DOOVER: DETECT FAILURE FROM DEMOS, DO IT OVER ON DEMAND

Anonymous authors
Paper under double-blind review

ABSTRACT

Current robot policies often continue acting obliviously after execution goes wrong. We introduce **DoOver**, which couples lightweight, policy-native failure detection with on-demand recovery mechanism. With a 0.526M-parameter head that learns jointly with the policy from timestamps in expert demonstrations and reuses policy's visual features, DoOver tracks changes in predicted progress to detect stalls and regressions. It requires no failure data, pretrained foundation models, separate detector training, or runtime action resampling. Across 40 simulated tasks with flow-matching and diffusion backbones and 13 detection baselines per backbone, DoOver leads the eligible baselines in aggregate average precision across task-calibrated detection, alarm-threshold sweeps, and threshold generalization, with $6{,}600\times$ lower failure-detection compute than the sampling-based detectors STAC and ACE. Beyond detection, we establish a systematic benchmark of failure-triggered recovery for frozen policies, comparing seven operators spanning five inference-time intervention families. Among them, failure-triggered feedback world model improves recovery rates over plain replanning by 6.8/5.1 pp and over normal execution by 16.8/16.3 pp for flow-matching/diffusion policies, while avoiding the high cost of guided sampling during normal execution.

1 INTRODUCTION

Imitation learning lets robots acquire manipulation skills directly from expert demonstrations. Recent generative policies based on denoising diffusion probabilistic models (DDPM) (Ho et al., 2020; Chi et al., 2025) and flow matching (FM) (Lipman et al., 2023; Jia et al., 2026; Bai et al., 2026) have made this strikingly effective at turning raw pixels into dexterous, complex behaviors. Yet at deployment such policies remain brittle: a grasp slips, an object shifts, or the arm wanders off the demonstrated manifold, and the policy usually keeps emitting actions while the task has already gone off track. A truly intelligent robot must not only know how to act, but also recognize when its own execution is going wrong and take corrective action, making *runtime failure detection and recovery* a prerequisite for reliable deployment (Kroemer et al., 2021; Agia et al., 2024).

Existing methods for runtime failure detection in generative manipulation policies introduce several practical resource requirements. Supervised detectors learn from successful and failed rollouts (Gu et al., 2026; Park et al., 2026), but representative failures are expensive and often unsafe to collect. Methods that avoid failure rollouts may instead fit density or novelty models (Xu et al., 2025; Römer et al., 2026) or retain reference embeddings (Liu et al., 2024), while others pay at runtime, repeatedly resampling the action distribution (Römer et al., 2026; Agia et al., 2024) or querying an external vision-language model (VLM) (Agia et al., 2024). These additional resource requirements motivate a different design question: **Can the same expert demonstrations that teach a policy to act also teach it to detect failure?**

Our answer is affirmative. We introduce **DoOver** (Fig. 1), which learns progress from policy visual features and uses its temporal evolution to decide when to allocate additional computation to recovery. A lightweight MLP learns from the policy's existing expert demonstrations, using each frame's normalized position within its episode as progress supervision. At runtime, stagnation in predicted progress indicates that execution may have gone off track, without an external foundation model or resampling the action distribution. Jointly trained with the policy, the head reuses its visual features to enable effective *policy-native* failure detection without a separate detector-fitting stage.

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.

Progress learning also requires no simulator state, manual annotations, or failed rollouts. We also demonstrate the detector's plug-in capability: it can be added to an already-trained policy by fitting only the lightweight head.

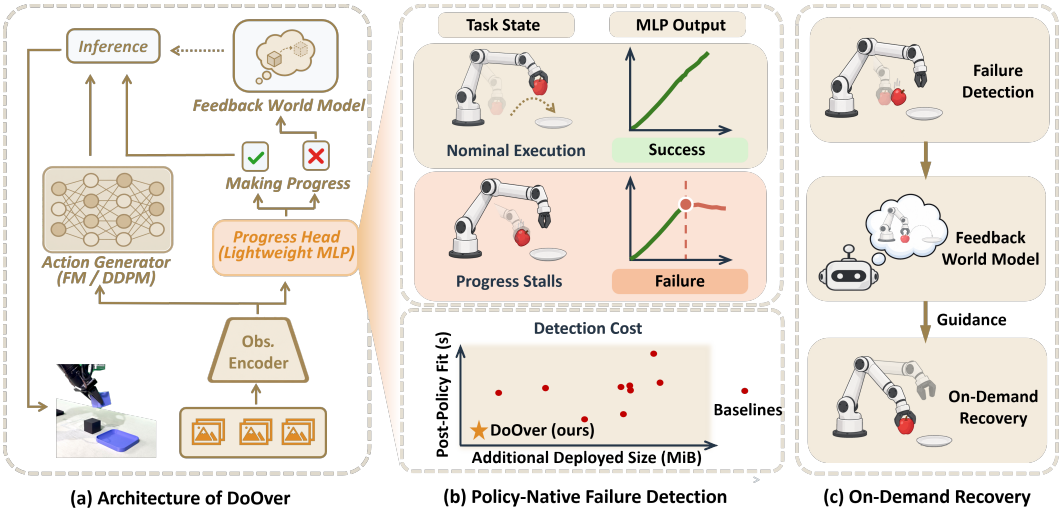

Fig. 1: **DoOver's framework. (a)** Architecture: a lightweight progress head reuses the policy's visual features and learns jointly from normalized expert timestamps. **(b)** Progress stagnation triggers failure alarms; the cost schematic highlights the policy-native detector's small deployment footprint. **(c)** The alarm enables high-budget inference-time steering for on-demand recovery.

Most dedicated runtime failure detectors often stop at the alarm (Gu et al., 2026). Reliable deployment should also enable robots to respond to these alarms by autonomously replanning within the same execution. However, effective recovery requires replanning in situations where the nominal policy struggles, particularly in unfamiliar states that the policy has not learned to handle reliably. Inference-time steering offers additional support in such situations: VLM-based verification (Zhang & Shah, 2026) and Feedback World Model (FWM) guidance (An et al., 2026) can improve policy execution without updating policy parameters. These methods, however, introduce heavy computational cost at inference time. They repeat candidate sampling and verification or world-model-guided inference at every policy decision step.

DoOver addresses this through *failure-triggered, on-demand recovery*: lightweight progress monitoring runs throughout execution, while an alarm activates additional computation for replanning. This separates *when to allocate computation* from *how to replan*, allowing the same detector to support different recovery operators. We evaluate this design through a systematic benchmark of seven recovery operators across five intervention families (sampling diversity, higher NFE, best-of-N, VLM guidance, and world-model guidance) under a shared controller. Our main instantiation, *failure-triggered FWM*, enables feedback-guided, future-aware sampling after an alarm. The same allocation principle could support more capable, even more computationally intensive inference-time steering methods as they become available.

Our failure detection evaluation covers 40 simulated tasks with two mainstream policy backbones FM and DDPM, and 13 detection baselines (Agia et al., 2024; Liu et al., 2024; Xu et al., 2025; Römer et al., 2026; Gu et al., 2026) per backbone, together with two real-robot tasks. DoOver leads the compared baselines in mean aggregate average precision (AP) across task-calibrated detection, alarm-threshold sweeps, and threshold generalization. The central result is strong measured detection from a lightweight head that reuses policy features and requires no repeated action sampling. In our cost audit, its head forward takes 0.077% of a policy decision's time. Joint progress learning preserves aggregate policy success and avoids a separate fitting stage. For recovery, failure-triggered FWM achieves the best aggregate performance among the evaluated operators, with recovery rate gains of 6.8/5.1 pp over the plain recovery operator on FM/DDPM, respectively.

2 Related Work

2.1 Failure Detection

Runtime failure detectors measure how far observations deviate from the training distribution (Liu et al., 2024; Xu et al., 2025), assess consistency across repeated action samples (Agia et al., 2024; Römer et al., 2026), or learn to distinguish successful from failed rollouts (Gu et al., 2026; Park et al., 2026). These approaches can require separately fitted detectors, repeated policy inference, or failed rollouts beyond the policy's expert demonstrations, motivating a monitoring signal that reuses the policy's existing data and computation. Appx. C expands the detail of prior methods and their additional cost requirements in comparison with DoOver.

Task progress provides such a potential signal: a robot that stops advancing toward its goal may indicate failure. The cost of progress monitoring, however, depends on how progress is estimated and what additional models or data are needed. Sentinel (Agia et al., 2024) queries an external VLM to classify progression failures from execution videos and task context. RISE (Yang et al., 2026) adapts a pretrained VLA into a progress value model, while Robometer (Liang et al., 2026) trains an external VLM-based reward model. Earlier VLN work learns progress from goal distances for discrete graph navigation; we use demonstration timestamps to monitor continuous manipulation (Appx. C.4). DoOver makes progress monitoring a lightweight complementary task of policy learning: a 0.526M-parameter head learns jointly from normalized timestamps in the policy's existing demonstrations and reuses visual features already computed for control. Temporal changes in its predictions yield failure alarms, without a separate detector-fitting stage, failed training rollouts, manual progress annotations, external foundation-model inference, or additional action samples.

2.2 Inference-Time Steering

VERITAS (Zhang & Shah, 2026) selects among sampled action candidates using a VLM-generated visual plan. Latent Policy Barrier (LPB) (Sun & Song, 2026) guides diffusion sampling toward predicted states within the expert distribution; Feedback World Model (FWM) (An et al., 2026) develops this approach with observation-based corrections to future-state predictions. Their steering procedures operate throughout execution, adding candidate sampling and verification or world-model gradient computation at each policy decision, which introduces heavy computational cost at inference time. DoOver uses lightweight failure monitoring to activate these expensive inference mechanisms after an alarm, allocating additional computation to on-demand recovery.

3 Method

3.1 Framework Overview

DoOver couples continuous, low-cost failure detection with on-demand additional computation for inference-time recovery (Fig. 1). A jointly trained lightweight progress head reuses the policy's visual features to detect stalled execution. An alarm enables computationally intensive replanning through failure-triggered FWM guidance (An et al., 2026), which we adapt to this recovery setting and to FM. The alarm controls when to increase the action-generation budget; FWM steers sampling using feedback-corrected outcome predictions, with the policy frozen throughout.

3.2 The Native Progress Signal in Expert Demonstrations

Expert demonstrations provide more than action supervision: their temporal ordering also offers a natural proxy for task progress. DoOver uses this supervision to learn a visual progress predictor alongside the action policy, without additional demonstration data or manual progress annotations.

For a demonstration containing $T > 1$ frames, the frame at zero-based timestep $t \in \{0, \dots, T-1\}$ is assigned the scalar progress label $p_t \in [0, 1]$,

$$p_t = \frac{t}{T-1}, \tag{1}$$

ranging from task start (0) to completion (1). These labels require no simulator state, manual annotation, or failure trajectories.

A lightweight MLP maps the policy's visual-history vector $\mathbf{h}_k \in \mathbb{R}^{d_f}$ to a progress curve $\hat{\mathbf{p}}_k \in (0,1)^H$ at policy decision k, where d_f is the feature dimension and H is the number of positions in the policy's planning window.[1] Our evaluated head uses $d_f = 4096$ and $H = 16$. We integrate this head into the policy through joint training:

$$\mathcal{L} = \mathcal{L}_{\text{policy}} + \lambda_p \mathcal{L}_{\text{progress}}, \tag{2}$$

where $\mathcal{L}$ is the total loss, $\mathcal{L}_{\text{policy}}$ is the action-prediction loss, $\mathcal{L}_{\text{progress}}$ is the mean squared error against timestamp-derived progress, and $\lambda_p \geq 0$ weights progress supervision. The progress loss updates both the head and the shared visual encoder in the same optimization step, without a separate detector-training stage. Policy-native monitoring reuses these visual features at deployment, adding only the head and progress-score computation. The prediction for the current observation supplies the progress estimate used for detection.

This policy-native monitor can also be added to an already-trained policy. In this setting, We freeze the trained policy, including its visual encoder, and optimize only a new head with $\mathcal{L}_{\text{progress}}$, retaining the same visual features and detector definitions.

The head uses vision alone to discourage proprioceptive shortcuts to trajectory phase; the policy retains its full observation.

3.3 Runtime Failure Detection

We detect failures from stalled progress using two main detectors, Progress Stagnation (PS) and Multi-Scale Progress Stagnation (MS-PS).

Progress Stagnation. Let $k \geq 0$ index policy decisions from the start of a rollout, and let $\hat{p}_k \in (0,1)$ denote the scalar current-progress entry of $\hat{\mathbf{p}}_k$. PS, one of our two main detectors, measures insufficient advancement over a fixed temporal window of w policy decisions, where w is a positive integer,

$$s_k^{(w)} = -\left(\hat{p}_k - \hat{p}_{\max(k-w,0)}\right), \tag{3}$$

where $s_k^{(w)} \in \mathbb{R}$ is the stagnation score at decision k for window w. The reference estimate $\hat{p}_{\max(k-w,0)}$ comes from the start of the window, or from the first decision if fewer than w decisions have elapsed. Larger scores indicate stronger stagnation or regression. With a fixed window $w = 2$, PS raises an alarm when progress gain falls below 0.001, requiring no deployment-time calibration rollouts.

Multi-Scale Progress Stagnation. MS-PS, our other main detector, standardizes and combines the scalar progress-stagnation scores over a fixed set $\boldsymbol{\mathcal{W}} = \{1, 2, 3, 5\}$ of window lengths measured in policy decisions:

$$S_k = \max_{w \in \boldsymbol{\mathcal{W}}} \frac{s_k^{(w)} - \mu_k^{(w)}}{\sigma_k^{(w)} + \epsilon}, \tag{4}$$

where $S_k \in \mathbb{R}$ is the multi-scale stagnation score at decision k. The normalization statistics $\mu_k^{(w)} \in \mathbb{R}$ and $\sigma_k^{(w)} \geq 0$ are the mean and standard deviation of $s_k^{(w)}$ across successful calibration rollouts of the same task, at the same decision and window length. A small constant $\epsilon > 0$ ensures numerical stability. We set a term to zero if its variance is numerically zero or its decision index falls outside the calibration data.

Normalization makes the scores comparable across window lengths: short windows capture abrupt regressions, while longer windows capture sustained stalls. Benchmark alarms follow the shared protocol in Sec. 4.

We also present Forecast Error (FE) and Multi-Scale Forecast Error (MS-FE), which detect discrepancies between predicted and subsequently estimated progress. Their formulas and interpretation are in Appx. A.1.

[1] We use bold lowercase symbols for vectors, bold uppercase symbols for matrices, and bold calligraphic symbols for sets; scalars are unbolded. Dimensions describe one sample, omitting the batch axis.

3.4 Failure-Triggered On-Demand Recovery

The policy retains its nominal action-generation budget until the alarm. For recovery, we choose fixed-threshold PS to enable deployment without task-specific calibration rollouts (Appx. B.7). Once an alarm fires, the current action chunk is interrupted, the robot retreats toward a previously visited pose, briefly settles to refresh its observation history, and replans. The policy remains frozen.

We compare seven methods spanning five recovery families, with two variants of sampling diversity. **Plain Replanning** regenerates actions with the unmodified sampler. **Stochasticity Injection** (SI) (Singh & Fischer, 2024; Xu et al., 2023) adds sampling noise to test whether greater stochasticity aids recovery. **Source-Noise Scaling** (SNS) (Kingma & Dhariwal, 2018) instead lowers the initial noise scale to favor less extreme samples. **Higher-NFE Replanning** (H-NFE) uses more solver steps to test whether finer sampler discretization alone improves recovery. **Best-of-N Replanning** (BofN) executes the medoid of N sampled action chunks, where N is the candidate count. **VLM-Guided Replanning** (Zhang & Shah, 2026) uses a VLM-generated visual waypoint plan to geometrically select candidate actions. **Failure-triggered FWM** (An et al., 2026) (FWM in figures and tables) enables outcome-guided sampling after an alarm. Its guidance minimizes the squared distance between feedback-corrected predicted outcomes and the nearest expert visual state.

We extend FWM from DDPM to FM, and raise the FM sampler budget from 10 to 100 solver steps only after the alarm, and H-NFE controls for this increased solver budget. On DDPM, the alarm enables guidance within the policy's native solver budget.

Appx. A.2 specifies the recovery mechanisms and backbone-specific sampling updates; Appx. B.7 gives the evaluation protocols, solver budgets, and results.

4 Experiments

We evaluate failure detection and its computational cost, then test whether activating recovery after an alarm improves task completion without updating the policy. We also evaluate DoOver on a real robot and study its design choices through ablations.

4.1 Experimental Setup

Failure detection. Our primary evaluation concerns runtime failure detection for compact generative manipulation policies without external foundation-model inference. Appx. B.2 contrasts successful object transport with empty-gripper motion, providing qualitative evidence that the learned progress signal captures task-relevant visual information. Following prior policy-level monitoring evaluations, we compare against the STAC component of Sentinel and detectors from FAIL-Detect, FIPER, and SAFE (Agia et al., 2024; Xu et al., 2025; Römer et al., 2026; Gu et al., 2026). We evaluate DoOver on both FM and DDPM backbones, including every robustness test. The benchmark compares four DoOver detectors against 13 baselines across four LIBERO suites (Liu et al., 2023), using identical rollouts for all detectors. Four failure-trained baselines are reported but excluded from the default rankings below. On both backbones, PS and MS-PS lead all eligible baselines in mean aggregate BA across task-calibrated detection, alarm-threshold sweeps, and threshold generalization. Fig. 2 presents task-calibrated detection on both backbones; the robustness plots focus on FM, with additional DDPM comparisons in Appx. B.3.

Policy-native DoOver reuses the executing policy's visual features to predict progress within the same policy forward pass. All detectors use the same calibration and test rollouts, with the two sets kept disjoint. We report AP, balanced accuracy (BA), false positive rate (FPR), and horizon-normalized detection time (DT), averaged equally across four suites over three seeds; the scalar $\alpha \in (0, 1)$ denotes the nominal calibration level. Appx. B.1 details task eligibility, protocols, and metrics. Rankings refer to observed means. We report detection performance alongside training, storage, and sampling costs in Appx. B.5 and Tab. 27. All policy-native robustness tests retain the same monitoring architecture and sampling budget.

