobile manipulation. *arXiv preprint arXiv:2607.26789*, 2026. URL https://arxiv.org/abs/2607.26789.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied AI. *IEEE Transactions on Neural Networks and Learning Systems*, 37(7):3031–3051, 2026. URL https://doi.org/10.1109/TNNLS.2025.3650584.

E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1–2):100–115, 1954. URL https://doi.org/10.1093/biomet/41.1-2.100.

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

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 3942–3951, 2018. URL https://doi.org/10.1609/aaai.v32i1.11671.

Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. Conformalized quantile regression. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5103c3584b063c431bd1268e9b5e76fb-Abstract.html.

Ralf Römer, Adrian Kobras, Luca Worbis, and Angela P. Schoellig. Failure prediction at runtime for generative robot policies. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2510.09459.

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

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, 2008. URL https://jmlr.org/papers/v9/shafer08a.html.

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

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005. URL https://doi.org/10.1007/b106715.

Isaac R. Ward, Michelle Ho, Houjun Liu, Aaron Feldman, Joseph Vincent, Liam Kruse, Sean Cheong, Duncan Eddy, Mykel J. Kochenderfer, and Mac Schwager. Foundational world models accurately detect bimanual manipulator failures. *arXiv preprint arXiv:2603.06987*, 2026. URL https://arxiv.org/abs/2603.06987.

Jia-Fong Yeh, Kuo-Han Hung, Pang-Chi Lo, Chi-Ming Chung, Tsung-Han Wu, Hung-Ting Su, Yi-Ting Chen, and Winston H. Hsu. AED: Adaptable error detection for few-shot imitation policy. In *Advances in Neural Information Processing Systems*, volume 37, pp. 136805–136836, 2024. URL https://doi.org/10.52202/079017-4347.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of The 3rd Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Fan Zhang, Seongbin Park, Baharan Mirzasoleiman, Shahriar Talebi, and Nader Sehatbakhsh. ProbeAct: Probe-guided training-free failure recovery in vision-language-action models. *arXiv preprint arXiv:2606.09740*, 2026a. URL https://arxiv.org/abs/2606.09740.

Haoran Zhang, Yifu Lu, Boyang Wang, Xuhui Kang, Yen-Ling Kuo, Zezhou Cheng, Mengdi Wang, and Odest Chadwicke Jenkins. Foresight: Failure detection for long-horizon robotic manipulation with action-conditioned world model latents. *arXiv preprint arXiv:2606.23085*, 2026b. URL https://arxiv.org/abs/2606.23085.

Ganlong Zhao, Zijia Tang, Xingping Chen, Zhanghui Kuang, Ye Tian, and Guanbin Li. FLARE: A failure-aware framework for autonomous correction and recovery in visual-language robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2608.26645.

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

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

A CONFORMAL QUANTILE THRESHOLD AND CUSUM TRIGGERING

Execution anomaly score At time t, the detector accesses the latent state at the window start z_{t-k} and the issued actions $a_{t-k:t}$; the forward prediction, the true displacement, and the inverse-head reconstructed action are

$$\hat{\Delta Z}_t = M_\phi(z_{t-k},\, a_{t-k:t}),\ \Delta Z_{\text{real}} = z_t - z_{t-k},\ \hat{a}_t = h_\psi(z_{t-k},\, \Delta Z_{\text{real}}). \tag{11}$$

This yields three scalar components, namely the forward mean squared error, the directional inconsistency, and the inverse reconstruction mean squared error:

$$e_t = \text{mean}_d[(\Delta Z_{\text{real}} - \hat{\Delta Z}_t)^2],\ d_t = 1 - \cos(\hat{\Delta Z}_t, \Delta Z_{\text{real}}),\ c_t = \text{mean}_d[(\hat{a}_t - a_{t-k:t})^2]. \tag{12}$$

where the subscript d denotes averaging over all elements of the vector.

The EAD run-time anomaly score is

$$r_t = \frac{e_t}{s_e} + \lambda_{\text{dir}}\, d_t + \lambda_{\text{inv}} \frac{c_t}{s_c}. \tag{13}$$

The scale factors $s_e = \text{median}(\{e_t\}) + \text{MAD}(\{e_t\}), s_c = \text{median}(\{c_t\}) + \text{MAD}(\{c_t\})$ are estimated on nominal training frames, and are used to place the forward and inverse terms on a comparable scale and to reduce the sensitivity of the score to the overall magnitude of the latent representation.

A single global threshold adapts poorly to manipulation data. Nominal anomaly scores vary with task phase and task: scores in contact-rich phases are higher than in free-space motion, and baseline dynamics differ across tasks. Therefore, CASH learns an upper quantile conditioned on the state and calibrates it with split conformal prediction (Vovk et al., 2005; Shafer & Vovk, 2008; Gibbs & Candès, 2021).