On-demand recovery. Within each backbone and suite, recovery methods share a frozen policy and PS detector. Action generation retains its nominal sampling budget before the alarm; the alarm activates the selected recovery method. Recovery Rate (RR) measures completion on a fixed set

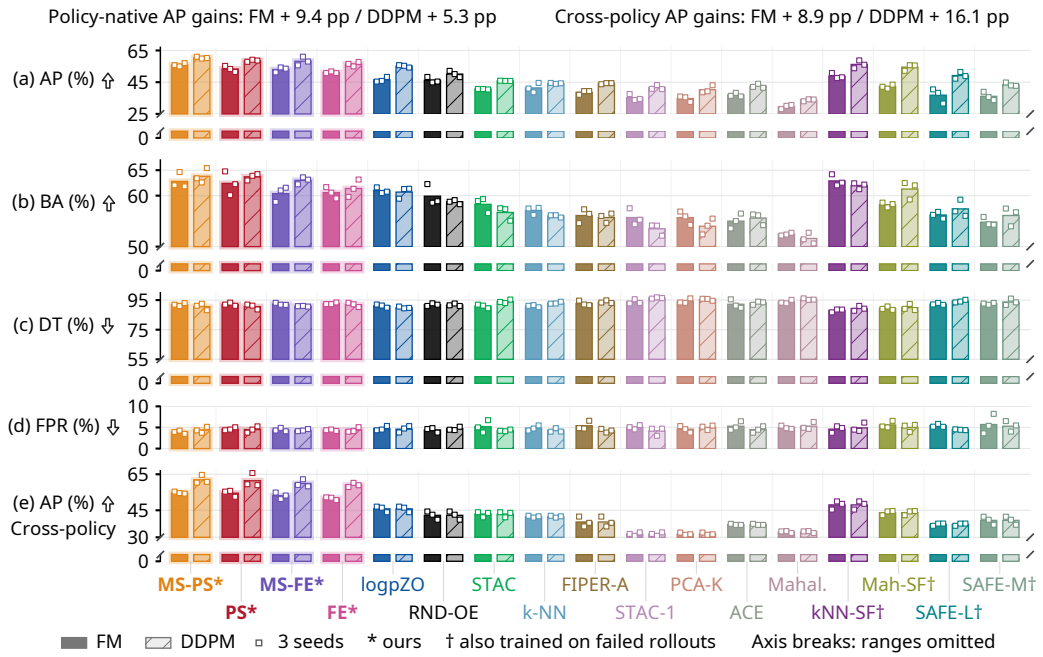

Fig. 2: Task-calibrated failure detection. Policy-native results: **(a)** AP, **(b)** BA, **(c)** DT, **(d)** FPR. Solid/hatched bars denote FM/DDPM policies in (a–d). **(e)** Cross-policy AP: solid/hatched bars denote FM/DDPM DoOver detectors monitoring rollouts from a separate FM policy without a progress head; the shared baseline results are repeated for comparison. Markers show three seeds. AP gains above the figure (pp) compare our best main detector with the strongest eligible baseline for each backbone in (a) and (e). The axes contain breaks, so bar lengths cannot be used to compare ratios. Calibration details and complete cross-policy results are in Appx. B.3 and B.4, respectively. Monitoring costs are reported in Fig. 4(a) and Tab. 27.

of episodes in which the detector raises an alarm before any intervention; Success Rate (SR) measures completion over all episodes. Both metrics average all suites equally. Appx. B.7 specifies aggregation and budgets.

4.2 Failure Detection Results

Across the policy-native evaluations, DoOver achieves competitive detection performance, with very low incremental monitoring cost compared to all baselines.

Monitoring Cost. The policy-native monitor adds a jointly trained 0.526M-parameter head, with no separate training stage or extra action samples. In the FM cost measurements, logpZO/RND-OE add about $118\times/81\times$ as many parameters; logpZO's separate fitting takes 2.45 GPU-hours. A forward pass through the head takes 0.019 ms, while STAC/ACE's additional sampling takes 125.30 ms per decision (about $6{,}600\times$). Fig. 4(a) and Appx. B.5 detail these component measurements and training costs.

Task-Calibrated Failure Detection. We first assess policy-native detection when each target task supplies successful rollouts for calibrating its own functional conformal band (Xu et al.; Römer et al.; Gu et al.) at $\alpha = 0.075$. On held-out rollouts of those tasks, MS-PS and PS rank first and second in AP and BA (Fig. 2(a,b)). On FM, MS-PS exceeds the strongest eligible baseline, by 9.4 percentage points (pp) AP and 1.8 pp BA while avoiding its separate detector fitting and large scoring network (Tab. 27). Both MS-PS and PS achieve lower FPR than STAC at comparable DT (Fig. 2(c,d)). Even including all four failure-trained baselines, MS-PS remains first in FM AP and has the lowest measured FPR. On DDPM, both PS and MS-PS retain the top two AP and BA positions under this full comparison.

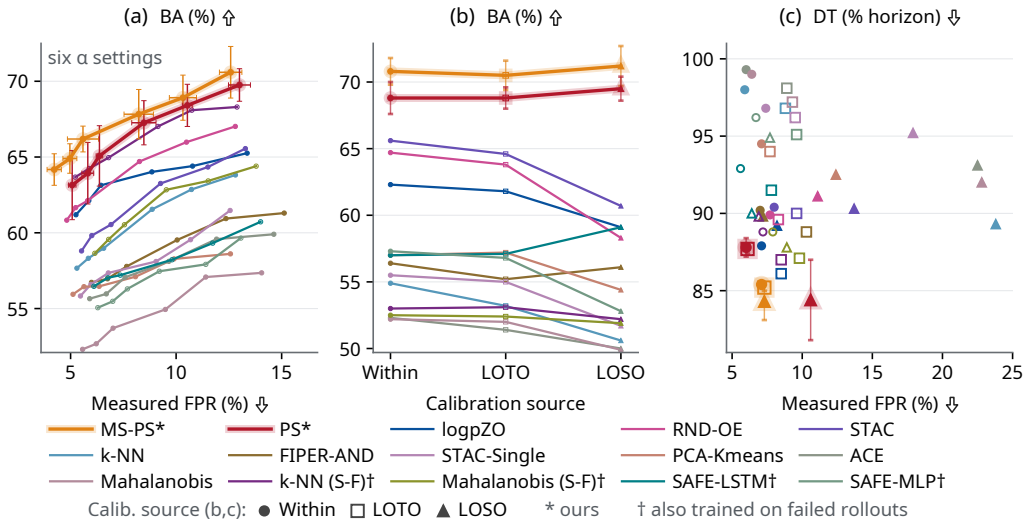

Fig. 3: FM policy-native detection robustness. **(a)** Robustness to alarm threshold choice: BA versus measured FPR over six calibration quantiles. **(b)** BA across different calibration sources. **(c)** DT versus FPR; marker shapes identify calibration sources. Error bars show ± 1 standard deviation (SD) across three seeds. Panels **(b,c)** use raw scalar thresholds; **(a)** and Fig. 2 use functional bands. Detector architectures and sampling budgets are unchanged; monitoring costs are in Fig. 4(a). Aggregate DDPM results are in Appx. B.3.1 and B.3.2. †: also trained on failed rollouts.

Robustness to Alarm Threshold Choice. To test sensitivity to alarm threshold choice, we vary only the threshold using six empirical quantiles without finite-sample correction ($\alpha = 0.01$–0.1), with detector scores and test rollouts fixed. On FM, MS-PS and PS retain the top two BA positions (Fig. 3(a); full FM/DDPM sweeps in Appx. B.3.1). Even including all four failure-trained baselines, PS or MS-PS ranks first in aggregate BA at every tested alarm threshold on both FM and DDPM.

Threshold Generalization Across Tasks and Suites. We test threshold generalization without target-task threshold calibration: leave-one-task-out (LOTO) calibrates only on other tasks in the same suite; leave-one-suite-out (LOSO) uses only other suites. Within-suite calibration includes the target task as a reference. All use raw-score scalar thresholds at $\alpha = 0.075$, distinct from task-calibrated functional bands. Transferred thresholds are assessed by measured FPR, without a per-task coverage guarantee. On FM, both main detectors exceed the eligible baselines in BA under every source (Fig. 3(b)); FM LOSO gains reach 10.5/8.8 pp for MS-PS/PS. MS-PS also has the lowest FM DT across the three sources (Fig. 3(c)). Appx. B.3.2 provides protocols and aggregate FM/DDPM results.

Cross-Policy Failure Detection. Our main use case is policy-native monitoring. As an additional transfer experiment, we test whether a progress signal learned within one policy can also detect failures during another policy's execution. Separately trained FM- and DDPM-based DoOver detectors use their own encoders to observe a standard non-DoOver FM executor without a progress head. With bands calibrated on the executor's successful rollouts, PS or MS-PS ranks first in AP, BA, and ROC-AUC on either detector backbone (Fig. 2(e)). PS reaches 68.2% BA with DDPM detectors, versus 60.1% for the strongest eligible baseline. Appx. B.4 gives complete detection results, including threshold-generalization and alarm-threshold tests.

4.3 On-Demand Recovery Results

Holding each policy fixed, we evaluate all tasks in each suite (Figs. 4(b,c)). Appx. B.7 reports RR and end-to-end SR for every method and for normal execution without replanning.

FWM achieves the highest aggregate RR and SR among the evaluated methods on both backbones. Relative to Plain, RR rises from 56.1% to 62.9% on FM and from 57.3% to 62.3% on DDPM,

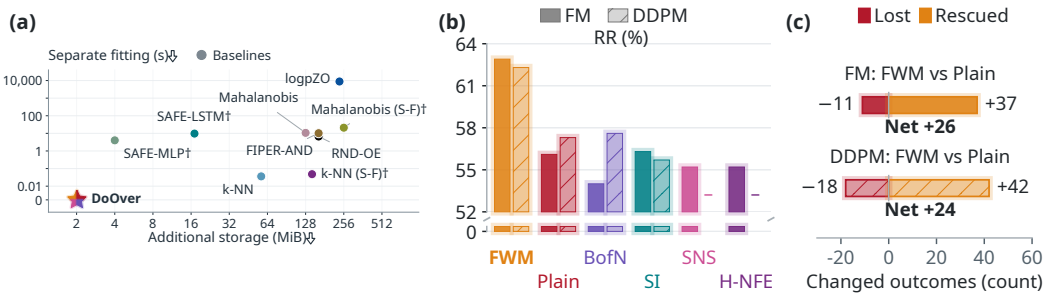

Fig. 4: Detection cost and failure-triggered recovery. **(a)** Policy-native additional storage versus post-policy fitting time for the selected baselines; DoOver's head is jointly trained. †: also trained on failed rollouts. The measured cost components are described in Appx. B.5. **(b)** Suite-macro RR by method; solid/hatched bars denote FM/DDPM. The axis contains a break, so bar lengths cannot be used to compare ratios. Full RR results, including VLM, are in Tab. 16(a); SR and the no-intervention reference are in Tab. 17. **(c)** Changes in episode outcomes when FWM replaces Plain on each backbone: colors distinguish failures changed to successes (Rescued) and successes changed to failures (Lost); labels show the net increase in completed episodes. Solver budgets are detailed in Appx. B.7.

gains of 6.8/5.1 pp. Against normal execution without replanning, FWM improves RR by 16.8/16.3 pp and SR by 4.2/4.9 pp on FM/DDPM (Tab. 17): the no-intervention reference has 46.2%/46.0% RR and 86.5%/82.8% SR. Thus, the gain holds both on alarmed episodes and over the full evaluation set. Differences are computed before rounding. On FM, FWM also exceeds H-NFE at the same increased solver budget (62.9% vs. 55.2% RR; 90.7% vs. 88.8% SR). Appx. B.7 provides the complete comparisons and paired outcome analysis.

Allocating Recovery Computation On Demand. FWM preserves nominal action sampling before the alarm and enables guided sampling for subsequent execution. We compare failure-triggered FWM with always-on FWM on FM. Failure-triggered FWM achieves 90.7% SR versus 89.1% for always-on FWM, while reducing estimated cumulative inference time from 6.17 to 3.22 s per episode (47.8%). Appx. B.8 details this comparison, component measurements, observer updates, and world-model deployment storage.

4.4 Real-Robot Deployments

Failure Detection. An FM-based DoOver policy with its own progress head performs *Color Match*: *Purple Place* and *Black Place* move a cube to its same-color tray from an initial position near the opposite-color tray (Fig. 5(a)). With recovery disabled and task-calibrated corrected bands at $\alpha = 0.10$, PS/MS-PS achieve 84.6%/81.7% BA versus 71.5% for the strongest of six baselines (Fig. 5(b)). Appx. B.9 gives the data split, DT/FPR, and robustness results.

On-Demand Recovery. On ten PS-triggered rollouts per task and method, FWM completes $3/10$ Purple Place and $2/10$ Black Place rollouts ($5/20$, 25% overall). Plain and BofN each complete $0/10$ on both tasks (Fig. 5(b)). Appx. B.9 gives the hardware recovery setting and interpretation.

4.5 Ablations

Detailed protocols and results are in Appx. B.6.

Adding a Monitor to a Frozen Policy. We test whether monitoring can be added without retraining the policy by training only a new head to an already-trained policy. All four detectors retain AP/BA close to joint training in a supplementary LIBERO-Goal comparison (Appx. B.6.1).

Does Proprioception Improve Failure Monitoring? We vary the progress head's access to proprioception while keeping policy inputs unchanged (Appx. B.6.2). Adding proprioception improves

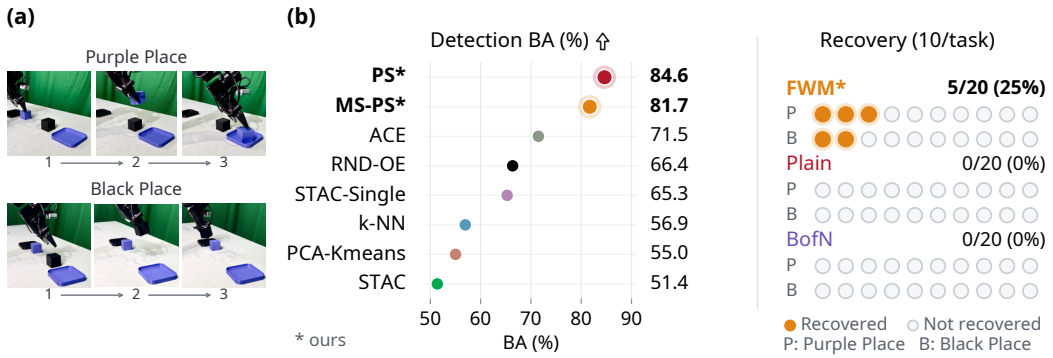

Fig. 5: Real-robot experiments on Color Match. **(a)** Purple Place (top) and Black Place (bottom), each shown in temporal order from left to right. **(b)** Left: detection BA with finite-sample-corrected bands at $\alpha = 0.10$, averaged equally over two tasks. Right: each method has ten detector-triggered rollouts per task; filled circles show successful recoveries (P/B: Purple/Black Place). FWM completes $3/10$ and $2/10$, respectively; Plain and BofN complete $0/10$ on each. Labels beside the methods report the combined counts across both tasks. For detection, PS/MS-PS reuse the policy's visual features without extra action sampling or an external scoring network. $*$: our methods. Protocols and additional detection results appear in Appx. B.9.

Tab. 1: Ablations: (a) Different inputs of PS detector ablation (DT: env. steps); (b) policy SR with recovery disabled.

(a) Head input (FM)

Input	BA (%) ↑	DT (steps) ↓
Vision-only	91.62	**108.8**
+ Proprioception	**93.10**	125.0

(b) Policy SR (%) ↑

Policy	No head	With head	Gain (pp)
FM	83.18	**84.00**	+0.82
DDPM	75.05	**78.20**	+3.15

BA by less than 1.5 pp (from 91.62% to 93.10%), but delays mean detection from 108.8 to 125.0 environment steps (Tab. 1(a)). Appx. B.6.2 discusses the implications for vision-only monitoring.

Does Joint Progress Learning Affect Policy Success? We compare policies trained with and without DoOver's progress head, keeping recovery disabled to isolate the effect of joint progress learning on task SR. DoOver's progress learning improves aggregate SR by 0.82/3.15 pp on FM/DDPM (Tab. 1(b)). Appx. B.6.3 details the comparison and a possible explanation.

5 Conclusion

DoOver couples continuous, lightweight failure detection with on-demand recovery. A small detector determines when to activate more expensive action generation: a jointly trained, policy-native 0.526M-parameter head reuses policy visual features and demonstration timestamps, avoiding failed training rollouts, external foundation models, and repeated action sampling. DoOver leads the eligible baselines in mean aggregate AP across the evaluated FM/DDPM protocols on more than 40 tasks while preserving policy's own success. In the cost comparison experiment, its head forward takes approximately 6,600× less time than STAC and ACE's sampling overhead. Our replanning benchmark identifies failure-triggered FWM as the strongest evaluated method in aggregate RR and SR on both backbones, improving both metrics over Plain recovery and normal execution.

AI USE STATEMENT

In this work, we have not used generative AI tools for any of the tasks with required disclosure under the ICLR 2027 AI Policy for Authors: they were not used to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate mathematical claims or assist with their proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean or reformat datasets, or interpret results, and qualitative and thematic data analysis is not applicable to this work. Our only use of generative AI tools was to edit the manuscript for readability, namely to polish the phrasing of text written by the authors; generative AI tools were not used at any other stage of this research. We have reviewed all AI-assisted work: every AI-suggested edit was manually checked by the authors to ensure that it preserved the intended technical meaning. We take full responsibility for the final content of this work, including all text edited with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken several measures to ensure the reproducibility of our results. Sec. 3 and Appx. A present the formulation of our method, including detailed equations. The code necessary to reproduce our method is included in the anonymized supplementary material. The datasets and benchmarks used in our experiments are described in detail in Section 4, together with citations to their original sources, so that readers can readily access them and reproduce our experimental setup.

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 *Conference on Robot Learning*, 2024.

Tuo An, Jindou Jia, Gen Li, Jingliang Li, Chuhao Zhou, Pengfei Liu, Bofan Lyu, Jiaqi Bai, Xinying Guo, Geng Li, et al. Feedback world model enables precise guidance of diffusion policy. *arXiv preprint arXiv:2605.15705*, 2026.

Jiaqi Bai, Jindou Jia, Yuxuan Hu, Gen Li, Xiangyu Chen, Tuo An, Kuangji Zuo, and Jianfei Yang. Flash: Efficient visuomotor policy via sparse sampling. *arXiv preprint arXiv:2605.15492*, 2026.

Kendrick Boyd, Kevin H Eng, and C David Page. Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 451–466. Springer, 2013.

Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *International Conference on Pattern Recognition*, pp. 3121–3124. IEEE, 2010.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. Racer: Rich language-guided failure recovery policies for imitation learning. In *IEEE International Conference on Robotics and Automation*, pp. 15657–15664. IEEE, 2025.

Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

Ruiqi Feng, Chenglei Yu, Wenhao Deng, Peiyan Hu, and Tailin Wu. On the guidance of flow matching. In *International Conference on Machine Learning*, 2025.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Jindou Jia, Tuo An, Yuxuan Hu, Gen Li, Jingliang Li, Bohan Hou, Xiangyu Chen, Jiaqi Bai, Bofan Lyu, and Jianfei Yang. Mars policy: Multimodality only when it matters. *arXiv preprint arXiv:2605.29766*, 2026.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 22(30): 1–82, 2021.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. In *IEEE International Conference on Robotics and Automation Workshop*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *Proceedings of International Conference on Learning Representations*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939.

Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-task interactive robot fleet learning with visual world models. In *Conference on Robot Learning*, 2024.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *International Conference on Learning Representations*, 2019a.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019b.

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. In *International Conference on Machine Learning*, 2026.

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

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Saurabh Singh and Ian Fischer. Stochastic sampling from deterministic flow models. *arXiv preprint arXiv:2410.02217*, 2024.

Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution, 2026.

Shivam Vats, Devesh K Jha, Maxim Likhachev, Oliver Kroemer, and Diego Romeres. Recoverychaining: Learning local recovery policies for robust manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 9776–9783. IEEE, 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. In *Robotics: Science and Systems*, 2025.

Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart sampling for improving generative processes. In *Advances in Neural Information Processing Systems*, volume 36, pp. 76806–76838, 2023.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, et al. Rise: Self-improving robot policy with compositional world model. In *Robotics: Science and Systems*, 2026.

Mingtong Zhang and Dhruv Shah. Visual verification enables inference-time steering and autonomous policy improvement. *arXiv preprint arXiv:2606.18247*, 2026.

A METHOD AND IMPLEMENTATION DETAILS

The following sections specify the temporal alignment definition of DoOver detectors, two forecast-based detector variants and recovery mechanisms. Appx. B then describes their evaluation protocols and additional results.

A.1 FAILURE DETECTION DEFINITIONS AND CALIBRATION

Detector families. The four detectors use the same predicted progress curve. Progress Stagnation (PS) and Multi-Scale Progress Stagnation (MS-PS), defined in Sec. 3.3, measure changes in current progress estimates. Forecast Error (FE) and Multi-Scale Forecast Error (MS-FE) compare predicted progress with the estimates obtained as execution unfolds. We specify the temporal alignment definition and define the forecast-based variants here.

Temporal alignment. The policy uses $H_o = 8$ observation frames and executes $H_a = 8$ actions per decision. With zero-based indexing, current progress and the next-decision forecast are

$$\hat{p}_k = [\hat{\mathbf{p}}_k]_{H_o-1}, \qquad \hat{p}_{k\to k+1} = [\hat{\mathbf{p}}_k]_{H_o-1+H_a},$$

corresponding to indices 7 and 15 in the $H = 16$ progress curve.

Forecast Error. FE compares the progress predicted for the next policy decision with the progress actually estimated when that decision is reached,

$$e_{k+1} = \hat{p}_{k\to k+1} - \hat{p}_{k+1}, \tag{5}$$

where $\hat{p}_{k\to k+1} \in (0, 1)$ is the scalar forecast made at decision k for progress at decision $k + 1$, $\hat{p}_{k+1} \in (0, 1)$ is the current progress estimate obtained upon reaching that decision, and $e_{k+1} \in \mathbb{R}$ is the signed forecast-error score, with positive values indicating that the forecast exceeds the subsequent estimate. We use a large positive discrepancy as a possible sign of failure: estimated progress has fallen short of the earlier forecast.

Multi-Scale Forecast Error. Following the same multi-scale motivation as MS-PS, we extend FE across temporal windows to obtain MS-FE,

$$E_k = \max_{w\in\boldsymbol{\mathcal{W}}} \frac{1}{\sqrt{w}} \sum_{i=\max(k-w+1,1)}^{k} e_i, \tag{6}$$

where $E_k \in \mathbb{R}$ is the scalar multi-scale forecast-error score at decision k, $\boldsymbol{\mathcal{W}}$ is the same set of window lengths used by MS-PS, w is the length of an individual window, and i indexes the realized policy decisions in that window. Each e_i is the one-decision forecast-error score defined above; the lower summation limit excludes decision 0, which has no preceding forecast. The factor $1/\sqrt{w}$ rescales the accumulated discrepancy by the square root of the window length, and the maximum captures persistent violations of the head's own progress forecast across scales.

All four scores follow the shared benchmark calibration and thresholding protocol in Appx. B.1.3.

A.2 FAILURE-TRIGGERED RECOVERY DETAILS

Recovery methods. The recovery benchmark tests five ways to modify action generation after an alarm. Sampling diversity includes Stochasticity Injection (SI), which adds sampling noise (Singh & Fischer, 2024; Xu et al., 2023), and Source-Noise Scaling (SNS), which reduces initial noise as a low-temperature proxy (Kingma & Dhariwal, 2018); both reuse the existing network evaluations. Higher-NFE Replanning (H-NFE) increases solver resolution. Best-of-N Replanning (BofN) samples N action chunks and executes their medoid, while VLM-Guided Replanning selects candidates using a visual waypoint plan (Zhang & Shah, 2026). FWM instead refines an action chunk using feedback-corrected outcome predictions (An et al., 2026).

All methods share the detector and replan controller. Sampling and guidance updates are adapted to FM and DDPM. Evaluation settings and cost budgets appear in Appx. B.7.

FWM world model and observation feedback. FWM Guidance (An et al., 2026) steers the frozen policy's sampler after an alarm; neither the policy nor the world model is updated during evaluation. The latent observer maintains its feedback state from episode start, before guided sampling is enabled. Offline, a chunk-level latent dynamics model is trained on the policy's training demonstrations. The latent of a frame is the evaluated policy's own frozen image feature followed by layer normalization. Given the current latent, normalized proprioception, the language instruction, and one normalized action chunk, the model predicts the latent reached after executing that chunk. The demonstration latents of each task form its expert memory. At regular intervals of executed environment steps, it advances its belief by the world model's predicted latent change plus a correction proportional to the belief's gap from the previously measured latent, and then records the new measurement. The feedback correction used for guidance is proportional to the gap between the latest measurement and the belief.

FWM guidance energy and sampler update. FWM refines a single action chunk per policy inference. At a guided sampling step, the portion of the sampler's current clean-action estimate that will be executed is passed through the world model from the current observation, and the feedback correction is added to the predicted latent. The energy is the squared Euclidean distance from this corrected prediction to its nearest expert latent of the same task. The neighbor is retrieved without gradient, so the gradient reaches the action estimate only through the predicted latent and steers sampling toward expert-like outcomes. Only the final, low-noise portion of sampling is guided. On DDPM, the energy is evaluated at the scheduler's predicted clean sample, and its gradient with respect to the current noisy sample, scaled by the noise level, is added to the predicted noise. The FM update is specified below.

FM probability path and guidance update. Let $\mathbf{c} \in \mathbb{R}^{d_c}$ denote the fixed, concatenated observation context for one policy inference, including visual features, normalized proprioception, and the language embedding over the observation history. The LIBERO policies use $d_c = 8(512 + 8 + 512) = 8256$. We flatten each action window from $H \times d_a$ to $\mathbf{x} \in \mathbb{R}^D$, where $D = Hd_a$; in LIBERO, $H = 16$, $d_a = 7$, and $D = 112$. FM uses the linear Gaussian probability path

$$\mathbf{x}_\tau = (1-\tau)\boldsymbol{\epsilon} + \tau\mathbf{x}_1, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D), \quad \mathbf{x}_1 \sim p_{\text{data}}(\cdot \mid \mathbf{c}), \quad \boldsymbol{\epsilon} \perp \mathbf{x}_1, \tag{7}$$

where $\tau \in [0,1]$ is sampling time, $\boldsymbol{\epsilon} \in \mathbb{R}^D$ is independent Gaussian noise, $\mathbf{0} \in \mathbb{R}^D$, $\mathbf{I}_D \in \mathbb{R}^{D\times D}$ is the identity matrix, and $p_{\text{data}}(\cdot \mid \mathbf{c})$ is the conditional distribution of normalized demonstration action chunks. Let p_τ be the conditional density along this path and $\mathbf{s}_\tau(\mathbf{x} \mid \mathbf{c}) = \nabla_{\mathbf{x}} \log p_\tau(\mathbf{x} \mid \mathbf{c}) \in \mathbb{R}^D$ its score. For $0 < \tau < 1$, the ideal velocity $\mathbf{v}(\mathbf{x}, \tau \mid \mathbf{c}) \in \mathbb{R}^D$ satisfies

$$\mathbf{s}_\tau(\mathbf{x} \mid \mathbf{c}) = \frac{\tau\mathbf{v}(\mathbf{x},\tau \mid \mathbf{c}) - \mathbf{x}}{1-\tau}, \qquad \mathbf{v}(\mathbf{x},\tau \mid \mathbf{c}) = \frac{\mathbf{x} + (1-\tau)\mathbf{s}_\tau(\mathbf{x} \mid \mathbf{c})}{\tau}. \tag{8}$$

Replacing the score by $\mathbf{s}_\tau - \lambda\nabla_{\mathbf{x}} E_\tau(\mathbf{x})$ therefore gives the coefficient $(1-\tau)/\tau$ in Eq. **??**. This is the standard Gaussian-path conversion (Feng et al., 2025); we evaluate the energy through a clean-action estimate to apply FWM guidance to FM.

With context suppressed in the learned velocity $\mathbf{v}_{\boldsymbol{\theta}} : \mathbb{R}^D \times [0,1] \to \mathbb{R}^D$, the implemented scalar energy is

$$\hat{\mathbf{x}}_1(\mathbf{x},\tau) = \mathbf{x} + (1-\tau)\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x},\tau), \tag{9}$$

$$E_\tau(\mathbf{x}) = \left\| \mathbf{f}_{\boldsymbol{\psi}}\Big(\mathbf{c}, \mathbf{P}\ \text{clip}_{[-1,1]}[\hat{\mathbf{x}}_1(\mathbf{x},\tau)]\Big) + \delta\mathbf{z} - \mathbf{z}^\star \right\|_2^2. \tag{10}$$

Here $E_\tau(\mathbf{x}) = E(\hat{\mathbf{x}}_1(\mathbf{x},\tau))$ relates this expression to Eq. **??**. The world model $\mathbf{f}_{\boldsymbol{\psi}} : \mathbb{R}^{d_c} \times \mathbb{R}^{H_a d_a} \to \mathbb{R}^{d_z}$ reads the current-frame components of $\mathbf{c}$ and predicts the visual latent after the executed chunk, with $d_z = 512$. The vectors $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ and $\boldsymbol{\psi} \in \mathbb{R}^{d_\psi}$ collect the velocity-model and world-model parameters; d_θ and d_ψ are their parameter counts. The binary selection matrix $\mathbf{P} \in \{0,1\}^{H_a d_a \times D}$ extracts planning-window positions $H_o - 1$ through $H_o + H_a - 2$; here it has shape 56×112 and selects positions 7–14. The observer correction $\delta\mathbf{z} \in \mathbb{R}^{d_z}$ is held fixed throughout that sampling call. For task q, the expert memory is a finite set $\boldsymbol{\mathcal{Z}}_q \subset \mathbb{R}^{d_z}$ containing $M_q = |\boldsymbol{\mathcal{Z}}_q|$ latent vectors. The target $\mathbf{z}^\star \in \boldsymbol{\mathcal{Z}}_q$ is nearest to the corrected prediction; it is retrieved anew at each guided step and held constant during differentiation. The evaluated configuration uses the unweighted visual-latent distance and clips clean-action estimates elementwise to the normalized action range $[-1,1]$.

Let N_{FM} be the number of integration steps and $i \in \{0, \ldots, N_{\mathrm{FM}} - 1\}$ their index. For step size $h = 1/N_{\mathrm{FM}}$ and step time $\tau_i = i/N_{\mathrm{FM}}$, define $\tau_m = \tau_i + h/2$ and $\mathbf{x}_m = \mathbf{x}^{(i)} + (h/2)\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \tau_i)$, where $\mathbf{x}^{(i)}, \mathbf{x}_m \in \mathbb{R}^D$ are the step-start and midpoint action states, and subscript m denotes the midpoint of step i. When guidance is active and $\tau_m \geq 0.85$, the implemented update is

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + h \left[\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_m, \tau_m) - \lambda \frac{1 - \tau_m}{\tau_m} \nabla_{\mathbf{x}_m} E_{\tau_m}(\mathbf{x}_m) \right]. \tag{11}$$

Otherwise the guidance term is zero. We use $\lambda = 2$ with no gradient norm clipping. Gradients pass through the velocity prediction, clean-action clipping, and world model to $\mathbf{x}_m$; policy and world-model parameters remain frozen. Guidance modifies the midpoint update, while its start-point velocity is the original policy prediction. Backward passes are additional to the forward-evaluation count (Appx. B.7).

Why increase the FM solver budget? At $N_{\mathrm{FM}} = 10$, only two solver midpoints lie in the guidance window $\tau \geq 0.85$; at $N_{\mathrm{FM}} = 100$, there are 15, with step size reduced from 0.1 to 0.01. The finer grid allows repeated outcome prediction, expert-target retrieval, and gradient correction as the candidate action changes within this low-noise window. It also matches the number of guided updates in our native 100-step DDPM configuration. The budget is chosen to resolve the guidance window, rather than to increase the nominal guidance strength, whose contribution is scaled by h at every update. H-NFE evaluates the same 100-step FM budget without guidance, separating the observed benefit of outcome guidance from solver refinement alone (Appx. B.7).

FWM activation. Once switched on, guidance applies at replanning. Before each inference, an optional cost gate compares the distance from the current observation latent to its nearest expert latent of the task with a threshold, and skips guidance for that inference when the observation is already close to the expert data. FWM is not combined with SI, SNS, BofN, or VLM.

B Experimental Details and Additional Results

B.1 Experimental Setup

We first specify the simulation benchmarks, baseline detectors, and common evaluation protocol. Policy-native detection uses progress predicted by the executing policy itself. Cross-policy monitoring and real-robot experiments follow the same evaluation structure, with their data and calibration settings given in Appx. B.4 and B.9, respectively.

B.1.1 Benchmarks, Policies, and Rollouts

We evaluate DoOver on two popular generative policy families: flow-matching policy and diffusion policy, denoted FM and DDPM throughout the experiments.

The FM failure-detection benchmark evaluates 38 eligible tasks on four LIBERO suites (eligibility rules in Appx. B.1.3). All detectors are evaluated on identical rollouts from the same FM policy within each benchmark setting, so the comparison changes the detector rather than the executed actions or episode outcomes. Each rollout round contains 50 episodes on each of the ten tasks of a suite. We pool five rollout rounds for detector fitting and calibration, providing rollout-trained baselines with the same size of fitting sets to the expert demonstrations. Evaluation uses held-out rollouts. Only successful calibration rollouts are used by our detectors and the nine baselines without failure data; the four failure-trained baselines additionally use failed calibration rollouts. Real-robot tasks and their separate calibration protocol are described in Appx. B.9.

B.1.2 Baselines

All baselines score the same policy rollouts and use the same calibration data and thresholding protocol as our detectors (Appx. B.1.3). Below we summarize how each method scores a rollout, following the original papers.

Action consistency. STAC (Agia et al., 2024) detects erratic behavior through temporal inconsistency: it samples a batch of action chunks at each policy inference, computes a statistical distance (maximum mean discrepancy with an RBF kernel) between the action distributions of consecutive inferences over their temporally overlapping steps, and accumulates this distance over the episode. STAC-Single, the real-time variant used by SAFE (Gu et al., 2026), draws only one chunk per inference; it therefore accumulates the distance between the overlapping steps of consecutive single samples instead of a distance between action distributions.

Deviations from successful observations. These baselines flag observations whose policy embeddings differ from the embeddings seen in successful executions. logpZO (Xu et al., 2025) fits a separate flow-matching model that maps the embeddings of the policy's training demonstrations to Gaussian noise; for a new embedding, it follows the learned flow to a latent noise estimate and uses the squared norm of this estimate as the score, which is large when the estimate is unlikely under the Gaussian. RND-OE (Römer et al., 2026) applies random network distillation in the policy's observation-embedding space: a predictor network is trained on successful rollouts to match the outputs of a frozen, randomly initialized target network, and the discrepancy between the two outputs grows for novel embeddings. k-NN and Mahalanobis (Agia et al., 2024; Gu et al., 2026) use distances in the embedding space directly: the mean Euclidean distance to the k nearest reference embeddings, or the Mahalanobis distance to the reference distribution, with references taken from successful rollouts; as in Sentinel, the per-inference distance is accumulated over the episode. PCA-Kmeans (Liu et al., 2024), in the success-only form used by FIPER (Römer et al., 2026), projects the embeddings onto their leading principal components, clusters the projected reference embeddings with k-means, and scores a new embedding by the distance of its projection to the nearest cluster center.

Action uncertainty and combined detectors. ACE (Römer et al., 2026) measures the uncertainty of the generated actions: for a batch of sampled action chunks, it estimates with histograms the entropy of the samples in the Cartesian end-effector space at each predicted step, averages these entropies over the chunk, and aggregates the resulting chunk scores over a sliding window of policy inferences, so that the score is high when the samples are spread out rather than concentrated in a few distinct behavior modes. FIPER-AND, the full FIPER predictor (Römer et al., 2026), combines the two predictors with a logical AND: RND-OE and ACE are each aggregated over a sliding window and compared with their own time-varying thresholds calibrated on successful rollouts, and a failure is flagged only when both windowed scores exceed their thresholds.

Failure-trained baselines. Following SAFE (Gu et al., 2026), the S-F variants of k-NN and Mahalanobis also keep embeddings from failed rollouts and score a new embedding by its distance to the successful references minus its distance to the failed ones; this difference, accumulated in the same way, is high for embeddings far from successes and close to failures. SAFE-MLP and SAFE-LSTM (Gu et al., 2026) learn failure predictors on the policy embeddings from both successful and failed rollouts. SAFE-MLP maps the embedding at each inference to a scalar score with a small multilayer perceptron and accumulates these scores over time, whereas SAFE-LSTM processes the embedding sequence with an LSTM and outputs a failure score at each inference; both are trained to raise the score on failed rollouts and lower it on successful ones.

B.1.3 Protocol, Aggregation, and Metrics

We use disjoint data subsets for detector fitting, temporal normalization, and threshold calibration; test rollouts are held out.

A common observation horizon. LIBERO ends an episode when the task succeeds, while unsuccessful execution continues to the task timeout. Complete traces can therefore reveal the outcome through their duration: a longer trace offers more opportunities to cross an alarm threshold, and accumulated scores can grow simply because execution continues. We remove this duration cue by evaluating every detector over the same number of policy decisions within each task. The comparison then concerns the monitoring signal over a common observation period.

For task q, let i index a rollout and let $L_i \in \mathbb{Z}_{>0}$ be its untruncated number of recorded policy decisions. Let $\mathcal{D}_{\mathrm{FIT},q}$ and $\mathcal{D}_{\mathrm{NORM},q}$ be the disjoint finite sets of successful-rollout indices assigned

to detector fitting and temporal normalization. Their union $\boldsymbol{\mathcal{E}}_q = \boldsymbol{\mathcal{D}}_{\mathrm{FIT},q} \cup \boldsymbol{\mathcal{D}}_{\mathrm{NORM},q}$ has cardinality $|\boldsymbol{\mathcal{E}}_q| = |\boldsymbol{\mathcal{D}}_{\mathrm{FIT},q}| + |\boldsymbol{\mathcal{D}}_{\mathrm{NORM},q}|$. Before threshold calibration or test scoring, we fix

$$K_q = \min_{i \in \boldsymbol{\mathcal{E}}_q} L_i, \qquad T_q = 8K_q. \tag{12}$$

Here $K_q \in \mathbb{Z}_{>0}$ is the task's horizon in policy decisions, and $T_q = 8K_q$ is its environment-step cutoff, since a policy decision occurs every eight environment steps. We use the observed minimum with no additional margin and require at least six decisions. Threshold-calibration and test rollouts do not determine this horizon. This choice keeps an initial observation window available in every successful reference rollout, including the fastest successful executions.