Conditioning variables. The quantile head receives a vector composed of three parts:

$$x_t = [\, z_{\text{pool}} \mid h_{\text{fwd}} \mid h_{\text{inv}} \,]. \tag{14}$$

z_{pool} is the mean pooling of the current latent state along the token dimension, and describes the current observation. h_{fwd} is an intermediate-layer feature of the forward model. h_{inv} is an intermediate-layer feature of the inverse head on the true displacement. The three parts provide information about the current state, the predicted transition, and the observed transition, respectively. The quantile head uses three independent linear projections, followed by a hidden layer and a one-dimensional output:

$$q_\xi(x_t) \approx Q_{1-\alpha}(r_t \mid x_t), \tag{15}$$

where α is the target upper-tail probability and $1 - \alpha$ is close to 1.

Pinball training. The dynamics model produces an anomaly score r_t and a conditioning vector x_t for every frame of the training data. The quantile head minimizes the pinball loss

$$\mathcal{L}_\tau(y, q) = \max(\tau(y - q),\ (\tau - 1)(y - q)), \tag{16}$$

where $y = r_t$ and $q = q_\xi(x_t)$. The minimizer of this loss is the τ quantile of r_t, that is, roughly a fraction τ of nominal scores lie below this output and the remaining $1 - \tau$ lie above it; setting $\tau = 1 - \alpha$ makes the output approach $Q_{1-\alpha}(r_t \mid x_t)$ in the equation above. The quantile head is trained on the same nominal frames as the dynamics model, whereas the conformal correction is computed on a disjoint calibration split of successful trajectories.

Conformal correction: For each calibration sample i, we define the nonconformity score $s_i = r_i - q_\xi(x_i)$. Sorting the scores as $s_{(1)} \le \cdots \le s_{(n)}$, the conformal correction is

$$\hat{Q} = s_{(k)}, \qquad k = \lceil (1 - \alpha)(n + 1) \rceil, \tag{17}$$

The online threshold is $\delta_{\text{on},t} = q_\xi(x_t) + \hat{Q}$, Provided that the calibration and test scores satisfy exchangeability, the marginal probability of a nominal frame exceeding the threshold does not exceed α:

$$\Pr(r_t > \delta_{\text{on},t}) \le \alpha. \tag{18}$$

This is a marginal guarantee over the calibration distribution; it does not guarantee that conditional coverage holds for every x_t, nor does it directly control the false positive rate that depends on sequential decisions. The temporal aggregation layer addresses the second issue.

CUSUM aggregation: Let C_t denote the one-sided cumulative sum

$$C_t = \max(0,\ C_{t-1} + r_t - \delta_{\mathrm{on},t})\,. \tag{19}$$

An isolated spike contributes a bounded value for only one step, and decays once the score falls back below the threshold. If a slow drift produces a small positive excess at each step, C_t keeps growing. The detector triggers when $C_t > h$ where h is the CUSUM threshold.

B Derivation of the latent complement-displacement correction

Suppose EAD triggers at time t. The window starts at $t-k$, and the controller has already executed the actions $a_{t-k:t}$. We define the observed displacement and the displacement that the executed actions should have produced:

$$\Delta z_t = z_t - z_{t-k}, \qquad \Delta z_{\mathrm{right}} = M_\phi(z_{t-k}, a_{t-k:t}). \tag{20}$$

The latent complement displacement is

$$\mathrm{target} = \Delta z_{\mathrm{right}} - \Delta z_t. \tag{21}$$

If the executed actions produced the expected change, the complement displacement is zero. A nonzero complement displacement indicates the change still required to bring the latent trajectory back onto the expected path. The controller then obtains a new action window from the VLA at the current observation. Let a_{vla} denote the first k steps of this proposal, clipped to the environment's action range; this proposal provides an in-distribution initialization. The correction window is obtained by projected gradient descent:

$$\mathcal{L}(a) = \frac{\|M_\phi(z_t, a) - \mathrm{target}\|_2^2}{\|\mathrm{target}\|_2^2}, \tag{22}$$

$$a^{(0)} = \mathrm{clip}(a_{\mathrm{vla}}, a_{\min}, a_{\max}), \tag{23}$$

$$g^{(n)} = \nabla_a \mathcal{L}(a^{(n)}), \tag{24}$$

$$a^{(n+1)} = \mathrm{clip}\left(a^{(n)} - \eta \frac{g^{(n)}}{\left\|g^{(n)}\right\|}, a_{\min}, a_{\max}\right). \tag{25}$$

The normalized gradient makes the step size η approximately equal to the action displacement per iteration, independently of the scale of the M_ϕ Jacobian. The controller computes the loss at every iteration, including the initial proposal, and returns

$$a_{\mathrm{correct}} = a^{(n^*)}, \qquad n^* = \underset{0 \le n \le N_{\mathrm{pb}}}{\arg\min}\ \mathcal{L}(a^{(n)}). \tag{26}$$

Returning the best iteration rather than the final iteration avoids overshooting caused by an overly large step size. The correction window is executed for k steps.

C Window reset operator

When the number of events in the PAD sliding window reaches C_{window}, the controller applies the reset operator $\mathrm{Reset}(s_t)$ to the current episode, so that the policy returns from an unfamiliar state to an initial state within the training distribution before re-proposing actions. The reset consists of two phases executed in a fixed order: first release the gripper, then restore the arm. The order is not interchangeable; if the arm moves while the gripper is still closed and holding an object, it drags the object along and generates friction, while also disturbing the object's original layout. The steps executed during the reset and the actions used for re-inference after the reset are neither written into the detection window nor counted against the episode step budget.

The operator itself is benchmark-agnostic; it only requires the benchmark to provide three quantities: the set of dimension indices of the gripper in the action space and in the state, the arm joint positions and velocities at episode initialization, and the end-effector target pose at episode initialization.

Phase 1: release the gripper. The arm stays still (the remaining action dimensions are set to zero), while each gripper dimension moves continuously toward the opening direction; the opening direction is taken as the lower bound of the action space along each gripper dimension (since the action spaces of all benchmarks are symmetric intervals, this is equivalent to saturating the gripper dimensions to the open end). The gripper aperture is read from the state dimensions specified by the configuration, and for a multi-dimensional gripper we take a representative value. This phase ends when any of the following conditions is met:

1. the gripper aperture reaches 0.95, regarded as fully open;
2. after at least $N_{\text{open}} = 10$ executed steps, the aperture changes by less than 10^{-3} for 3 consecutive steps, regarded as fully extended or saturated;
3. the step cap $N_{\text{open}}^{\max} = 40$ is reached.

Opening first releases the object from the gripper, so that the arm motion of Phase 2 does not carry the object along, and it guarantees that the gripper is in the open state after the reset.

Phase 2: restore the arm. Let q_0 and $\dot{q}_0$ be the arm joint positions and velocities at episode initialization, and p_0 the end-effector target pose at episode initialization, all recorded at the beginning of the episode. The reset writes back directly

$$q_t[\,\mathcal{J}_{\text{arm}}\,] \leftarrow q_0[\,\mathcal{J}_{\text{arm}}\,], \qquad \dot{q}_t[\,\mathcal{J}_{\text{arm}}\,] \leftarrow 0, \qquad p_t \leftarrow p_0, \tag{27}$$

where $\mathcal{J}_{\text{arm}}$ is the set of arm joint dimensions and q and $\dot{q}$ are joint positions and velocities. We write back the joint state directly rather than only the end-effector target pose because the end effector is usually driven from the target pose through constraints (e.g., welding constraints), and writing back only the target leaves the arm merely dragged passively, unable to truly return to the initial pose subject to joint limits and singular configurations. The object's state dimensions and the gripper joint dimensions are not in $\mathcal{J}_{\text{arm}}$ and are therefore not written: the object remains at its position before the reset and must be re-grasped by the policy, while the gripper keeps the open state from Phase 1. After restoration, holding actions are used to refresh the observation, for at most 20 steps, terminating early when the end-effector position is within $0.02\,\text{m}$ of the initial position.

Accompanying state reset. At the same time as the reset, the controller clears the EAD CUSUM statistic, clears the PAD event window $\mathcal{W}_v \leftarrow \emptyset$, and increments the reset counter (cap N_r). It then re-invokes the VLA on the restored observation and attempts grasping again from the initial pose with the gripper open. Once the reset count reaches N_r, no further resets occur.

D FULL INFERENCE PIPELINE

Algorithm 1 CASH deployment-time inference pipeline.

Input: frozen VLA π_θ, encoder E; trained modules $M_\phi, h_\psi, q_\xi, V_\theta$; conformal correction $\hat{Q}$;
thresholds h, τ_{val}; scales s_e, s_c; weights $\lambda_{\text{dir}}, \lambda_{\text{inv}}$;
window length k; foreshortening horizon N_v; PAD sliding-window length W and event count C_{window};
reset budget N_r; correction budget $N_{\text{trig}}^{\max}$; PGD iterations N_{pb}, step η;
action bounds $a_{\min}, a_{\max}$
Init: CUSUM $C \leftarrow 0$; counters $N_{\text{trig}}, N_{\text{reset}} \leftarrow 0$;
PAD event window $\mathcal{W}_v \leftarrow \emptyset$; executed-step buffer $\mathcal{B} \leftarrow \emptyset$