For normalization, threshold calibration, and testing, trajectories with at least K_q recorded decisions contribute their first K_q scores, indexed $0, \ldots, K_q - 1$. Shorter trajectories are excluded under the same rule for all methods; we do not pad them or give them a different observation period. The original rollouts are still collected to success or timeout: truncation specifies the observations available to the detection benchmark. The horizon is determined before any detector's test performance is known, and every detector receives the same retained episodes and prefixes. In the FM benchmark, this excludes 0–8 successful test rollouts per suite and seed, totaling 24 across the three seeds. This prevents method-specific observation lengths and gives each detector the same opportunity to raise an alarm.

Metrics and aggregation. With failure as the positive class, true positive rate (TPR) (Fawcett, 2006) is the fraction of failed episodes detected, true negative rate (TNR) is the fraction of successful episodes without alarms, and false positive rate $\mathrm{FPR} = 1 - \mathrm{TNR}$. balanced accuracy (BA) is $(\mathrm{TPR} + \mathrm{TNR})/2$ (Brodersen et al., 2010). average precision (AP) evaluates episode ranking using the maximum time-standardized score (Boyd et al., 2013). The cross-policy tables additionally report ROC-AUC (Fawcett; Appx. B.4). Detection time (DT) (Agia et al.; Gu et al.) averages the first-alarm inference index divided by the corresponding task horizon K_q over failed episodes, assigning 1 to missed detections. We report measured FPR alongside BA and DT. For each seed, metrics are computed per eligible task, averaged over tasks within each suite, and then averaged equally over the four suites. We report the mean and sample standard deviation over three seeds. The figures express BA, AP, FPR, and horizon-normalized DT as percentages; differences in BA and AP are reported in pp. Two eligibility rules depend only on the executing policy's outcomes and apply identically to all failure detection methods. We select the tasks before detector evaluation, excluding tasks with fewer than eight successful rollouts in the initial calibration round (FM LIBERO-Spatial task 5). We compute all reported metrics on the same tasks, requiring both successful and failed test rollouts. This excludes FM LIBERO-Object task 8 and DDPM LIBERO-10 task 5.

B.2 Qualitative Analysis of the Learned Progress Signal

Normalized demonstration timestamps provide progress supervision without manual progress labels. To examine what the learned signal tracks during execution, we compare two rollouts of the same FM policy on the LIBERO-Spatial task: *picking up the black bowl in the top drawer*. Figs. 7 and 6 align video frames with their recorded predictions, using identical frame times, display windows, and axis scales.

Progress during successful object transport. In Fig. 6, the bowl is grasped, lifted, and carried with the gripper successfully. Over steps 72–96, predicted progress increases from 0.506 to 0.654. The rollout subsequently succeeds, with no recorded PS alarm.

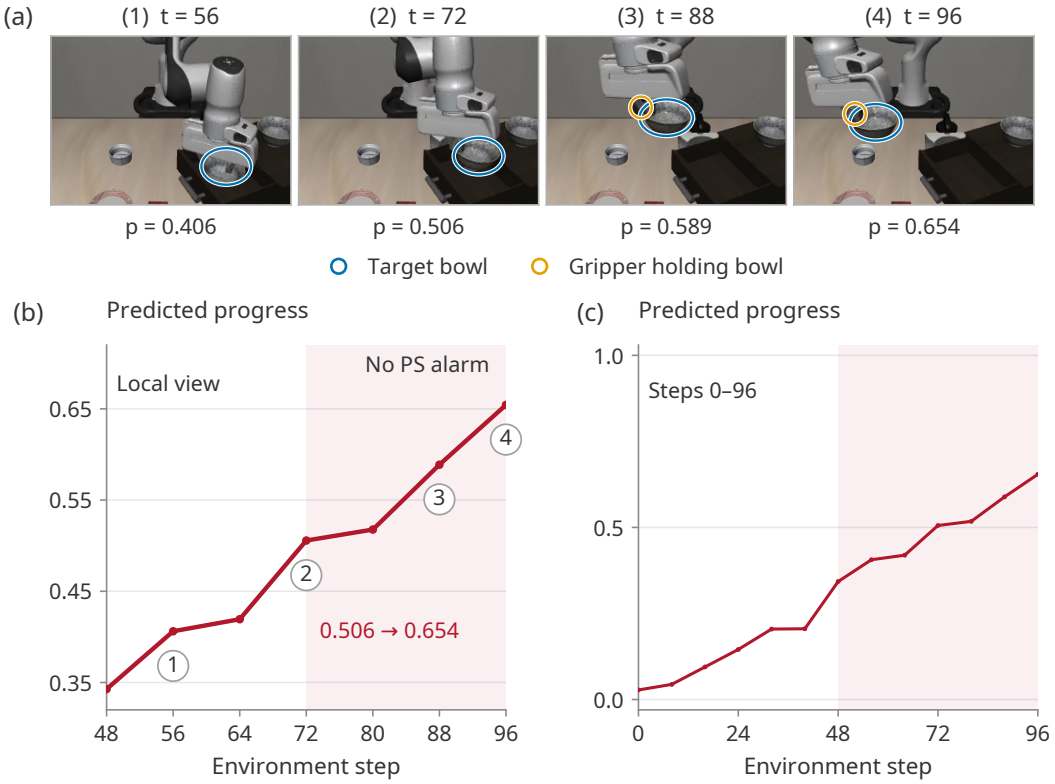

Fig. 6: Predicted progress and video frames during a successful grasp. (a) Four frames from a successful rollout, with their corresponding predicted progress values. Blue outlines mark the target bowl; orange outlines highlight the closed gripper which holding the bowl. (b) A local view of the recorded predictions. Four numbered markers correspond to the frames in (a). As the bowl moves with the gripper, predicted progress increases from 0.506 at step 72 to 0.654 at step 96. (c) Predictions from step 0 to step 96, with the interval shown in (b) shaded.

Same continued motion after a missed grasp. In Fig. 7, the gripper withdraws from the drawer and continues moving while the bowl remains behind. Between the same steps 72 and 96, predicted progress decreases from 0.471 to 0.416, despite the continued arm motion. The recorded PS alarm occurs at step 96. During this interval, the robot keeps moving without transporting the object. The two traces show different progress estimates despite continued arm motion in both rollouts: progress falls when the bowl is left behind and rises when it moves with the gripper.

Interpretation. In these examples, the learned signal distinguishes object transport from empty-gripper motion. It therefore appears to reflect the observed interaction, rather than simply how far the arm has moved through a familiar trajectory.

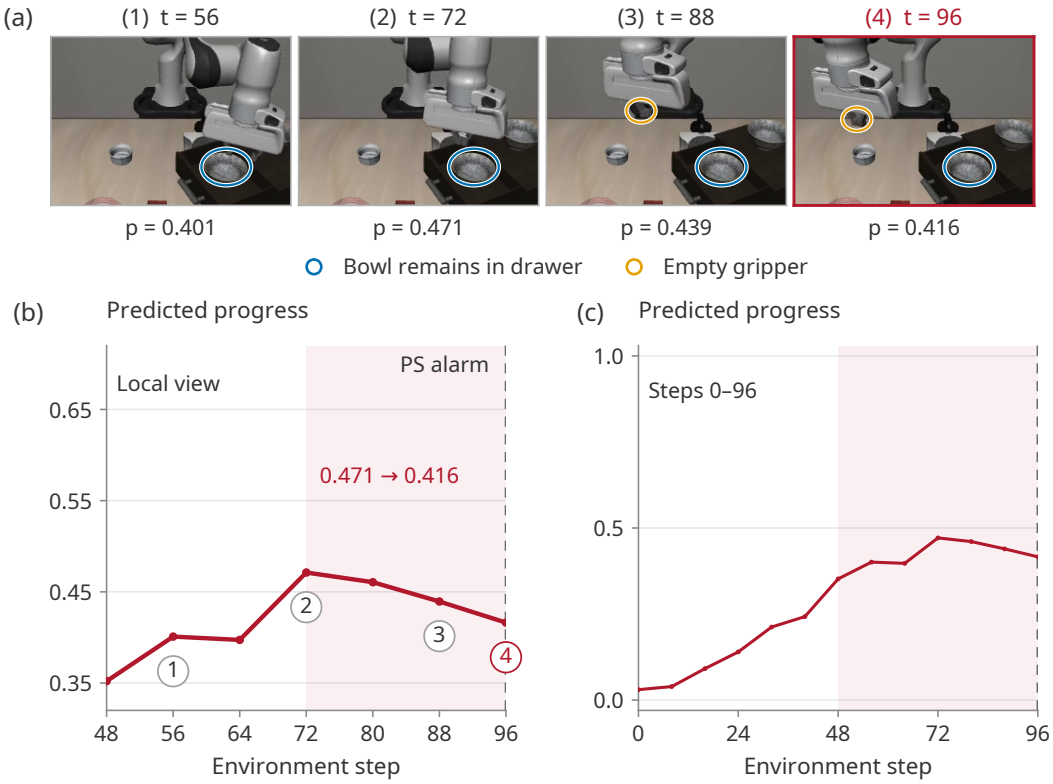

Fig. 7: Predicted progress and video frames during a missed grasp. (a) Four frames from an unsuccessful rollout of the same task as Fig. 6, with their corresponding predicted progress values. Blue outlines mark the bowl, which remains in the drawer; orange outlines mark the empty closed gripper. (b) A local view of the recorded predictions. Numbered markers correspond to the frames in (a). As the empty gripper withdraws, predicted progress decreases from 0.471 at step 72 to 0.416 at step 96, where the recorded PS alarm occurs (dashed line). (c) Predictions from step 0 to step 96, with the interval shown in (b) shaded. Frame times, axis scales, and display windows match Fig. 6; both curves end at step 96 to align with the last frame in (a).

B.3 Failure-Detection Protocols and Robustness Results

This section details task-calibrated evaluation and reports complete FM/DDPM threshold-robustness results, supplementing the FM plots in Fig. 3. All values are percentages. Error bars and table uncertainties show the sample standard deviation over three seeds. Failure-trained baselines are reported separately and excluded from ranking. Computational costs are reported in Appx. B.5; Tab. 27 presents detection performance and costs for PS/MS-PS and all 13 baselines.

Figs. 2 and 3 include all four DoOver detectors and all 13 baselines.

Task-calibrated functional bands. Following the functional conformal prediction protocol used in prior runtime monitors (Xu et al.; Römer et al.; Gu et al.), we construct one-sided time-varying bands from held-out successful rollouts at $\alpha = 0.075$, where $\alpha \in (0, 1)$ is the nominal calibration level. For each task q, let $\boldsymbol{\mathcal{D}}_{\mathrm{CP},q}$ be its finite set of successful threshold-calibration rollout indices and $n = |\boldsymbol{\mathcal{D}}_{\mathrm{CP},q}|$ its size. The band width is the finite-sample-corrected $(1 - \alpha)$ quantile, i.e., the $\lceil (n+1)(1-\alpha) \rceil$-th smallest of the maximum standardized deviations of the n successful threshold-calibration rollouts, and an alarm is raised at the first decision whose score reaches the band. The DDPM evaluation uses the same protocol, covering 39 eligible tasks (LIBERO-10 task 5 has only successful test rollouts).

B.3.1 Robustness to Alarm Threshold Choice: Complete Results

We hold the score functions, taskwise temporal normalization, and test rollouts fixed and vary the empirical calibration quantile over $\alpha \in \{0.01, 0.02, 0.03, 0.05, 0.075, 0.1\}$. The FM and DDPM tables below include all 17 methods, including FE and MS-FE, and report aggregate BA with SD and measured FPR across three seeds. These empirical-quantile results use different alarm thresholds from the finite-sample-corrected bands in Fig. 2(a–d). At each quantile level, we compare each main detector's mean BA with the largest mean BA among baselines without failure data. On FM, the resulting advantages are 1.8–2.7 pp for PS and 2.8–3.6 pp for MS-PS, computed from unrounded means. On DDPM, PS and MS-PS occupy the top two BA positions at all six quantiles, even when the failure-trained references are included (Tabs. 4 and 5).

B.3.2 Threshold Generalization Across Tasks and Suites

We hold detector scores and test rollouts fixed and vary the source of threshold-calibration data. Within-suite pools successful rollouts from the target suite, including the target task; leave-one-task-out (LOTO) uses only the other tasks in that suite; leave-one-suite-out (LOSO) uses only the other three suites. All three apply a scalar threshold to raw scores at $\alpha = 0.075$, distinct from the task-calibrated functional bands above. We report empirical FPR under threshold transfer, without a per-task conformal coverage guarantee.

PS and MS-PS exceed the compared baselines in aggregate BA under every calibration source on both FM and DDPM (Tabs. 6 and 7; Fig. 3(b) shows FM). On FM, MS-PS achieves 70.8%/70.5%/71.2% BA under Within-suite/LOTO/LOSO. The MS-PS/PS margins over the strongest compared baseline increase from 5.9/4.2 pp under LOTO to 10.5/8.8 under LOSO. MS-PS also has the lowest DT across all three sources, at 84.3%–85.4%, with measured FPR of 7.1%–7.3%. On DDPM, MS-PS reaches 70.4%/70.4%/70.6% BA under Within-suite/LOTO/LOSO, and PS reaches 66.7%/66.7%/69.8%.

Tabs. 6 and 7 report BA in panel (a) and DT with measured FPR in panel (b), so that earlier detection can be assessed alongside false alarms. SD are reported for BA and DT, while FPR is shown as a mean; the FM plots in Fig. 3(b,c) show vertical error bars.

B.4 Cross-Policy Failure Detection

Our main evaluations reuse the executing policy's own visual features. This supplementary experiment tests whether a progress signal learned within one policy can also monitor another policy's execution, without changing the executor.

Tab. 2: FM robustness to alarm threshold choice at $\alpha \in \{0.01, 0.02, 0.03\}$. Cells report mean±SD over three seeds, as percentages. FPR is measured at the chosen threshold; α indexes the empirical calibration quantile, not a guaranteed FPR. Results are macro averages over four suites. All 17 methods are retained. † marks methods trained with failure data; S-F denotes success-minus-failure scoring. No rank highlighting is applied.

Method	$\alpha = 0.01$		$\alpha = 0.02$		$\alpha = 0.03$	
	BA ↑	FPR ↓	BA ↑	FPR ↓	BA ↑	FPR ↓
DoOver detectors (no failure data)						
MS-PS	64.2±1.0	4.2±0.3	64.9±1.0	5.0±0.3	66.2±0.9	5.6±0.7
PS	63.1±2.3	5.1±0.2	63.9±2.0	5.8±0.3	65.1±2.0	6.4±0.2
MS-FE	61.5±0.5	5.0±0.5	62.3±0.8	5.5±0.5	63.6±1.5	6.3±0.7
FE	61.1±1.6	4.9±0.2	61.4±1.5	5.6±0.2	62.6±0.6	6.1±0.5
Baselines without failure data						
logpZO	61.2±0.5	5.3±0.5	62.1±0.5	5.9±0.5	63.1±0.4	6.4±0.6
RND-OE	60.8±1.9	4.8±0.8	61.7±1.9	5.2±0.9	62.1±2.2	5.8±1.2
STAC	58.8±1.3	5.5±1.4	59.8±0.6	6.0±1.3	60.5±1.0	6.9±1.3
k-NN	57.7±0.2	5.3±0.6	58.3±0.7	5.8±0.5	59.0±0.9	6.6±0.6
FIPER-AND	56.7±1.1	6.0±0.9	57.0±1.3	6.7±1.0	57.8±0.6	7.7±0.8
STAC-Single	55.8±1.5	5.5±0.7	56.6±1.1	6.0±0.9	57.4±0.4	6.8±1.0
PCA-Kmeans	55.9±1.3	5.1±0.6	56.4±1.6	5.6±0.3	56.5±1.7	6.4±0.2
ACE	55.7±1.4	5.9±0.9	56.0±0.9	6.7±0.8	57.1±1.1	7.6±0.7
Mahalanobis	52.3±0.3	5.6±0.8	52.7±0.1	6.2±0.8	53.7±0.8	7.0±1.0
Baselines with failure data						
k-NN (S-F)†	63.7±0.9	5.2±0.7	64.2±1.2	5.9±0.6	65.0±0.7	6.8±0.9
Mahalanobis (S-F)†	58.6±1.0	6.1±0.9	59.6±0.7	6.8±0.8	60.5±1.2	7.6±0.7
SAFE-LSTM†	56.5±1.0	6.1±0.4	57.0±0.8	6.8±0.6	57.2±0.8	7.4±0.7
SAFE-MLP†	55.1±0.9	6.3±2.6	55.5±0.9	7.0±2.6	56.3±1.0	7.7±2.6

Tab. 3: FM robustness to alarm threshold choice at $\alpha \in \{0.05, 0.075, 0.1\}$. Cells report mean±SD over three seeds, as percentages. FPR is measured at the chosen threshold; α indexes the empirical calibration quantile, not a guaranteed FPR. Results are macro averages over four suites. All 17 methods are retained. † marks methods trained with failure data; S-F denotes success-minus-failure scoring. No rank highlighting is applied.

Method	$\alpha = 0.05$		$\alpha = 0.075$		$\alpha = 0.1$	
	BA ↑	FPR ↓	BA ↑	FPR ↓	BA ↑	FPR ↓
DoOver detectors (no failure data)						
MS-PS	67.8±1.6	8.2±0.7	68.9±1.5	10.3±0.6	70.6±1.7	12.6±0.5
PS	67.3±1.5	8.5±0.6	68.4±1.4	10.5±0.2	69.8±1.1	13.0±0.5
MS-FE	65.6±2.0	8.9±0.2	67.3±1.7	10.8±0.5	67.9±1.6	12.9±0.4
FE	65.2±0.9	8.1±0.7	66.8±1.9	10.4±1.0	68.1±1.6	12.8±0.5
Baselines without failure data						
logpZO	64.0±0.4	8.9±0.5	64.4±0.9	10.8±0.5	65.3±1.3	13.4±0.2
RND-OE	64.7±1.9	8.3±1.2	66.0±1.8	10.5±1.7	67.0±1.1	12.8±2.0
STAC	63.3±0.7	9.3±1.0	64.3±0.4	11.5±1.0	65.6±0.7	13.3±1.4
k-NN	61.6±1.9	8.9±0.6	62.9±2.3	10.7±0.5	63.8±2.4	12.8±0.5
FIPER-AND	59.5±0.9	10.0±0.6	60.9±1.1	12.4±0.7	61.3±2.4	15.1±0.4
STAC-Single	58.1±0.6	9.0±1.3	59.5±0.9	10.7±0.8	61.5±1.8	12.6±0.4
PCA-Kmeans	57.1±0.7	8.1±0.3	58.3±0.4	9.9±0.2	58.6±0.2	12.6±0.9
ACE	58.1±1.5	9.7±0.0	59.6±0.4	11.9±0.6	59.9±1.2	14.6±0.4
Mahalanobis	54.9±0.8	9.5±0.6	57.1±1.2	11.4±1.0	57.4±0.8	14.1±0.5
Baselines with failure data						
k-NN (S-F)†	67.0±0.2	9.1±1.4	68.1±0.6	10.7±1.4	68.3±1.1	12.9±1.0
Mahalanobis (S-F)†	62.8±0.8	9.5±1.1	63.4±0.3	11.5±1.0	64.4±0.8	13.8±0.8
SAFE-LSTM†	58.2±1.4	9.8±0.8	59.3±1.3	11.7±1.3	60.7±2.4	14.0±1.5
SAFE-MLP†	57.5±0.8	9.2±2.4	57.9±0.4	11.4±2.5	59.7±1.1	13.1±2.1

Tab. 4: DDPM robustness to alarm threshold choice at $\alpha \in \{0.01, 0.02, 0.03\}$. BA reports mean±SD over three seeds; FPR reports the mean, as percentages. FPR is measured at the chosen threshold; α indexes the empirical calibration quantile, not a guaranteed FPR. Results are macro averages over four suites. All 17 methods are retained. † marks methods trained with failure data; S-F denotes success-minus-failure scoring. No rank highlighting is applied.