while episode not terminated **do**
  $z_t \leftarrow E(o_t)$, pooled to z_{pool}; $A \leftarrow \pi_\theta(o_t, l)$ `// propose chunk` $a_{t:t+k}$
  $v \leftarrow V_\theta(z_{\text{pool}}, A, l_{\text{emb}})$ `// PAD, before execution`
  push $\mathbf{1}[v < \tau_{\text{val}}]$ into $\mathcal{W}_v$, keeping only the last W decision points
  if $\sum_{w \in \mathcal{W}_v} w \geq C_{\text{window}}$ **and** $N_{\text{reset}} < N_r$ **then**
    open gripper; $s_t \leftarrow \text{Reset}(s_t)$; $N_{\text{reset}} \leftarrow N_{\text{reset}} + 1$
    $\mathcal{W}_v \leftarrow \emptyset$; $\mathcal{B} \leftarrow \emptyset$; $C \leftarrow 0$; **continue** `// re-propose at restored state`
  end if
  $N_{\text{exec}} \leftarrow N_v$ **if** $v < \tau_{\text{val}}$, **else** k `// foreshortening`
  execute $A_{0:N_{\text{exec}}}$; append the executed (z, a, z') steps to $\mathcal{B}$
  while $|\mathcal{B}| \geq k$ **do** `// EAD: score every completed` k`-step window`
    pop the oldest k steps $z_{t-k}, a_{t-k:t}, z_t$ from $\mathcal{B}$
    compute e_t, d_t, c_t by Eq. equation 12, r_t by Eq. equation 13, x_t, and $\delta_{\text{on},t} = q_\xi(x_t) + \hat{Q}$
    $C \leftarrow \max(0,\, C + r_t - \delta_{\text{on},t})$
    if $C > h$ **and** $N_{\text{trig}} < N_{\text{trig}}^{\max}$ **then**
      target $\leftarrow M_\phi(z_{t-k}, a_{t-k:t}) - (z_t - z_{t-k})$ `// latent complement, App. B`
      $a^{(0)} \leftarrow \text{clip}(\pi_\theta(o_t, l)_{0:k}, a_{\min}, a_{\max})$
      for $n = 0, \ldots, N_{\text{pb}} - 1$ **do**
        $g^{(n)} \leftarrow \nabla_a \mathcal{L}(a^{(n)})$;
        $a^{(n+1)} \leftarrow \text{clip}\Big(a^{(n)} - \eta\, g^{(n)} / \Big\|g^{(n)}\Big\|, a_{\min}, a_{\max}\Big)$
      end for
      $a_{\text{correct}} \leftarrow \arg\min_{0 \leq n \leq N_{\text{pb}}} \mathcal{L}(a^{(n)})$; execute a_{correct} for k steps
      $C \leftarrow 0$; $N_{\text{trig}} \leftarrow N_{\text{trig}} + 1$; $\mathcal{B} \leftarrow \emptyset$; **break**
    end if
  end while
end while

E EXPERIMENTAL BENCHMARKS AND UNIFIED SETUP

The experiments of CASH are conducted on two manipulation benchmarks: Meta-World and LIBERO. The two are complementary in action space, control frequency, camera configuration, task structure, and action-chunk length, and they correspond respectively to the short-action-chunk and long-action-chunk deployment regimes. This section describes the composition of these two benchmarks and the experimental setting we adopt, and gives the unified evaluation protocol and hyperparameters shared by all experiments other than the detection-method comparison (whose protocol appears in Appendix F); all tables and figures of the main-text experiments are produced under this unified configuration.

Meta-World. Meta-World is a manipulation benchmark for multi-task continuous control (Yu et al., 2020). We use its MT50 subset, comprising 50 contact-rich manipulation tasks (assembly, insertion, pushing and pulling, opening doors and drawers, placement, etc.). Each task is performed by a Sawyer arm, and the action is 4-dimensional: a three-dimensional end-effector position increment and one-dimensional gripper opening and closing; the observation is a single third-person RGB image plus the proprioceptive state. The control frequency is 80 Hz, and the cap for a single episode is 300 steps. We follow the difficulty split of the benchmark and divide the 50 tasks into four groups, Easy / Medium / Hard / Very Hard, containing 28 / 11 / 6 / 5 tasks respectively, and

we report success rates by group in the main comparison and the ablations. The demonstration data are taken from the public MT50 dataset; SmolVLA is fine-tuned on this data, and π_0 and $\pi_{0.5}$ use public MT50 SFT weights. SmolVLA uses an action chunk with $C = 50$, $H = 10$ (detector window $k = H = 10$), and π_0 and $\pi_{0.5}$ use an action chunk with $C = H = 5$ ($k = 5$).

LIBERO. LIBERO is a manipulation benchmark for lifelong robot learning (Liu et al., 2023), containing four task suites: LIBERO-Object (object manipulation), LIBERO-Spatial (spatial relations), LIBERO-Goal (goal-conditioned), and LIBERO-Long (i.e., LIBERO-10, long-horizon multi-stage tasks), with 10 tasks per suite and 40 tasks in total. The environment uses a Franka Panda arm, and the action is 7-dimensional: a six-DoF end-effector pose increment and one-dimensional gripper opening and closing; the observation comprises two RGB images (a third-person view and a wrist view). The control frequency is 10 Hz, and the episode caps of the four suites are 220 / 280 / 300 / 520 steps, respectively. On LIBERO we use the LIBERO fine-tuned weights of $\pi_{0.5}$, with an action-chunk length of $C = H = 50$ (detector window $k = H = 50$). The demonstration dataset used contains 40 tasks, 1693 episodes, and 273465 frames in total.

Unified setup. On both benchmarks, the VLA backbone remains frozen during both training and deployment of CASH, and the only components that require training are the dynamics model of EAD (including the inverse head and the quantile head) and the value model of PAD. All learnable modules are trained on cached latent features and use only successful demonstrations, requiring neither failure data nor human fault labels. The action dimension, token dimension, token pooling scheme, and window length k are aligned automatically from the cache metadata and the evaluation script; we explicitly distinguish two quantities in this paper: the action-chunk length C is determined by each backbone's own training horizon, whereas the window k used for detection and scoring must be consistent with the number of action steps that the backbone actually executes.

Unified evaluation protocol. All experiments use the same fixed random seed $\texttt{seed} = 1000$, and the evaluation trajectories of every table and figure are produced under the same random sequence and can be compared directly across experiments; the detection-method comparison is specified separately, see Appendix F. Table 1 covers multiple backbones and benchmarks: on Meta-World we use SmolVLA, π_0, and $\pi_{0.5}$ as policy backbones, where SmolVLA is fine-tuned on MT50 and π_0 and $\pi_{0.5}$ use public MT50 SFT weights; on LIBERO the $\pi_{0.5}$ backbone uses open-source weights. The remaining experiments (discriminability of the two detection signals, the recovery mechanism, the action-sensitivity mechanism, PAD triggering and state-space coverage, computational overhead) uniformly use the SmolVLA backbone and the Meta-World benchmark.

Hyperparameters. The hyperparameters of the unified evaluation are as follows, all given in the notation of this paper (C is the action-chunk length, k the detection window, W and C_{window} the PAD sliding-window length and trigger event count, N_v the foreshortening steps, N_r the reset budget, $N_{\text{trig}}^{\max}$ the EAD correction budget, N_{pb} and η the PGD iteration count and step size of the complement-displacement correction, and h the CUSUM threshold):

- EAD: CUSUM threshold $h = 3$, score weights $\lambda_{\text{dir}} = 1.0$, $\lambda_{\text{inv}} = 1.0$, direction-term weight of the forward loss $\beta_{\text{dir}} = 1.0$;
- PAD foreshortening: $N_v = 3$;
- PAD reset trigger: $W = 15$, $C_{\text{window}} = 3$; the gripper opening step interval is $[10, 40]$ (corresponding to N_{open} and $N_{\text{open}}^{\max}$, see Appendix C);
- Trigger budgets: EAD correction cap $N_{\text{trig}}^{\max} = 3$, reset cap $N_r = 1$;
- Complement-displacement correction: the loss takes the normalized $\|\cdot\|_2^2$ form, the initialization is the VLA proposal, normalized PGD is used, $N_{\text{pb}} = 50$, $\eta = 0.05$.

F EXPERIMENTAL PROTOCOL AND IMPLEMENTATION DETAILS OF THE DETECTION COMPARISON

This section gives the evaluation data, calibration protocol, metric definitions, and implementation configuration of each method for Table 2, as well as the deliberate deviations from the official implementations.

Compared methods. The four classes of baselines in Table 2 are: the classical one-class baseline PCA+k-means, Random Network Distillation RND (Burda et al., 2019), the consistency score of conditional flow matching CFM (Lipman et al., 2023), and the learning-based detector for manipulation failure FAIL-Detect (Chen et al., 2025). FAIL-Detect reports its two scoring variants, logpZO and logpO, which share the same conditional UNet, where logpZO is the primary score of that paper. The last row of the table is CASH, referring to the complete method after fusing the two signals; the discriminability analysis of each of the two signals is carried out in the main-text analysis of signal discriminability (the C_t peak convention) and is not expanded in this table.

Decision points and feature contract. Within each episode, decision points $t_0 = 0, k, 2k, \ldots$ are taken only internally until the episode ends; the window of the last decision point may contain fewer than k steps and includes only the remaining steps. The input at every decision point is uniformly the latent observation z_{t_0} (64×960, average-pooled along the token dimension to 960 dimensions) and the action chunk $a_{t_0:t_0+k}$ (flattened to 40 dimensions). Since all methods share this contract, differences among methods come only from their respective score functions and calibration schemes.

Calibration protocol. Thresholds may only be calibrated on successful episodes: half of the successful episodes are chosen at random as the calibration split, and the remaining successful episodes together with all failed episodes form the test set; failed episodes never participate in threshold selection at any stage. Calibration takes the $1 - \alpha$ quantile of the scores of all decision points in the calibration split ($\alpha = 0.05$), so that the per-point false positive rate hits the target value. The calibration split is determined by a fixed seed (`calib-seed`=0, `calib-ratio`=0.5), and all methods share the same split. Thus, the calibration and test sets of every method are identical. The methods differ only in their score functions and network training, while scoring, calibration, and metric computation share the same code path; the differences in Table 2 can therefore come only from the score functions and the calibration schemes, not from a looser trigger threshold.