Method	$\alpha = 0.01$		$\alpha = 0.02$		$\alpha = 0.03$	
	BA ↑	FPR ↓	BA ↑	FPR ↓	BA ↑	FPR ↓
DoOver detectors (no failure data)						
MS-PS	64.4±1.3	4.9	65.6±1.4	5.8	66.4±1.3	6.6
PS	64.1±0.8	5.2	65.0±0.7	6.2	65.9±0.5	6.8
MS-FE	64.0±0.9	4.8	64.8±0.6	5.5	65.8±1.1	6.3
FE	62.1±2.0	4.9	63.0±1.5	5.4	63.9±1.6	6.2
Baselines without failure data						
logpZO	61.9±0.2	4.9	62.1±0.3	5.4	62.7±0.8	6.1
RND-OE	59.4±0.3	5.0	59.9±0.4	5.4	60.6±0.8	6.0
STAC	57.1±1.4	4.6	57.3±1.6	5.0	58.2±1.3	5.5
k-NN	57.0±0.8	4.7	57.1±1.1	5.2	57.3±1.0	5.7
FIPER-AND	56.8±1.0	4.8	57.0±1.4	5.5	57.4±1.2	6.0
STAC-Single	53.8±1.6	4.6	54.7±1.4	5.3	55.7±1.8	6.2
PCA-Kmeans	54.1±1.7	5.5	54.7±2.2	6.1	55.2±1.3	7.0
ACE	56.4±1.4	5.1	56.7±1.5	5.8	57.7±0.7	6.5
Mahalanobis	51.9±0.8	5.7	52.0±0.6	6.3	52.1±0.6	7.2
Baselines with failure data						
k-NN (S-F)†	62.9±0.2	5.7	63.0±0.1	6.5	64.0±0.3	7.3
Mahalanobis (S-F)†	61.9±1.8	5.5	62.9±2.3	6.1	63.6±2.3	6.6
SAFE-LSTM†	58.6±1.3	5.2	59.0±1.5	5.9	59.5±1.1	6.4
SAFE-MLP†	56.3±1.9	6.0	56.7±1.9	6.8	57.0±2.0	7.7

Tab. 5: DDPM robustness to alarm threshold choice at $\alpha \in \{0.05, 0.075, 0.1\}$. BA reports mean±SD over three seeds; FPR reports the mean, as percentages. FPR is measured at the chosen threshold; α indexes the empirical calibration quantile, not a guaranteed FPR. Results are macro averages over four suites. All 17 methods are retained. † marks methods trained with failure data; S-F denotes success-minus-failure scoring. No rank highlighting is applied.

Method	$\alpha = 0.05$		$\alpha = 0.075$		$\alpha = 0.1$	
	BA ↑	FPR ↓	BA ↑	FPR ↓	BA ↑	FPR ↓
DoOver detectors (no failure data)						
MS-PS	67.7±0.9	8.4	68.0±0.4	10.3	68.7±1.0	12.0
PS	67.0±0.6	8.9	68.2±0.8	10.6	68.9±0.9	12.7
MS-FE	67.0±0.9	8.5	67.3±0.9	11.0	68.4±2.2	13.4
FE	65.2±1.5	8.6	66.0±1.6	11.0	67.2±1.7	13.3
Baselines without failure data						
logpZO	64.6±1.1	8.3	66.1±1.2	10.3	67.1±0.8	12.8
RND-OE	63.1±1.4	8.0	64.6±1.5	10.1	65.4±1.4	12.4
STAC	59.4±1.1	7.6	60.5±0.7	9.3	61.7±1.0	11.3
k-NN	58.2±1.2	8.1	58.9±0.9	10.1	60.2±0.3	12.1
FIPER-AND	58.0±0.7	8.5	58.6±0.6	10.2	58.9±0.8	12.9
STAC-Single	56.3±1.4	8.4	57.2±0.9	10.5	58.3±1.6	12.9
PCA-Kmeans	55.6±0.9	8.8	56.7±1.1	10.9	57.3±1.0	14.0
ACE	58.1±0.4	8.7	58.3±0.9	10.5	58.8±1.0	12.6
Mahalanobis	52.7±0.6	9.3	53.0±0.6	11.7	53.9±1.0	14.1
Baselines with failure data						
k-NN (S-F)†	65.3±0.8	9.2	66.3±1.2	11.2	67.4±1.4	13.4
Mahalanobis (S-F)†	65.2±0.9	8.7	66.6±1.3	11.1	67.0±1.8	13.5
SAFE-LSTM†	60.8±0.7	8.8	61.7±0.8	11.3	63.5±1.5	13.6
SAFE-MLP†	59.3±2.5	9.7	59.2±2.6	12.8	60.9±0.9	14.6

Tab. 6: FM threshold generalization across tasks and suites. Panel (a) reports BA; panel (b) reports DT with the mean measured FPR in parentheses. Values are mean±SD over three seeds. Bold/underline mark the best/second-best BA and DT among methods without failure data; $^{+}$ marks a DoOver result strictly better than every such baseline. FPR is not ranked.

Method	Within-suite	LOTO	LOSO
(a) BA (%) ↑			
DoOver detectors (no failure data)			
MS-PS	**70.8±1.0**$^{+}$	70.5±1.1^{+}	**71.2±1.5**$^{+}$
PS	68.8±1.2^{+}	68.8±0.8^{+}	69.5±0.9^{+}
MS-FE	70.6±0.7^{+}	**70.6±0.3**$^{+}$	70.7±0.4^{+}
FE	68.1±0.3^{+}	68.4±0.2^{+}	68.9±0.1^{+}
Baselines without failure data			
logpZO	62.3±0.8	61.8±1.1	59.1±0.3
RND-OE	64.7±1.4	63.8±2.0	58.3±0.7
STAC	65.6±0.8	64.6±1.3	60.7±0.7
k-NN	54.9±0.6	53.2±0.9	50.6±0.3
FIPER-AND	56.4±0.6	55.2±0.9	56.1±0.5
STAC-Single	55.5±0.5	55.0±0.2	51.7±0.2
PCA-Kmeans	57.0±1.2	57.2±0.7	54.4±1.8
ACE	52.3±1.4	51.4±0.9	50.0±0.0
Mahalanobis	52.2±0.1	52.0±0.4	49.9±0.1
Baselines with failure data			
k-NN (S-F)	53.0±0.1	53.1±0.3	52.2±0.6
Mahalanobis (S-F)	52.5±0.3	52.4±0.7	51.9±0.6
SAFE-LSTM	57.0±0.5	57.1±1.2	59.1±0.4
SAFE-MLP	57.3±0.5	56.8±0.4	52.8±0.1
(b) DT (%) ↓ (measured FPR (%) ↓)			
DoOver detectors (no failure data)			
MS-PS	**85.4±0.2**$^{+}$(7.1)	**85.2±0.2**$^{+}$(7.3)	**84.3±1.2**$^{+}$(7.3)
PS	87.8±0.6^{+}(6.0)	87.7±0.3 (6.1)	84.4±2.6^{+}(10.6)
MS-FE	85.9±0.5^{+}(7.2)	85.8±0.4^{+}(7.4)	85.6±0.3^{+}(6.0)
FE	87.6±0.1^{+}(6.3)	87.4±0.4 (6.3)	86.7±0.2^{+}(5.8)
Baselines without failure data			
logpZO	87.9±1.1 (7.1)	86.1±0.6 (8.5)	89.2±0.2 (8.2)
RND-OE	89.9±0.9 (7.7)	89.6±1.5 (8.3)	91.1±0.5 (11.1)
STAC	90.4±0.7 (8.0)	90.0±0.6 (9.6)	90.3±0.5 (13.7)
k-NN	98.0±0.3 (5.9)	96.8±0.2 (8.8)	89.3±0.5 (23.8)
FIPER-AND	90.2±1.1 (7.0)	88.8±0.4 (10.3)	89.8±0.7 (7.2)
STAC-Single	96.8±0.1 (7.4)	96.2±0.3 (9.5)	95.2±0.3 (17.9)
PCA-Kmeans	94.5±0.9 (7.1)	94.0±0.5 (7.7)	92.5±1.6 (12.4)
ACE	99.3±0.1 (6.0)	98.1±0.2 (8.9)	93.1±0.2 (22.5)
Mahalanobis	99.0±0.2 (6.4)	97.2±0.3 (9.3)	92.0±0.3 (22.8)
Baselines with failure data			
k-NN (S-F)	88.8±0.8 (7.2)	87.0±0.3 (8.5)	89.8±0.4 (6.9)
Mahalanobis (S-F)	88.8±2.1 (7.9)	87.1±2.3 (9.8)	87.8±1.7 (8.9)
SAFE-LSTM	92.9±0.3 (5.6)	91.5±0.5 (7.8)	90.0±0.3 (6.4)
SAFE-MLP	96.2±0.4 (6.7)	95.1±0.5 (9.6)	94.9±0.1 (7.7)

Tab. 7: DDPM threshold generalization across tasks and suites with raw-score scalar thresholds at $\alpha = 0.075$. Panel (a) reports BA; panel (b) reports DT with mean measured FPR in parentheses. All values are percentages, macro-averaged over four suites. BA and DT report mean±SD over three seeds; FPR reports the mean. Bold/underline mark the best/second-best BA and DT among methods without failure data; $^{+}$ marks a DoOver result strictly better than every such baseline. FPR is not ranked; † marks failure-trained references.

Method	Within-suite	LOTO	LOSO
(a) BA (%) ↑			
DoOver detectors (no failure data)			
MS-PS	**70.4±1.6**$^{+}$	**70.4±1.9**$^{+}$	**70.6±0.9**$^{+}$
PS	66.7±0.3^{+}	66.7±0.2^{+}	69.8±0.7^{+}
MS-FE	67.2±1.1^{+}	66.6±0.8^{+}	68.7±0.5^{+}
FE	65.2±0.9^{+}	64.7±0.6	68.6±0.4^{+}
Baselines without failure data			
logpZO	65.2±0.5	65.4±0.4	59.9±1.2
RND-OE	63.3±0.6	62.8±0.7	50.9±0.2
STAC	64.0±1.8	63.9±1.6	61.3±0.4
k-NN	56.4±0.8	55.3±0.7	50.0±0.0
FIPER-AND	56.1±0.3	55.4±0.4	55.0±0.3
STAC-Single	58.7±0.4	57.8±0.6	51.8±0.4
PCA-Kmeans	54.3±0.7	54.4±1.1	50.4±0.1
ACE	53.9±0.8	52.5±0.5	50.0±0.0
Mahalanobis	52.4±1.1	53.0±0.1	50.0±0.0
Baselines with failure data			
k-NN (S-F)†	54.6±0.7	53.8±0.9	54.8±0.7
Mahalanobis (S-F)†	56.6±1.2	56.0±0.7	57.2±1.2
SAFE-LSTM†	61.2±1.6	60.6±2.3	58.1±1.4
SAFE-MLP†	58.2±0.8	57.9±0.9	53.4±1.1
(b) DT (%) ↓ (measured FPR (%) ↓)			
DoOver detectors (no failure data)			
MS-PS	85.3±1.6 (6.5)	85.1±1.7 (6.9)	84.7±1.6 (7.5)
PS	88.1±0.4 (6.5)	88.1±0.3 (6.5)	85.9±1.2 (6.9)
MS-FE	87.7±0.8 (7.1)	87.2±0.5 (8.6)	86.6±0.5 (6.9)
FE	88.8±0.5 (6.9)	88.6±0.3 (7.8)	86.8±0.4 (6.9)
Baselines without failure data			
logpZO	**84.9±0.2** (8.3)	**84.5±0.0** (8.4)	**76.1±1.1** (17.7)
RND-OE	89.0±0.4 (8.9)	88.9±0.3 (9.3)	79.5±0.3 (25.0)
STAC	91.4±1.0 (7.3)	91.2±1.0 (7.8)	90.5±0.9 (12.3)
k-NN	97.3±0.3 (7.5)	96.0±0.2 (10.3)	83.3±0.2 (25.0)
FIPER-AND	91.3±0.5 (6.5)	89.5±0.4 (8.9)	91.1±0.8 (6.9)
STAC-Single	96.8±0.1 (6.7)	96.0±0.3 (9.0)	94.9±0.6 (19.1)
PCA-Kmeans	95.1±0.9 (7.6)	94.2±1.0 (8.9)	79.0±1.1 (25.2)
ACE	99.1±0.1 (6.8)	97.7±0.1 (11.0)	91.9±0.2 (25.0)
Mahalanobis	98.7±0.2 (8.0)	96.8±0.1 (9.5)	87.3±0.3 (25.0)
Baselines with failure data			
k-NN (S-F)†	91.2±0.6 (7.6)	89.5±0.4 (9.7)	88.3±0.7 (8.7)
Mahalanobis (S-F)†	89.8±1.3 (8.2)	87.5±0.9 (10.1)	87.8±2.0 (8.7)
SAFE-LSTM†	89.8±2.1 (8.1)	88.6±2.1 (10.1)	88.9±2.2 (8.8)
SAFE-MLP†	95.9±1.0 (8.4)	94.9±1.1 (10.8)	93.1±0.7 (9.4)

Executing policy, monitoring models, and evaluation tasks. A non-DoOver frozen FM policy without a progress head executes the tasks. Separately trained DoOver detectors with both FM and DDPM backbones use image-only progress heads learned from the same suite demonstrations. Each DoOver detector processes the executor's images with its own encoder and head, without running its denoiser or modifying the executed actions. All failure detection methods share the frozen executor rollouts, disjoint fitting, normalization, calibration, and test sets. Results are averaged over three seeds. Tab. 8 compares eight DoOver configurations (two backbones and four detectors) with 13 baselines.

Task eligibility follows the frozen executor's outcomes before any detector is compared. LIBERO-Spatial task 5 lacks enough successful threshold-calibration rollouts for a finite corrected threshold at $\alpha = 0.075$ and LIBERO-Goal task 4 has only successful test outcomes. Both are excluded for every failure detection method, leaving 38 tasks for the BA, AP and FPR comparisons.

Task-calibrated detection. Functional bands are calibrated on successful rollouts from the executor at $\alpha = 0.075$. All four detectors exceed the nine baselines without failure data in AP (Fig. 2(e)). PS achieves 64.8% BA with the FM observer and 68.2% with the DDPM observer, compared with 60.1% for logpZO. For either observer backbone, PS or MS-PS ranks first in each aggregate BA, AP, and ROC-AUC comparison, including all four failure-trained baselines (Tab. 8). Fig. 2(e) compares AP for both observer backbones; the baselines use the same executor rollouts and are shared between observers.

Threshold generalization and alarm-threshold sweeps. The raw-score Within-suite, LOTO, and LOSO protocols follow Appx. B.3.2. MS-PS for DDPM achieves 72.0%/71.8%/74.2% BA under all the three sources (Tab. 9–10). Under LOTO and LOSO, PS exceeds the compared baselines in BA with either observer backbones. Both main detectors also exceed the compared baselines throughout the six-quantile sweep (Tab. 11–12). As in Appx. B.3.1, these sweep thresholds use empirical quantiles without finite-sample correction.

Tab. 8: Cross-policy detection with task-calibrated functional bands at $\alpha = 0.075$. FM/DDPM identify the DoOver observer's backbone; the executor is FM. All values are percentages; $\pm$ denotes one sample SD over three seeds. †: also trained on failed rollouts.

Method	BA ↑	AP ↑	ROC-AUC ↑	DT ↓	FPR ↓
MS-PS (FM)	63.6±1.7	54.7±0.6	79.2±1.7	90.9±0.6	4.4
MS-PS (DDPM)	68.3±1.4	61.9±2.4	82.3±0.5	89.2±1.4	4.3
PS (FM)	64.8±1.2	54.5±1.8	79.0±0.9	89.3±1.2	4.1
PS (DDPM)	68.2±0.6	61.4±3.8	83.0±0.8	88.5±0.3	4.4
MS-FE (FM)	62.5±2.1	53.1±1.8	77.1±1.1	90.0±1.8	4.4
MS-FE (DDPM)	66.4±2.1	59.9±2.1	81.0±0.4	89.3±0.4	4.7
FE (FM)	62.1±1.9	51.6±0.8	77.9±1.0	91.5±1.4	4.2
FE (DDPM)	65.7±0.8	59.0±1.2	79.9±1.6	90.5±0.1	4.2
logpZO	60.1±0.9	45.8±1.7	73.2±0.6	88.8±1.6	4.0
RND-OE	57.2±2.1	42.2±2.3	72.7±0.8	92.6±2.4	4.5
STAC	58.0±2.2	42.9±1.5	72.0±0.9	91.1±3.1	4.1
k-NN	58.3±0.2	41.5±0.5	72.4±0.8	88.9±1.8	4.6
FIPER-AND	55.8±0.6	38.5±2.8	68.3±0.9	93.3±0.7	5.2
STAC-Single	53.5±0.8	31.8±1.0	60.5±0.6	94.3±1.4	5.8
PCA-Kmeans	53.7±1.0	31.9±0.7	66.3±0.9	94.8±0.7	4.5
ACE	56.1±0.3	37.1±0.3	67.8±0.2	92.4±1.0	4.8
Mahalanobis	55.3±1.3	33.0±0.6	64.2±0.2	88.6±2.3	4.7
k-NN (S-F)†	61.5±1.1	48.0±2.3	76.8±0.8	90.0±0.8	4.9
Mahalanobis (S-F)†	59.4±1.2	43.7±1.3	73.0±0.8	89.5±1.1	5.0
SAFE-LSTM†	54.5±0.4	37.2±0.7	67.9±0.3	93.7±1.0	4.3
SAFE-MLP†	56.5±0.6	39.4±2.3	67.1±0.8	94.1±0.2	4.0

B.5 Policy-Native Detection Cost Details

Tab. 13 reports the audited LIBERO-Goal FM reference configuration on an NVIDIA RTX 5090 GPU and an AMD Ryzen 9 9950X CPU. We report network weights and stored reference data separately, as well as batched sampling and detector scoring. These measurements cover our jointly trained, policy-native method. The supplementary experiment that adds a head to a frozen policy ap-

Tab. 9: Cross-policy threshold generalization: BA (measured FPR) with raw scalar thresholds at $\alpha = 0.075$. Superscripts count the suite–seed combinations with BA near 50%, out of 12 combinations. All values are percentages; ± denotes one sample SD over three seeds. †: also trained on failed rollouts.