Column definitions. The column names of Table 2 are used uniformly below, with the following meanings (in the order in which they appear in the table).

- AUC / AUPRC: *episode-level* AUROC and AUPRC. The episode score is the maximum over all decision-point scores in that episode, computed on the test set with episode labels as targets.
- AUC_{pt}: *decision-point-level* pooled AUC, without episode-level max aggregation, computed directly over all test decision points. The gap between this column and AUC is used to check the "max-over-n" bias introduced by episode-level max aggregation.
- AUC_{H4}: *equal-duration truncation* convention, keeping only the first 4 decision points of each episode and then computing as in AUC. This convention is used to remove length bias; the rationale is given in the next paragraph.
- TPR_{pt}: *per-point* true positive rate, i.e., the fraction of all decision points in test failed episodes whose score crosses the threshold.
- FPR_{pt}: *per-point* false positive rate, i.e., the fraction of all decision points in test successful episodes whose score crosses the threshold. This column is the *measured* false positive rate of the calibration protocol on the test set, used to confirm that all methods indeed operate at the same operating point (target 5%).
- FailRec: *episode-level* failure recall, i.e., the fraction of test failed episodes with at least one decision point crossing the threshold.
- SucTrig: *episode-level* success trigger rate, i.e., the fraction of test successful episodes with at least one decision point crossing the threshold, characterizing the cost of false triggers.

- Dens: trigger density, i.e., the average number of threshold crossings per 100 decision points.
- FirstAl: the time of first alarm in failed episodes, defined as the t_0 of the first threshold-crossing decision point divided by the actual number of steps of that episode (normalized to $[0, 1]$), reporting the median and interquartile range over triggered episodes. Values closer to 0 indicate earlier alarms.
- Early50: early detection rate, i.e., the fraction of all test failed episodes whose first trigger time does not exceed 50% of the episode length. Its denominator includes non-triggered episodes, and thus this value never exceeds FailRec. The looser 80% threshold convention overlaps heavily with FailRec (most failures are detected early in the episode, and the two columns are identical digit for digit for some methods), and it is omitted from the table to control the table width.

Length bias and robustness conventions. The main convention of Table 2 takes the maximum over episode-level labels, while the two classes of trajectories differ markedly in length: failed episodes are always 300 steps long, whereas successful episodes are 44–239 steps; long episodes naturally obtain more trigger opportunities, a direct comparison may thus overestimate one side. For this reason the table also reports two conventions. $\mathrm{AUC}_{\mathrm{pt}}$ performs no episode-level aggregation at all and is computed directly over all test decision points, measuring how large the "max-over-n" effect is. $\mathrm{AUC}_{\mathrm{H4}}$ truncates each episode to its first 4 decision points and then computes under the episode-level convention, equalizing the available duration for success and failure; $H = 4$ is chosen because the shortest successful episode is 44 steps (5 decision points), and 4 guarantees that all episodes are covered. The latter is the most conservative reading: under this convention CASH attains 0.780, FAIL-Detect (logpZO) 0.562, and the remaining methods no more than 0.536, showing that the lead of CASH does not vanish once the length bias is removed, and the main-text conclusions follow this convention.

Training corpus. All detectors that require training share the same training corpus: 2500 successful episodes from the expert demonstration set (covering 50 tasks, 50 episodes per task), encoded frame by frame with exactly the same encoder as in the evaluation and segmented by exactly the same decision-point definition as in the evaluation, giving 19329 decision points in total. This corpus contains only successful demonstrations and no failure data whatsoever. The training samples are expert demonstrations while the evaluation samples are actual rollouts of the fine-tuned policy; the two do not have the same state distribution, and the evaluation performance of the detectors still reflects this distribution shift; however, this is not equivalent to task-level out-of-distribution generalization.

Implementation and configuration of each method.

- **PCA+k-means**: PCA whitening is first applied to z_{t_0} of the training corpus to reduce to 192 dimensions (retaining 95.9% of the variance, automatically taking 0.2 times the original dimensionality), then 64 cluster centers are fitted, and the distance to the nearest center is used as the score. Training-free; fits in seconds.
- **RND**: two identically structured conditional UNets (76.7 M parameters each), where the target network parameters are randomly initialized and then frozen, the prediction network approximates the output features of the target network by mean squared error, and the pairwise distance between the two is used as the score. The input is the action chunk and z_{t_0}; Adam, $lr = 10^{-4}$, batch 128, 200 epochs.
- **CFM**: a conditional UNet (62.1 M parameters), whose score is the curvature corresponding to the flow-matching consistency loss, computed at inference with 5 Euler inversion steps. Adam, $lr = 10^{-4}$, batch 128, 200 epochs.
- **FAIL-Detect**: a conditional UNet (62.1 M parameters), using the flow-matching likelihood of the action chunk as the anomaly score, i.e., logpZO in that paper; the observation-conditional likelihood logpO is computed on the same network as a second scoring variant, and the two reuse the same checkpoint. Training configuration as above.