Method	Within-suite	LOTO	LOSO
MS-PS (FM)	68.1±1.4 (6.5)	67.9±1.6 (6.4)	70.7±1.1 (6.6)
MS-PS (DDPM)	72.0±1.9 (6.5)	71.8±1.4 (6.6)	74.2±1.8 (6.5)
PS (FM)	66.6±1.5 (5.3)	66.5±1.4 (5.3)	68.4±1.3 (10.4)
PS (DDPM)	70.4±0.8 (4.6)	70.1±1.0 (5.0)	70.6±1.4 (8.6)
MS-FE (FM)	68.3±1.5 (7.0)	68.3±1.8 (7.4)	69.7±0.4 (6.1)
MS-FE (DDPM)	72.2±0.4 (6.7)	71.9±0.2 (6.7)	71.6±0.9 (5.8)
FE (FM)	67.2±1.4 (6.7)	67.4±1.6 (6.7)	68.3±0.7 (6.4)
FE (DDPM)	71.2±1.8 (6.6)	70.9±1.4 (6.8)	71.2±1.2 (6.3)
logpZO	61.0±0.6 (6.8)	60.8±0.3 (8.7)	56.8±0.2 (10.0)
RND-OE	60.7±1.7 (7.7)	59.4±0.7 (8.3)	56.5±0.9 (12.7)$^{2/12}$
STAC	60.4±0.3 (7.2)	60.1±0.4 (8.6)	58.3±1.2 (13.2)
k-NN	54.9±1.1 (6.8)	53.3±1.5 (9.8)$^{2/12}$	50.2±0.3 (24.5)$^{10/12}$
FIPER-AND	55.3±0.8 (6.3)	54.9±0.7 (9.4)	54.4±1.2 (6.5)
STAC-Single	56.8±1.1 (7.0)	56.0±1.4 (9.2)$^{3/12}$	52.5±0.1 (16.4)$^{3/12}$
PCA-Kmeans	53.2±0.6 (6.2)	53.9±0.4 (7.1)	52.3±1.4 (17.8)
ACE	53.7±1.9 (6.3)	51.2±0.9 (10.8)$^{8/12}$	50.0±0.0 (22.5)$^{12/12}$
Mahalanobis	50.3±0.5 (7.1)$^{1/12}$	50.5±1.4 (10.5)$^{9/12}$	50.5±0.2 (23.9)$^{9/12}$
k-NN (S-F)†	53.4±0.3 (6.1)	52.9±0.4 (7.5)	52.4±0.4 (6.9)
Mahalanobis (S-F)†	53.3±0.8 (7.3)	53.4±1.2 (8.7)	51.8±0.6 (7.8)
SAFE-LSTM†	61.4±1.6 (7.6)	60.3±0.4 (8.7)	60.1±2.4 (8.3)
SAFE-MLP†	59.2±0.7 (7.5)	59.0±0.3 (10.1)	52.9±0.1 (6.8)

Tab. 10: Cross-policy threshold generalization: DT. Read together with BA and measured FPR in Tab. 9. All values are percentages; ± denotes one sample SD over three seeds. †: also trained on failed rollouts.

Method	Within-suite	LOTO	LOSO
MS-PS (FM)	86.3±0.3	86.5±0.9	83.8±0.1
MS-PS (DDPM)	84.4±0.4	84.7±0.8	83.0±1.6
PS (FM)	87.7±1.1	87.7±1.0	82.6±2.6
PS (DDPM)	86.5±0.2	86.4±0.2	83.0±2.6
MS-FE (FM)	85.9±1.6	85.7±1.6	85.5±0.2
MS-FE (DDPM)	84.0±0.8	84.1±0.3	84.6±0.8
FE (FM)	86.7±1.4	86.6±1.5	85.8±0.9
FE (DDPM)	85.0±1.2	84.9±1.3	84.6±0.7
logpZO	87.1±2.7	85.3±2.1	87.7±1.2
RND-OE	90.6±0.7	90.6±0.7	89.3±0.8
STAC	92.6±0.2	92.1±0.0	91.2±0.2
k-NN	97.9±0.3	96.6±0.2	87.9±0.5
FIPER-AND	90.9±0.2	88.9±0.5	91.1±1.1
STAC-Single	97.3±0.2	96.4±0.4	95.3±0.1
PCA-Kmeans	96.0±0.2	95.0±0.2	89.1±1.0
ACE	99.0±0.1	97.7±0.0	93.4±0.2
Mahalanobis	99.4±0.0	97.5±0.4	90.6±0.3
k-NN (S-F)†	90.0±0.9	89.0±0.6	89.7±1.1
Mahalanobis (S-F)†	86.8±1.8	85.3±2.1	88.9±1.4
SAFE-LSTM†	89.6±1.1	89.4±1.3	89.5±1.5
SAFE-MLP†	94.5±0.1	93.4±0.4	94.7±0.3

Tab. 11: Cross-policy BA over the full empirical-quantile sweep. All values are percentages; ± denotes one sample SD over three seeds. †: also trained on failed rollouts.

Method	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$	$\alpha = 0.05$	$\alpha = 0.075$	$\alpha = 0.10$
MS-PS (FM)	64.3±1.3	64.8±0.9	65.6±0.5	67.4±0.6	69.1±1.3	69.1±1.3
MS-PS (DDPM)	68.9±1.0	69.7±1.0	70.3±1.0	72.8±0.3	73.4±0.7	73.8±0.5
PS (FM)	65.3±0.9	65.9±0.5	66.3±0.4	68.5±1.6	70.0±0.5	70.2±0.3
PS (DDPM)	68.9±0.4	69.6±0.1	70.2±0.6	71.9±1.2	72.6±0.9	72.5±1.7
MS-FE (FM)	63.3±1.6	64.0±1.0	64.6±0.8	66.1±0.7	67.1±0.7	68.9±1.8
MS-FE (DDPM)	67.6±1.2	68.0±1.0	68.6±0.7	70.5±0.6	71.3±0.6	72.7±0.3
FE (FM)	63.3±2.1	63.6±1.7	64.3±1.0	67.3±0.2	68.8±0.8	70.6±0.9
FE (DDPM)	66.4±1.3	66.9±1.5	67.6±1.5	69.2±1.2	70.8±1.4	70.7±1.1
logpZO	60.7±1.0	61.4±0.8	62.4±0.9	63.9±2.0	65.4±1.3	65.5±1.2
RND-OE	57.5±2.0	57.6±2.3	58.4±2.5	61.6±1.1	63.5±2.0	64.7±1.6
STAC	58.3±2.4	59.2±2.4	60.1±2.8	62.8±0.7	63.4±0.5	64.1±0.6
k-NN	58.5±0.3	59.0±0.2	59.0±0.2	60.6±0.6	61.8±0.4	63.2±1.5
FIPER-AND	56.3±0.8	56.7±0.4	56.9±0.4	57.6±1.5	58.8±1.1	59.5±1.2
STAC-Single	53.7±0.5	53.9±0.6	54.4±0.5	54.8±0.4	55.6±0.6	56.6±0.8
PCA-Kmeans	53.9±1.0	54.1±0.7	54.3±0.6	54.2±0.5	55.9±1.1	57.2±2.0
ACE	56.5±0.4	56.9±0.3	57.2±0.3	57.5±1.3	57.6±1.6	59.4±1.0
Mahalanobis	55.7±1.5	56.1±1.4	56.3±0.8	57.3±1.3	57.7±1.3	58.4±1.5
k-NN (S-F)†	62.1±1.7	62.6±1.8	63.7±0.9	65.0±0.7	65.9±0.7	67.1±0.4
Mahalanobis (S-F)†	60.3±0.9	61.2±1.3	61.5±1.2	62.3±0.7	64.3±1.7	65.1±1.4
SAFE-LSTM†	55.3±0.5	55.3±0.5	55.8±1.0	56.8±1.4	58.5±0.8	60.3±0.4
SAFE-MLP†	57.1±0.9	58.2±1.8	58.9±1.9	60.0±2.1	60.9±1.9	61.2±1.9

Tab. 12: Cross-policy FPR over the full empirical-quantile sweep. Values are mean percentages over three seeds; FPR SDs were not reported. †: also trained on failed rollouts.

Method	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$	$\alpha = 0.05$	$\alpha = 0.075$	$\alpha = 0.10$
MS-PS (FM)	4.8	5.2	5.8	7.7	9.7	12.4
MS-PS (DDPM)	4.8	5.4	6.1	8.5	10.4	12.6
PS (FM)	4.4	4.9	5.5	7.8	9.7	12.0
PS (DDPM)	4.8	5.3	6.1	8.1	10.0	12.5
MS-FE (FM)	4.8	5.4	5.9	8.1	10.2	12.8
MS-FE (DDPM)	5.4	6.1	6.8	9.1	11.3	13.2
FE (FM)	4.7	5.5	6.2	8.0	10.2	12.9
FE (DDPM)	4.6	5.2	6.1	8.5	10.5	13.2
logpZO	4.3	4.7	5.4	7.6	9.7	11.8
RND-OE	5.0	5.9	6.8	9.1	11.2	13.8
STAC	4.7	5.2	6.2	8.7	10.8	13.0
k-NN	5.0	5.5	6.3	8.2	9.8	11.9
FIPER-AND	5.6	6.1	7.1	9.4	11.5	13.7
STAC-Single	6.4	7.2	8.1	11.3	13.8	16.0
PCA-Kmeans	5.2	5.7	6.4	9.1	11.3	13.6
ACE	5.3	6.0	6.7	9.1	11.2	13.5
Mahalanobis	5.0	5.6	6.3	8.2	10.6	12.9
k-NN (S-F)†	5.4	5.9	6.4	8.3	10.1	12.5
Mahalanobis (S-F)†	5.5	6.1	7.0	9.2	11.2	13.7
SAFE-LSTM†	4.8	5.5	6.4	8.6	10.4	12.7
SAFE-MLP†	4.4	5.1	5.8	7.7	10.1	12.3

pears in Appx. B.6.1. At runtime, the policy already computes visual features for action generation, and the progress head reads those features inside the same forward pass.

Fig. 4(a) summarizes additional deployed storage and post-policy fitting time. The following paragraphs detail model storage, reference-bank scaling, training, and inference costs.

Size. In Tab. 13, size includes FP32 network weights (including RND targets) or stored reference data and statistics, excluding the shared policy, threshold layer, and optimizer. DoOver's four rules share one head; sizes are not additive.

Model size and storage. The progress head adds only $526{,}480$ parameters, a 1.5% increase over the headless policy, and occupies 2.01 MiB in FP32. By comparison, logpZO adds 62.07 million parameters and RND-OE adds 42.49 million including its frozen target, approximately $118\times$ and $81\times$ the head's parameter count. logpZO also requires a separate FP32 model file of 236.78 MiB (approximately 248 MB in decimal units), whereas the progress head is stored inside the policy checkpoint. Methods without learned parameters still need storage: the evaluated k-NN and Mahalanobis variants retain 57.16–256.06 MiB of reference embeddings or covariance statistics, roughly 28–$128\times$ the head's weight storage. This range includes the success-minus-failure variants, which retain both success and failure reference state. DoOver's policy-native monitor reuses the policy's visual representation and carries no separate high-dimensional reference bank.

Effect of reference-set size. The k-NN storage and query cost also grow with the reference bank. In a separate pooled-bank comparison, increasing the bank from $1{,}829$ to $6{,}159$ embeddings ($3.37\times$) increases scoring time from 0.756 to 2.582 ms per decision ($3.42\times$), approximately linear over this audited range. DoOver's fixed monitoring head and short progress history require no nearest-neighbor query. Mahalanobis instead retains one or two 4096×4096 inverse covariance matrices; at fixed embedding dimension, its storage and per-query computation do not grow with the calibration-bank size after fitting.

Training cost. In our main configuration, the jointly trained DoOver head is optimized in the same backward pass and optimizer step as the policy, with no post-policy detector training. Its supervision comes from existing demonstration timestamps, requiring no additional training rollouts or manual progress labels. In the cost analysis experiment, elapsed time between matching training checkpoints indicates 0.183% additional elapsed training time. In contrast, logpZO requires 2.45 GPU-hours of post-policy fitting on one suite and 13.11 GPU-hours across all four suites. Its LIBERO-Goal fit alone exceeds the approximately 1.16 GPU-hours estimated to train the entire visuomotor policy. Using the same checkpoint timing method, the head's marginal training time is about 7.6 s; logpZO's $8{,}835.667$ s fit is therefore approximately $1{,}200\times$ that incremental cost. This ratio compares additional detector-training time, rather than total policy-training time or DoOver's zero post-policy fitting time. Using the median time between checkpoints instead gives 8.48 s (0.204%) for the head on LIBERO-Goal and a corresponding ratio of approximately $1{,}000\times$. The two estimates differ because the added training time is small relative to the total time measured between checkpoints; the ratios should therefore be read as approximate. Tab. 13 measures the cost of learning and evaluating the monitoring signal; rollout collection and threshold estimation are outside the reported training costs.

Inference cost. The head adds 1.05 million counted FLOPs, approximately 0.00227% of an FM decision, and its separately measured forward pass takes 0.019 ms, equivalent to 0.077% of the 24.70 ms reference policy decision. It runs inside the policy forward pass, requiring neither another observation encoding nor another policy evaluation. The sampling-based monitors STAC, ACE, and FIPER-AND instead use 256 action chunks per decision. Let K denote the per-decision sample count. Generating the batch takes 150.00 ms instead of 24.70 ms: an additional 125.30 ms, or approximately $5.1\times$ the nominal decision time, before detector scoring. Relative to the head's forward-pass time of 0.019 ms, this generation overhead alone is approximately $6{,}600\times$ larger. Adding the audited scoring times gives estimated total overheads of 126.21 ms for STAC and 126.69 ms for ACE, approximately 6.6×10^3 and 6.7×10^3 times the head's forward-pass time, respectively. The extra generation adds 2.18 TFLOPs per decision ($46.94\times$ the base decision); full detector FLOPs were not measured. Policy-side peak GPU memory rises from 228.9 MiB at $K = 1$ to 268.4 MiB at $K = 256$; these peaks are measured separately from, and are not additive with, the table's scorer

Tab. 13: Policy-native detection costs in the LIBERO-Goal FM reference configuration, covering all 17 detectors. **Fit type:** J, joint training with the policy; G, gradient-based training after policy training; N, nonparametric fitting; G+N, both G and N. **Fit data:** D, demonstrations; S, successful rollouts; F, failed rollouts; S+F, both successful and failed rollouts. S-F in method names denotes success-minus-failure scoring. All methods use successful rollouts for benchmark calibration. Fit time covers post-policy fitting, excluding calibration; J's zero fit time means no separate fitting stage. **Superscripts:** ^{d}, derived from the shared implementation, not separately measured; ^{u}, no method-specific audit record; ^{h}, progress-head forward-pass time inside the policy, in milliseconds (not hours), excluding full detector execution; †, also fitted on failed rollouts. **Other symbols:** n/c, no comparable scoring measurement; –, unavailable or not applicable; 0.0 in replay GPU memory, CPU-only scoring; $<$, an upper bound. **Units:** M, million parameters; MiB, 2^{20} bytes; s, seconds; ms, milliseconds. Training, sampling, and memory comparisons appear in Appx. B.5.

(a) Model, fitting, and data requirements

Method	Extra params (M)	Size (MiB)	Fit type	Fit (s)	Fit data	External networks
MS-PS	0.526	2.01	J	0	D	0
PS	0.526	2.01	J	0	D	0
FE	0.526	2.01	J	0	D	0
MS-FE^{u}	0.526^{d}	2.01^{d}	J^{d}	0^{d}	D^{d}	0^{d}
logpZO	62.071	236.78	G	8,835.667	D	1
RND-OE	42.486	162.07	G	6.642	S	2
STAC	0	< 0.001	–	0	–	0
k-NN	0	57.16	N	0.036	S	0
FIPER-AND	42.486	162.07	G+N	10.203	S	2
STAC-Single	0	< 0.001	–	0	–	0
PCA-Kmeans	0	0.35	N	0.363	S	0
ACE	0	< 0.001	N	0.333	S	0
Mahalanobis	0	128.03	N	10.676	S	0
k-NN (S-F)†	0	143.84	N	0.048	S+F	0
Mahalanobis (S-F)†	0	256.06	N	20.969	S+F	0
SAFE-LSTM†	4.459	17.01	G	9.708	S+F	1
SAFE-MLP†	1.049	4.00	G	4.013	S+F	1

(b) Per-decision computation and replay memory

Method	Extra samples	Generation overhead (ms)	Score (ms)	Estimated total overhead (ms)	Replay GPU peak (MiB)
MS-PS	0	0	0.01900^{h}	0.019^{h}	16.2
PS	0	0	0.01900^{h}	0.019^{h}	16.2
FE	0	0	0.01900^{h}	0.019^{h}	16.2
MS-FE^{u}	–	–	–	–	–
logpZO	0	0	0.42330	0.423	–
RND-OE	0	0	0.03391	0.034	278.2
STAC	255	125.30	0.90773	126.209	20.3
k-NN	0	0	0.75602	0.756	0.0
FIPER-AND	255	125.30	1.41492	126.716	278.2
STAC-Single	0	0	0.00199	0.002	20.3
PCA-Kmeans	0	0	0.01006	0.010	0.0
ACE	255	125.30	1.38778	126.689	0.0
Mahalanobis	0	0	n/c	n/c	0.0
k-NN (S-F)†	0	0	2.09552	2.096	0.0
Mahalanobis (S-F)†	0	0	n/c	n/c	0.0
SAFE-LSTM†	0	0	0.01355	0.014	255.7
SAFE-MLP†	0	0	0.00722	0.007	25.6

replay peaks. DoOver avoids this repeated sampling entirely. The 0.019 ms measurement covers only the forward pass through the head, not a separately measured end-to-end streaming detector.

Detection quality and deployment requirements. Together with the aggregate detection results in Sec. 4.2, these comparisons show that DoOver's consistent performance comparable to baselines without failure data on both backbones requires little additional computation and storage. The jointly trained head uses demonstration timestamps without failure labels, reuses the policy's encoder, and requires no runtime action resampling. Appx. C relates these resource differences to the design of each baseline family and the observed quality comparisons.

B.6 Ablation Details

B.6.1 Adding a Progress Head to a Frozen Policy

DoOver's primary design jointly trains a policy-native progress head. To test the additional capability of adding this monitor to an existing policy without retraining it, we freeze a headless FM-DiT policy on LIBERO-Goal, including its observation encoder, and train only a newly initialized two-layer progress head using the same demonstration-timestamp supervision as joint training. Only $526{,}480$ of $35{,}633{,}487$ parameters (1.48%) are trainable. After training, all 173 original policy tensors are bitwise identical to the source checkpoint: adding the monitor leaves the action policy intact.

Setup. This supplementary experiment uses ten LIBERO-Goal tasks. The policy without a head and the jointly trained policy use their $20{,}000$-step checkpoints. The added head is trained for $20{,}000$ steps, with the same $40{,}000$-step learning-rate schedule and progress-head configuration as the joint run. Each policy supplies its own calibration and test rollouts. We evaluate both configurations with the same detection procedure, per-timestep normalization, and corrected conformal thresholds at $\alpha = 0.075$. For each configuration, we fix task horizons using its own calibration data, following Appx. B.1.3. We average results over the same nine tasks that have both successful and failed test episodes under each policy; the remaining task has no failures under the frozen policy, which is unavailable. The comparison therefore uses a common task set, with separate policy rollouts and calibration horizons.

Tab. 14: Adding a progress head to a frozen FM policy on LIBERO-Goal. BA and AP are means over the same nine eligible tasks. With the policy frozen, only the added head is trained.

	Joint training		Frozen policy + new head	
Detector	BA ↑	AP ↑	BA ↑	AP ↑
PS	0.650	0.574	0.659	0.594
MS-PS	0.672	0.592	0.668	0.582
FE	0.645	0.563	0.657	0.580
MS-FE	0.658	0.569	0.681	0.578

Detection without policy updates. The frozen-policy monitor retains BA and AP close to joint training across all four detectors (Tab. 14). PS reaches BA/AP of $0.659/0.594$, compared with $0.650/0.574$ for joint training; MS-PS remains within 0.004 BA and 0.010 AP of its jointly trained counterpart. In this FM experiment, training only the new head provides useful monitoring without changing the action policy or its visual features. Joint training remains our primary method; this supplementary result shows that the monitor can also be added to an existing policy.

B.6.2 Input Modality (Proprioception)

We compare vision-only progress prediction with a head that additionally receives the eight-dimensional end-effector and gripper state, keeping the policy's observation inputs unchanged. We replay PS on outcome-matched rollouts. BA is pooled within each suite; DT is the mean first-alarm environment step over failures detected with both input configurations. Both metrics then average the four suites equally.

Tab. 15: Progress-head ablations. (a) PS diagnostic on FM with rollouts matched by outcome and unchanged policy inputs. BA pools episodes within each suite; DT is the mean first-alarm environment step over failures detected with both input configurations. Both average the four suites equally. (b) Policy SR with recovery disabled, averaged over three seeds and four equally weighted LIBERO suites. Δ: with head minus without head (pp). In each row, bold marks the better value; Δ is not marked. Protocols and aggregation details: Appx. B.6.

(a) Progress-head input

Metric	Vision-only	Vision + proprioception
BA (%) ↑	91.62	**93.10**
DT (env. steps) ↓	**108.8**	125.0
Earliest DT across tested windows (env. steps) ↓	**82.7**	100.5

(b) Effect on policy SR

	SR (%) ↑		
Policy	Without head	With head	Δ (pp)
FM	83.18	**84.00**	+0.82
DDPM	75.05	**78.20**	+3.15

Adding proprioception changes BA from 91.62% to 93.10% and DT from 108.8 to 125.0 environment steps (Tab. 15(a)). The table also reports the earliest DT across the tested windows. The timing difference motivates vision-only monitoring when prompt alarms are the priority. A possible explanation is that robot-state cues can track trajectory phase despite an unsuccessful interaction, consistent with the robot-state reliance discussed in FAIL-Detect and RISE (Xu et al., 2025; Yang et al., 2026).

For example, after an object slips from the gripper, the end effector may continue toward its destination even though the intended object motion has stopped. Proprioception can still indicate advancement along a demonstrated arm trajectory, whereas visual observations can reveal the failed interaction. This provides a possible explanation for the timing difference: a low-dimensional robot-state input can be highly predictive of demonstration timestamps.

Recorded progress traces. Fig. 8 shows a successful and a failed rollout from the same FM policy and LIBERO-Object task, with recovery disabled. The successful execution advances steadily, while the failed execution stalls and regresses. The marked alarm is the recorded PS alarm. These full-rollout examples illustrate the monitoring signal.

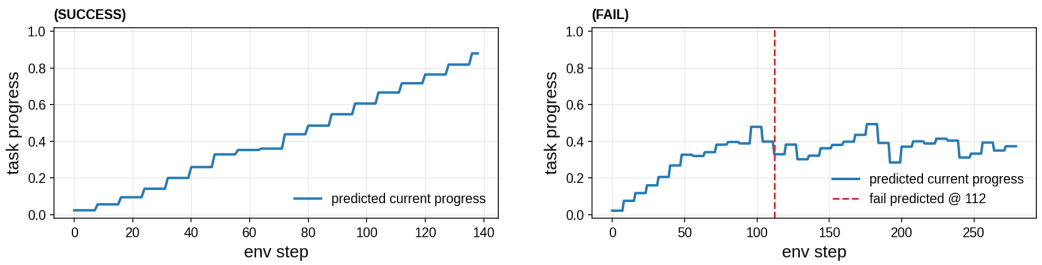

Fig. 8: Recorded current-progress predictions for a successful rollout (left) and a failed rollout (right) of task: *pick up the ketchup and place it in the basket*. The red dashed line marks the PS alarm at environment step 112 in the failed rollout; the successful rollout has no alarm. Horizontal axes show environment steps and reflect each rollout's recorded duration.

B.6.3 Joint Progress Learning and Policy Success

We compare FM and DDPM with progress-loss weight 0.1 against separately trained headless counterparts. Recovery is disabled, and training steps match the failure-detection experiments. Each

configuration is evaluated using the same three seeds, with 50 trials per task on all ten tasks in each of the four suites, totaling $24{,}000$ rollouts across both backbones and head settings. We report the mean over seeds and weight the four suites equally. Joint progress learning changes aggregate success from 83.18% to 84.00% for FM and from 75.05% to 78.20% for DDPM (Tab. 15(b)).

The progress loss updates the shared visual encoder as well as the head. A possible explanation for the aggregate improvement is that this auxiliary supervision encourages features that distinguish stages of task execution and also support action conditioning.

B.7 On-Demand Recovery Details

Detector choice and evaluation. We consider deployment settings where collecting successful rollouts for task-specific alarm calibration is impractical. Fixed-threshold PS directly uses recent progress predictions without requiring these additional rollouts.

Evaluation protocol. Within each backbone and suite, all recovery methods share the frozen policy and PS alarm controller. The evaluated intervention methods activate only after an alarm.

We first run the policy with recovery disabled and record which episodes trigger PS: 365 episodes for FM and 463 for DDPM. Within each backbone, every recovery method is evaluated on these same episodes. RR measures completion among these episodes, including those that would succeed without recovery. It therefore differs from the fraction of episodes changed from failure to success by an intervention. SR measures completion over all episodes. For either metric, we pool episodes within each suite and then average the suite rates equally. We additionally report pooled micro rates. Macro and micro SR coincide because each suite contains 500 episodes, whereas RR denominators differ across suites. Fig. 4(b) shows six of seven evaluated recovery methods; Tabs. 16 and 17 report all methods, paired counts, and the no-intervention reference.

Recovery methods. We compare Plain recovery with Stochasticity Injection (SI), Source-Noise Scaling (SNS), Higher-NFE Replanning (H-NFE), Best-of-N Replanning (BofN), VLM-Guided Replanning (VLM) (Zhang & Shah, 2026), and FWM Guidance (An et al., 2026).

Plain replanning regenerates actions with the unmodified sampler.

SI adds score-corrected noise to every solver step inside its activation window, reusing the model output already computed for that step, so it requires no additional network evaluations. On FM, sampling time τ increases from noise ($\tau = 0$) to the action ($\tau = 1$), with step size $\Delta\tau = 1/N_{\mathrm{FM}}$. Let $\mathbf{v}_{\mathrm{start}}, \mathbf{v}_{\mathrm{mid}} \in \mathbb{R}^D$ be the policy velocities evaluated at the start and midpoint of the step, respectively. The update to the flattened action state $\mathbf{x} \in \mathbb{R}^D$ is

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\tau\, \mathbf{v}_{\mathrm{mid}} + \Delta\tau\, \frac{g^2}{2}\, \hat{\mathbf{s}} + g\sqrt{\Delta\tau}\, \boldsymbol{\xi}, \qquad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D),$$

with scalar noise amplitude $g = 1.5\,(1-\tau)$ and score estimate $\hat{\mathbf{s}} = -(\mathbf{x} - \tau\mathbf{v}_{\mathrm{start}})/(1-\tau) \in \mathbb{R}^D$ evaluated at the step start ($\tau < 1$). The vector $\boldsymbol{\xi} \in \mathbb{R}^D$ is freshly sampled at each update. On DDPM, let $\ell \in \{0, \ldots, N_{\mathrm{DDPM}} - 1\}$ be the diffusion index, traversed in descending order. The same correction is appended to each ancestral step except the last,

$$\mathbf{x} \leftarrow \mathbf{x} + \frac{1}{N_{\mathrm{DDPM}}}\, \frac{g^2}{2}\, \hat{\mathbf{s}} + g\sqrt{1/N_{\mathrm{DDPM}}}\, \boldsymbol{\xi},$$

with $N_{\mathrm{DDPM}} = 100$ denoising steps, $g = 15\sqrt{1-\bar{\alpha}_\ell}$, and $\hat{\mathbf{s}} = -\hat{\boldsymbol{\epsilon}}_\ell/\sqrt{1-\bar{\alpha}_\ell}$. Here $\mathbf{x}$ on the right is the scheduler's ancestral-step output, $\hat{\boldsymbol{\epsilon}}_\ell \in \mathbb{R}^D$ is the noise predicted from that step's input, and $\bar{\alpha}_\ell = \prod_{r=0}^{\ell}(1-\beta_r)$ is the cumulative signal coefficient for scheduler noise variances $\beta_r \in (0,1)$.

SNS multiplies the initial Gaussian sample of FM by 0.3 and leaves the solver unchanged; this is a low-cost proxy for low-temperature sampling rather than exact tempering. We selected the FM amplitudes by checking offline that SI increased sample variability and SNS reduced it, then held them fixed throughout the recovery evaluation.

BofN samples 16 candidate chunks from the same observation and executes their medoid, i.e., the candidate with the smallest summed Euclidean distance to the others in the policy's normalized action space, computed over all action dimensions of the executed chunk, including the gripper.

Tab. 16: Failure-triggered recovery over all tasks. (a) RR averages all suites equally (365/463 triggered episodes for FM/DDPM). Bold and underline mark the best and second-best method per column; dashes denote untested conditions. (b) Episode-by-episode comparison of FWM with Plain: Rescued counts failures changed to successes, Lost counts successes changed to failures, and Tasks lists tasks with more, fewer, or the same completed episodes. FM uses 10/100 solver steps for Plain/FWM; DDPM uses 100 for both. Tab. 17 also reports SR and the no-intervention reference.

(a) Suite-macro RR

Recovery method	FM RR (%) ↑	DDPM RR (%) ↑
Plain Replanning	56.1	57.3
SI	<u>56.3</u>	55.7
SNS	55.2	—
H-NFE	55.2	—
BofN ($N = 16$)	54.0	<u>57.6</u>
VLM ($N = 16$)	46.5	47.1
FWM	**62.9**	**62.3**

(b) Changes in episode outcomes: FWM versus Plain

Policy	Reference	Rescued	Lost	Net	Tasks
FM	Plain Replanning	37	11	+26	19/3/8
DDPM	Plain Replanning	42	18	+24	16/5/9

VLM queries Gemini 3.1 Flash-Lite with the current camera image, the task description, the end-effector position in the image, and object hints; the model returns a two-dimensional waypoint plan and never receives the candidate trajectories. VLM also samples 16 candidates per selection. It keeps the 8 candidates with the smallest summed distance to the others, measured as for BofN, so the BofN medoid is always kept, and ranks the kept candidates locally by projected endpoint distance to two-dimensional VLM waypoints.

FWM refines a single action chunk using world-model feedback guidance inside the sampler after an alarm, with the world model frozen (Appx. A.2). We report solver budgets in integration or denoising steps. On FM, Plain uses 10 solver steps, whereas FWM and H-NFE use 100 for every policy inference after the first alarm; H-NFE applies no guidance. On DDPM, both Plain and FWM use the native 100-step solver. H-NFE provides the FM comparison at the same increased solver budget: FWM achieves 62.9% RR and 90.7% SR versus 55.2% RR and 88.8% SR for H-NFE. For DDPM, H-NFE coincides with the native 100-step solver budget, while SNS is omitted after a diagnostic showed negligible action changes from source-noise scaling under the full ancestral chain.

The progress monitor and recovery protocol are shared across FM and DDPM policies, while sampling and guidance updates are adapted to each generative process.

Overall results and changes in episode outcomes. FWM achieves the highest suite-macro RR and SR among the evaluated methods on both backbones (Tabs. 16 and 17). Relative to Plain, FM RR increases from 56.1% to 62.9% and SR from 89.0% to 90.7%; DDPM RR increases from 57.3% to 62.3% and SR from 86.1% to 87.7%. The respective RR gains are 6.8/5.1 pp and SR gains are 1.7/1.6 pp, computed before rounding. Suite-macro RR weights each suite equally. FM and DDPM have different sets of episodes that trigger an alarm.

Normal execution without replanning. The no-intervention comparison runs the same policy checkpoint with the same initial conditions and alarm rule, but leaves recovery disabled. Without replanning, suite-macro SR over all episodes is 86.5%/82.8% on FM/DDPM. FWM raises RR to 62.9%/62.3% ($+16.8$/$+16.3$ pp) and SR to 90.7%/87.7% ($+4.2$/$+4.9$ pp), yielding 63/74 additional completed episodes. Plain also improves on normal execution: its RR gains are 9.9/11.2 pp and SR gains are 2.5/3.3 pp, proving the value of failure-triggred on-demand recovery offered by DoOver.

All evaluated methods exceed the no-intervention reference in aggregate RR and SR on these tasks, with FWM achieving the largest gains.

Tab. 17: Recovery with and without intervention. (a) RR uses the fixed pre-intervention trigger sets (FM: 365; DDPM: 463). (b) SR uses all episodes per backbone, including unchanged non-triggered outcomes. All methods use the same policy checkpoint and episode initial conditions. Macro averages suite rates equally; micro pools episodes. Their SR values coincide because every suite has 500 episodes.

(a) RR on episodes with alarms (%)

Method	FM macro	FM micro	DDPM macro	DDPM micro
No intervention	46.2	44.7	46.0	44.5
Plain Replanning	56.1	54.8	57.3	55.3
SI	56.3	55.3	55.7	54.0
SNS	55.2	53.7	—	—
H-NFE	55.2	54.0	—	—
BofN ($N = 16$)	54.0	52.6	57.6	55.7
VLM ($N = 16$)	46.5	46.3	47.1	46.0
FWM	62.9	61.9	62.3	60.5

(b) SR over all episodes (%)

Method	FM macro	FM micro	DDPM macro	DDPM micro
No intervention	86.5	86.5	82.8	82.8
Plain Replanning	89.0	89.0	86.1	86.1
SI	89.1	89.1	85.7	85.7
SNS	88.7	88.7	—	—
H-NFE	88.8	88.8	—	—
BofN ($N = 16$)	88.5	88.5	86.3	86.3
VLM ($N = 16$)	86.9	86.9	83.3	83.3
FWM	90.7	90.7	87.7	87.7

Tab. 18: Recovery after true and false alarms, pooling tasks within each backbone. True alarms occur in episodes that fail without recovery; false alarms occur in episodes that succeed without recovery. Rescued counts failures changed to successes; Damaged counts successes changed to failures. Their difference is the net increase in completed episodes.

Policy	Method	Rescued	Rate (%)	Damaged	Rate (%)
FM	Plain	79/202	39.1	42/163	25.8
FM	FWM	91/202	45.0	28/163	17.2
DDPM	Plain	117/257	45.5	67/206	32.5
DDPM	FWM	132/257	51.4	58/206	28.2

Tab. 19: Failure-triggered versus always-on FWM. SR averages all suite rates equally. Per-episode costs are averaged over all episodes. We estimate each episode's cost from measured inference times and the number of calls, including observer updates during execution. The costs cover inference only, excluding simulator execution and rendering.

FWM activation	SR (%) ↑	Inference (s/ep.) ↓	FLOPs (T/ep.) ↓
PS-triggered	90.7	3.22	2.23
Always-on	89.1	6.17	3.45

Tab. 20: FM inference component costs. Times are medians per sampling call, measured with exclusive use of the GPU; counted FLOPs cover matrix multiplication and convolution. Observer cost is separate: 0.435 ms per environment step, with 4.82 G FLOPs per observer update every eight steps. These measurements describe individual calls; the cost per episode depends on how often each component is used.

Inference component	Time (ms)	FLOPs (G)
Nominal sampling (10 steps)	25.5	46.41
H-NFE sampling (100 steps)	215.7	123.00
FWM sampling, gate closed (100 steps)	217.6	127.74
FWM sampling, gate open (100 steps)	293.8	136.18

Recovery after true and false alarms. We divide episodes with alarms according to their outcome without recovery: a true alarm occurs in an episode that would fail, and a false alarm in one that would succeed. Relative to Plain, FWM increases the pooled true-alarm rescue rate from 39.1% to 45.0% on FM and from 45.5% to 51.4% on DDPM. It reduces false-alarm damage from 25.8% to 17.2% and from 32.5% to 28.2%, respectively. FWM changes 91 failures to successes and 28 successes to failures on FM. On DDPM, it changes 132 failures to successes and 58 successes to failures, giving the same net gains of 63/74 episodes as the SR comparison.

Sampling and search alternatives. Alternative samplers and candidate selection yield smaller aggregate gains than FWM. SI reaches 56.3% RR and 89.1% SR on FM, slightly above Plain's 56.1% and 89.0%; on DDPM its 55.7% RR and 85.7% SR are below Plain. BofN reaches 54.0%/57.6% RR and 88.5%/86.3% SR on FM/DDPM. The VLM method reaches 46.5%/47.1% RR and 86.9%/83.3% SR. These results evaluate the VERITAS-inspired waypoint method and its consensus filter described above, rather than the original VERITAS implementation.

B.8 On-Demand Computation: Recovery Cost

Failure-triggered versus always-on FWM. We compare enabling FWM after the first alarm with enabling it from the start of the episode. Both schedules use the same frozen policy, world model, expert memory, cost gate, and PS detector. Failure-triggered FWM turns on at the alarm; always-on FWM begins from episode start. Tab. 19 reports SR and cumulative inference cost over all episodes. Failure-triggered FWM achieves higher SR while using 47.8% less inference time in this comparison.

When to allocate recovery computation. Low-cost monitoring determines when the action sampler begins spending additional computation. Before the alarm, FM uses 10 integration steps; after it, FM with failure-triggered FWM uses 100 steps and applies guidance. Tab. 20 reports measured unit costs for these operations. The additional sampling cost is incurred only after an alarm.