- **CASH**: the forward/inverse dynamics heads, the conformal quantile head, and the value model are trained with the method of this paper, and scoring is performed offline on the evaluation trajectories according to the score definition of this paper.

Reproducibility. The evaluation cache and the decision-point segmentation are deterministic; all learned scorers share the same random seed (`seed`=0), and the threshold-calibration split is fixed by `calib-seed`=0 and `calib-ratio`=0.5. Given the same evaluation cache, therefore, the scores, operating points, and all metrics in the table of every method are reproducible digit for digit, independently of execution order or method submission order. The methods differ only in their own score functions and training pipelines, while the evaluation stage fully shares the same code path.

Deliberate deviations from the official implementations.

1. **Action-chunk step padding (RND)**: the official code hard-codes the action chunk to 16 steps, whereas here $k = 10$. The conditional UNet adopted contains three levels of downsampling and requires the sequence length to be a multiple of 4, otherwise the feature dimensions of the upsampling and skip connections cannot be aligned. We therefore pad the 10 steps with two zero steps to 12 steps, following the same padding rule as in the official data preprocessing, and leave the z_{t_0} side unpadded. The inputs of CFM and FAIL-Detect are the sequence of 240 steps obtained by rearranging z_{t_0} into 4-dimensional groups, which already satisfies the divisibility requirement and needs no padding.
2. **PCA centering**: the inference path of the official code applies the linear transform without subtracting the mean, whereas its own fitting path subtracts the mean and the checkpoint does not store the mean. Thus, the training and inference conventions are inconsistent. Our z_{t_0} exhibits a "large constant offset plus small variation" (the mean norm is about 103, of the same order as the per-sample norm), and not subtracting the mean degenerates the distance to the nearest center to an approximately constant value; we therefore subtract the mean throughout the pipeline and store the mean in the checkpoint.
3. **Score convention of the EAD branch**: the EAD component of CASH in Table 2 uses the per-point conditional excess $r_t - \delta_{\text{on},t}$ rather than the cumulative statistic C_t of the deployment trigger rule; the rationale is given in the next paragraph.

Composition of the CASH score and the convention of the EAD component. The CASH score in Table 2 is obtained by fusing the PAD and EAD branches. The EAD branch is triggered by a CUSUM state machine in closed-loop deployment: $C_t = \max(0, C_{t-1} + r_t - \delta_{\text{on},t})$, and a trigger occurs when C_t exceeds the threshold h. This table, however, follows the same convention as the other per-point scorers and compares the *per-point* conditional excess $r_t - \delta_{\text{on},t}$, for two reasons. First, C_t accumulates within an episode and is reset to zero after a trigger; its value depends on the history of that episode and is not directly comparable with per-point scores; were C_t used in this table, `AUC`, `FirstAl`, and `Early50`/`Early80` would all systematically underestimate EAD. Second, the per-point excess preserves the core mechanism of EAD (the threshold $\delta_{\text{on},t}$ is given adaptively per state by the conformal quantile head) without introducing cross-point accumulation.

G ACTION-SENSITIVITY METRICS

For a fixed state z_t and m action windows $\{a^{(j)}\}$, let $\hat{d}^{(j)} = M_\phi(z_t, a^{(j)})$. The action dispersion is

$$D_{\text{act}} = \frac{1}{LD} \sum_{i=1}^{LD} \text{std}_j(\hat{d}_i^{(j)}), \tag{28}$$

where the predictions are first normalized by their standard deviation over all action windows and all output dimensions, to eliminate the effect of the overall output scale. The action–output rank correlation is the Spearman correlation between the action distances $\rho_a(j,k) = \|a^{(j)} - a^{(k)}\|_2$ and the prediction distances $\rho_d(j,k) = \|\hat{d}^{(j)} - \hat{d}^{(k)}\|_2$:

$$R_{\text{rank}} = \text{Spearman}(\{\rho_a(j,k)\}, \{\rho_d(j,k)\}) \,. \tag{29}$$

Experimental setting: z_t is a state drawn at random from the demonstration data cache, $\{a^{(j)}\}$ are windows drawn at random from the pool of all real action windows, and the three variants share the same set of samples.

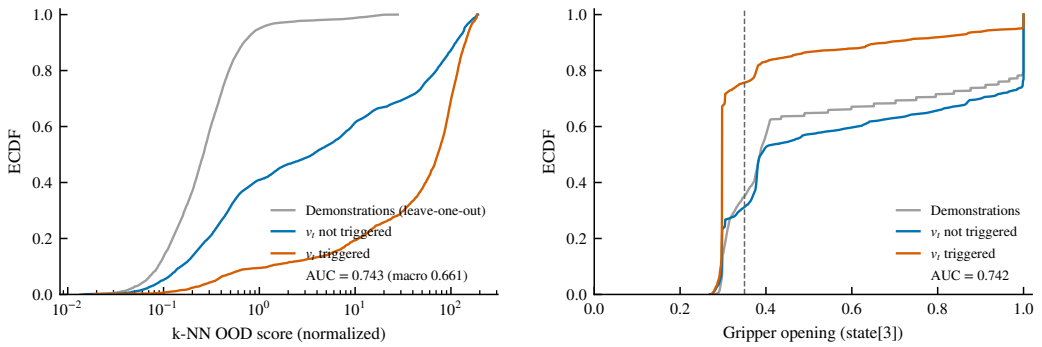


Figure 5: Distribution comparison between triggered and non-triggered decision points of v_t. Left: empirical cumulative distribution of the 5-nearest-neighbor distance to the expert demonstration set (normalized by the 95th percentile of the leave-one-out 5-nearest-neighbor distance of the demonstration set itself), with the demonstration set reference distribution in gray; the median score of triggered frames is 72.9, of non-triggered frames 3.59, and of the demonstration set itself 0.26, with an AUC of 0.743 (0.661 under episode-wise macro averaging). Right: empirical cumulative distribution of the gripper aperture; triggered frames are overall more closed (median 0.298 vs. 0.388), with an AUC of 0.742.

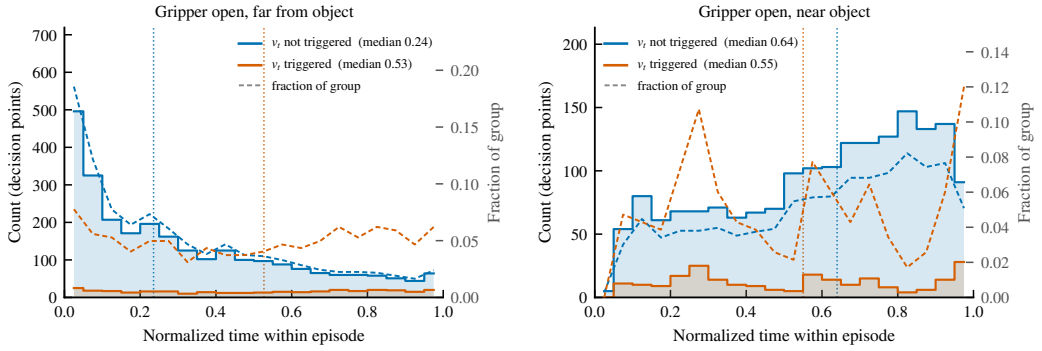


Figure 6: Temporal phase distribution of decision points in the gripper-open state. The horizontal axis is normalized time within the episode; the solid line and fill show the decision-point count (the left axis), the dashed line shows the within-group proportion (the right axis), and the dotted line marks the median phase of each group. Left: gripper open and far from the object; right: gripper open and close to the object.

H Out-of-distribution evidence for policy anomaly detection

Figure 4 gives the state-space coverage evidence. This section supplements the distribution comparison on two scalar features (Figure 5) and the temporal structure of triggering (Figure 6); both are taken from the same deployment run as Figure 4.

The horizontal axis of Figure 5 is the 5-nearest-neighbor distance normalized to the 95% percentile of the demonstration set itself, and the vertical coordinate of the curve at x is the fraction of samples in that group whose score does not exceed x. The triggered curve lies overall to the right of the non-triggered curve: the median score of triggered frames is 72.9, of non-triggered frames 3.59, and of the demonstration set itself 0.26, i.e., the median out-of-distribution score of triggered frames is about 20 times higher than that of non-triggered frames; 90.6% of triggered frames exceed the 95% percentile of the demonstration set itself, versus 59.1% of non-triggered frames. The right panel gives the projection of the same phenomenon onto the gripper aperture: triggered frames are overall more closed (median 0.298 vs. 0.388, AUC 0.742). It should be noted that the AUC measures the separability of the two classes of decision points, whereas the trigger sources of v_t are not unique

(Figure 6 shows that about 14% of triggered frames belong to the late-stage anomaly of "open and far from the object").

Figure 6 explains why most decision points do not trigger. The left panel (open and far from the object) shows a clear bimodal separation: the non-triggered group has a median phase of 0.24, with 52.2% falling in the first quarter of the episode, and a median object displacement of 0.000 m, corresponding to the normal stage in which the arm approaches the object from the initial pose with the gripper wide open, where it is reasonable for v_t not to report a low score; in the same panel the triggered group has a median phase of 0.53, with only 27.6% falling in the first quarter and the rest distributed over the middle and late segments, where the object has already been moved, corresponding to grasping failure or transport anomaly. It can be seen that the larger number of points in this panel of Figure 4 does not weaken the conclusion; it merely mixes in the normal approaching stage. The right panel (open and close to the object) has similar phase distributions for the two groups (median 0.64 vs. 0.55), indicating that whether a trigger occurs in this panel cannot be explained by temporal phase; combined with Figure 4, this panel is not an out-of-distribution region, and its trigger rate (10.2%–14.1%) is below the overall baseline of 26.0%.