Cost measurement. We measure the median time per policy inference call on three reference episodes, with exclusive use of the GPU and excluding CUDA warm-up. Policy sampling includes controller-side inference; observer updates are measured separately. The measurements exclude simulator execution and rendering. Counted FLOPs cover matrix multiplication and convolution. These component measurements depend on the network architecture and input dimensions; episode totals additionally depend on the number of calls to each component. To compare the two FWM activation schedules, we estimate each episode's cumulative inference time by weighting the measured

unit latencies by its policy-call counts, including additional controller calls, and adding observer updates during policy execution. The comparison measures the cost and success of the two schedules.

World-model training and deployment. For each of the suites, the world model adds 40.0 million parameters (160.1 MB of weights), while the expert latent memory occupies 127–152 MB in FP32 on the device. Neither the policy nor the world model is updated during recovery.

B.9 Real-Robot Experiments

We evaluate policy-native detection and PS-triggered recovery on the R1 Lite robot using the Color Match tasks. We describe the data split, detection results, and recovery trials below.

Color Match and data split. An FM-based DoOver policy with its own progress head controls the R1 Lite robot in a scene with two cubes and two trays, one purple and one black of each. Each cube starts beside the opposite-color tray and farther from the same-color tray. *Purple Place* instructs the robot to pick up the purple cube and place it in the purple tray; *Black Place* lets the robot to pick up the black cube and place it in the black tray. Fig. 5(a) shows an execution of each task. Recovery is disabled for detection evaluation. Tab. 21 gives the data allocation. Baseline fitting uses 50 successful rollouts per task. The 52 test rollouts comprise 17 successes and 35 failures. Fitting (FIT), temporal normalization (NORM), threshold calibration (CP), and test sets are mutually disjoint. The task horizon is frozen from FIT and NORM, with no CP or test rollout removed for insufficient length.

Tab. 21: Data split for Color Match. All calibration counts below denote successful rollouts; test counts include both outcomes. Horizons are policy decisions (environment steps).

Task	FIT	NORM	CP	Test success/failure	Horizon
Purple Place	50	11	9	9 / 17	17 (136)
Black Place	50	14	9	8 / 18	20 (160)

Task-calibrated detection. We compare the main detectors, MS-PS and PS, with six baseline detectors that do not require heavy post-policy training time. BA, DT, and FPR are first computed separately for each task and then equally averaged. DT is measured as a fraction of the frozen horizon, with missed detections assigned one. With nine successful CP rollouts per task and nominal level $\alpha = 0.10$, the corrected conformal order statistic is the ninth (largest) calibration score. Tab. 22 reports the resulting detection results. PS and MS-PS achieve 84.6% and 81.7% BA, respectively, compared with 71.5% for the strongest baselines. PS also detects failures earliest, at 62.9% of the task horizon.

Tab. 22: Color Match failure detection with finite-sample-corrected functional bands at $\alpha = 0.10$. Percentages are task-macro averages over Purple Place and Black Place, from one fixed split.

Method	BA ↑	DT ↓	FPR ↓
MS-PS	81.70	73.11	5.56
PS	84.64	62.91	16.67
RND-OE	66.36	65.85	40.97
STAC	51.39	99.58	0.00
k-NN	56.94	91.67	16.67
STAC-Single	65.28	94.31	0.00
PCA-Kmeans	55.02	82.26	34.72
ACE	71.49	70.92	11.11

Threshold generalization and empirical quantiles. Tab. 23 uses raw scores and a scalar threshold: Within-suite pools both colors' CP rollouts, while LOTO uses only the other color. All thresholds are finite; pooled and transferred thresholds are evaluated empirically without a per-task conformal coverage guarantee. MS-PS achieves 80.31% LOTO BA at 5.56% FPR, compared with 74.51% BA for the strongest baseline, ACE. Tab. 24–26 instead retain the task-calibrated scoring procedure and sweep six empirical quantiles. Both main detectors exceed the compared baselines

in BA at each quantile; MS-PS achieves 90.20% BA at empirical $\alpha = 0.10$. As in simulation, this sweep uses uncorrected empirical quantiles, while the task-calibrated comparison above uses finite-sample-corrected bands.

Tab. 23: Color Match threshold generalization with raw scores at $\alpha = 0.10$: BA (FPR) and DT. Superscripts count how many of the two tasks produce alarms on every test rollout or on none of them; all calibrated thresholds are finite. Percentages are task-macro averages over Purple Place and Black Place, from one fixed split.

Method	Within BA (FPR)	LOTO BA (FPR)	Within DT	LOTO DT
MS-PS	78.84 (5.56)	80.31 (5.56)	75.05	73.67
PS	71.41 (0.00)	77.29 (0.00)	89.98	88.08
RND-OE	58.05 (29.17)	59.68 (34.72)	85.28	83.90
STAC	54.17 (0.00)$^{1/2}$	61.11 (0.00)$^{1/2}$	98.89	94.44
k-NN	65.28 (0.00)$^{1/2}$	67.36 (12.50)$^{1/2}$	96.25	89.72
STAC-Single	68.06 (0.00)$^{1/2}$	72.22 (0.00)$^{1/2}$	95.42	89.44
PCA-Kmeans	61.09 (28.47)	61.09 (28.47)	81.98	81.98
ACE	74.51 (11.11)	74.51 (11.11)	86.69	86.52

Tab. 24: Color Match BA over all six empirical calibration quantiles. The scoring procedure and held-out rollouts match the corrected-band experiment; only the quantile rule changes. Percentages are task-macro averages over Purple Place and Black Place, from one fixed split.

Method	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$	$\alpha = 0.05$	$\alpha = 0.075$	$\alpha = 0.10$
MS-PS	83.09	83.09	83.09	84.56	87.34	90.20
PS	84.64	84.64	84.64	84.64	84.64	84.64
RND-OE	66.36	66.36	66.36	66.36	66.36	66.36
STAC	51.39	51.39	51.39	51.39	51.39	51.39
k-NN	56.94	56.94	56.94	56.94	56.94	56.94
STAC-Single	65.28	65.28	65.28	65.28	65.28	66.67
PCA-Kmeans	55.02	55.02	55.02	57.80	59.27	60.74
ACE	72.88	75.74	77.21	80.07	67.91	67.65

Tab. 25: Color Match FPR over all six empirical calibration quantiles. The scoring procedure and held-out rollouts match the corrected-band experiment; only the quantile rule changes. Percentages are task-macro averages over Purple Place and Black Place, from one fixed split.

Method	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$	$\alpha = 0.05$	$\alpha = 0.075$	$\alpha = 0.10$
MS-PS	5.56	5.56	5.56	5.56	5.56	5.56
PS	16.67	16.67	16.67	16.67	16.67	16.67
RND-OE	40.97	40.97	40.97	40.97	40.97	40.97
STAC	0.00	0.00	0.00	0.00	0.00	0.00
k-NN	16.67	16.67	16.67	16.67	16.67	16.67
STAC-Single	0.00	0.00	0.00	0.00	0.00	0.00
PCA-Kmeans	34.72	34.72	34.72	34.72	34.72	34.72
ACE	11.11	11.11	11.11	11.11	35.42	41.67

Tab. 26: Color Match DT over all six empirical calibration quantiles. The scoring procedure and held-out rollouts match the corrected-band experiment; only the quantile rule changes. Percentages are task-macro averages over Purple Place and Black Place, from one fixed split.

Method	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$	$\alpha = 0.05$	$\alpha = 0.075$	$\alpha = 0.10$
MS-PS	70.99	70.68	70.51	68.81	66.38	64.44
PS	62.74	61.63	61.63	61.63	61.63	60.49
RND-OE	65.85	65.85	65.85	65.85	65.85	65.85
STAC	99.58	99.44	99.44	99.31	99.17	99.17
k-NN	91.67	91.67	91.67	91.67	91.67	91.25
STAC-Single	94.31	94.17	94.03	94.03	93.89	93.47
PCA-Kmeans	82.26	82.26	82.26	79.62	78.79	78.62
ACE	68.80	66.69	64.68	59.06	58.61	51.64

On-demand recovery. For each of Purple Place and Black Place, we evaluate ten failure-triggered rollouts per method. FWM completes $3/10$ Purple Place rollouts and $2/10$ Black Place rollouts;

Plain Replanning and BofN each complete $0/10$ on both tasks. Across the two tasks, FWM completes $5/20$ rollouts (25% RR), compared with $0/20$ for each alternative (Fig. 5(b)).

C Extended Related Work and Detection Costs

C.1 Detection Quality and Resource Requirements

A runtime monitor must detect failures accurately while adding little cost to policy execution. DoOver learns its monitoring signal from expert timestamps and visual features already computed for action generation. Its 0.526M-parameter head requires no failed training rollouts, external foundation-model inference, or additional action samples. The policy-native head is jointly trained with the policy, avoiding a separate detector-training stage. A supplementary experiment also adds a new head to a frozen FM policy (Appx. B.6.1). The detection results and cost measurements show where this feature reuse improves BA and AP while reducing training, storage, or sampling costs.

Tab. 27 places FM detection quality beside the audited resource requirements of all 13 baselines. Quality values are the four-suite ID means in Fig. 2(a,b).

Tab. 27: Detection performance and additional computational resources. BA/AP are four-suite FM means over three seeds; individual seed values, FPR, and DT are shown in Fig. 2(a–d). Resource columns use the LIBERO-Goal FM reference audit (Appx. B.5).

	FM: four suites		FM reference costs: LIBERO-Goal			
Method	BA (%) ↑	AP (%) ↑	Extra params (M)	Stored state (MiB)	Post-policy fit (s)	Extra action samples
MS-PS	62.8	55.9	0.526	2.01	0	0
PS	62.4	53.3	0.526	2.01	0	0
logpZO	61.0	46.5	62.071	236.78	8,835.667	0
RND-OE	59.9	46.2	42.486	162.07	6.642	0
STAC	58.3	40.6	0	< 0.001	0	255
k-NN	57.1	41.4	0	57.16	0.036	0
FIPER-AND	56.1	38.6	42.486	162.07	10.203	255
STAC-Single	55.7	35.2	0	< 0.001	0	0
PCA-Kmeans	55.7	34.7	0	0.35	0.363	0
ACE	55.0	36.9	0	< 0.001	0.333	255
Mahalanobis	52.4	29.6	0	128.03	10.676	0
k-NN (S-F)†	62.9	48.8	0	143.84	0.048	0
Mahalanobis (S-F)†	58.2	42.2	0	256.06	20.969	0
SAFE-LSTM†	56.2	36.8	4.459	17.01	9.708	0
SAFE-MLP†	54.8	36.1	1.049	4.00	4.013	0

Resource measurements. In Tab. 27, stored state includes FP32 network weights or reference data and statistics, excluding the shared policy, optimizer, and common calibration layer. PS and MS-PS share one jointly trained head. They require no separate fitting stage, but the head adds computation during joint training. Fit times exclude rollout collection and threshold calibration. Extra samples are per decision, beyond the control sample. †: also trained or fitted on failed rollouts; all methods use successful rollouts for benchmark calibration. A forward pass through the head takes 0.019 ms; generating the 255 extra samples costs 125.30 ms.

Learned models of successful observations. FAIL-Detect's logpZO (Xu et al., 2025) learns a separate flow model over policy embeddings, while RND-OE (Römer et al., 2026) trains a predictor against a frozen random target. These models provide alternative monitoring signals at the cost of additional networks and fitting. logpZO is the strongest compared baseline in FM ID BA and AP: MS-PS improves its mean BA from 61.0% to 62.8% and AP from 46.5% to 55.9%. The progress head improves these metrics while reducing additional model size from 62.071M additional parameters and 236.78 MiB of weights to 0.526M and 2.01 MiB, a reduction by a factor of approximately 118. logpZO also requires 2.45 GPU-hours of post-policy fitting in the reference audit, whereas DoOver's head is optimized jointly with the policy. RND-OE adds 42.486M parameters across its predictor and target, about $81\times$ the head's size; it also requires a separate fitting stage.

Action consistency and sampling uncertainty. STAC (Agia et al., 2024) and ACE (Römer et al., 2026) derive alarms from repeated action generation. Each decision uses a batch of 256 action chunks, of which one supplies the control action. Batched generation adds 125.30 ms relative to an ordinary policy decision, before detector scoring. A separately timed forward pass through DoOver's head takes 0.019 ms; the additional sampling time is approximately 6,600× larger. The comparison uses batched sampling, not 256 serial policy calls. FIPER-AND combines ACE with RND-OE and therefore carries both the sampling requirement and the additional predictor/target networks. DoOver instead obtains its signal from changes in predicted progress without resampling actions. These are component measurements; the complete timing scope is given in Appx. B.5.

Nonparametric monitors and inexpensive alternatives. k-NN and Mahalanobis monitors (Agia et al., 2024; Gu et al., 2026) avoid learning another network, but retain reference embeddings or covariance statistics. Their success-only reference states occupy 57.16 and 128.03 MiB in the audit; DoOver keeps a fixed head and short progress history, without an embedding bank or nearest-neighbor query. PCA-Kmeans (Liu et al., 2024; Römer et al., 2026) compresses its reference state to 0.35 MiB, while STAC-Single (Gu et al., 2026) avoids extra action samples. Both provide low-cost baselines. Against both, MS-PS improves FM ID BA by 7.1 pp, and AP by 21.2 and 20.7 pp, respectively. Against these low-cost baselines, DoOver's advantage is better detection with a small learned head; it does not have the lowest cost in every category.

Failure-supervised monitors. SAFE-MLP, SAFE-LSTM, and the success-minus-failure variants of k-NN and Mahalanobis (Gu et al., 2026) additionally use failed rollouts. SAFE introduces a separately trained classifier; the S-F variants retain embeddings or statistics from failed rollouts. We report all four methods to show how their detection results compare with their data and storage needs: for example, k-NN (S-F) reaches 62.9% FM ID BA with 143.84 MiB of reference embeddings and failure data, while MS-PS reaches 62.8% BA with a 2.01 MiB head and no failed training rollouts; its AP is 55.9% versus 48.8%. Here MS-PS achieves similar BA and higher AP while using less storage and no failed training rollouts. All methods retain successful-rollout calibration under the shared benchmark protocol.

External foundation-model monitoring. Robometer (Liang et al., 2026) uses a pretrained 4B VLM and over one million trajectories, including failures. It uses successful and failed executions to learn reward and progress predictions. Its monitoring signal requires inference through the external VLM. DoOver obtains progress supervision from the action policy's existing demonstrations and learns a lightweight head jointly with that policy, reusing the visual features already computed for control. This combines demonstration reuse, shared computation, and a small additional network to make progress monitoring inexpensive. Our primary experiments compare monitors for compact generative manipulation policies without external foundation-model inference; Appx. C.2 details the distinction from Sentinel's VLM-based progress monitor.

Progress value learning for policy improvement. RISE (Yang et al., 2026) initializes a separate value model from a pretrained VLA, first fitting normalized temporal progress and then combining that objective with temporal-difference learning on successful demonstrations and failed rollouts. The model evaluates imagined outcomes for policy improvement. DoOver learns its monitoring head jointly with action prediction using only the policy's expert demonstrations, and converts changes in the predicted progress into runtime alarms. The technical distinction concerns the training data, model reuse, and computation required to obtain and use the signal.

Detection performance and cost. We report detection means and variation over three seeds together with model size, fitting time, and action-sampling costs. The relevant comparison depends on the baseline: DoOver reduces the resources required by larger or sampling-based monitors, and improves detection over several inexpensive alternatives. The jointly trained head has a small, nonzero training and inference cost: the reference audit estimates about 0.2% marginal training time and measures 0.019 ms per forward pass through the head. Its savings concern policy-native detection; the external encoder in cross-policy monitoring and the world model used for recovery are accounted for separately in Appx. B.4 and B.8.

C.2 Sentinel and Progress Monitoring from Policy Features

The shared failure mode. Sentinel (Agia et al., 2024) explicitly identifies *task progression failures*: a policy can act consistently while failing to accomplish its task. This is an important precedent for monitoring progress. DoOver's contribution concerns obtaining an informative progress signal from policy features and existing demonstrations using a small additional head.

Different monitoring mechanisms. Sentinel combines two complementary detectors. STAC compares distributions of overlapping action predictions across decisions, using repeated action samples to detect erratic behavior. Its progression detector instead queries an external VLM with the observed video, task description, elapsed time, and task time limit. The VLM reasons about the execution and returns a failure classification. Their experiments query this component twice per episode; additional prompt variants use a successful video or goal images. The combined Sentinel monitor raises an alarm when either component does.

DoOver learns a numerical progress curve from normalized expert timestamps through a 0.526M-parameter head attached to the policy's visual features. PS and MS-PS turn the temporal change in this prediction into a failure score at each policy decision. The head learns from the policy's demonstrations, and deployment reuses the encoder already evaluated for control. This requires neither video-language reasoning nor repeated action generation. The policy-native head learns jointly with action prediction. A supplementary experiment adds the monitor to a frozen FM policy by training only a new head (Appx. B.6.1).

Computational cost and evaluated components. DoOver monitors changes in progress using a small network and a short history of predictions. On FM, a forward pass through the head takes 0.019 ms, with no external VLM inference; the timing procedure is described in Appx. B.5. Our numerical Sentinel comparison evaluates STAC under the shared protocol. Sentinel's VLM component requires a different model and computation; our measured detection comparison covers STAC only.

C.3 On-Demand Recovery and Failure-Triggered Guidance

Interactive imitation learning improves robustness through corrective data and policy updates (Ross et al., 2011; Liu et al., 2024), while recovery-oriented methods learn additional recovery behaviors (Dai et al., 2025; Vats et al., 2025). Our recovery benchmark evaluates inference-time interventions with the deployed action policy held fixed.

VERITAS (Zhang & Shah, 2026) samples and verifies candidate action chunks at each decision throughout execution. Its VLM generates a reference trace once at episode start; the recurring cost comes primarily from multiple policy samples, with inexpensive geometric verification. FWM's original inference procedure (An et al., 2026) likewise applies world-model guidance at each policy decision, within the low-noise portion of denoising. We instead use a lightweight failure signal to decide when to enable the additional sampling and guidance.

FWM (An et al., 2026) already performs inference-time guidance with observation feedback. DoOver studies how lightweight monitoring can allocate this more expensive computation to recovery after an alarm. Our benchmark compares recovery methods within ongoing episodes. All methods use the same alarm rule and are evaluated on the same episodes with alarms, identified before any intervention. We also measure inference costs on FM and compare failure-triggered with always-on FWM. The current results place FWM highest in aggregate RR and SR among the evaluated methods on both backbones, with gains in both metrics over Plain and normal execution without replanning.

The central design choice is when to allocate the guided-sampling budget. DoOver preserves nominal action sampling before the alarm and enables high-budget guided sampling for the subsequent recovery phase. Appx. B.8 reports the FM activation-schedule comparison and unit inference costs. The monitor adds little computation; guided sampling is more expensive and starts only after an alarm. World-model training and deployment storage remain additional requirements and are accounted for separately. For FM, we use the standard Gaussian-path score-to-velocity conversion, consistent with the guidance framework of Feng et al. (2025). The probability path, energy, and

numerical update are specified in Appx. A.2. This adaptation enables the same feedback-guided recovery method to be evaluated on both backbones. The benchmark compares recovery methods under a shared protocol and measures the benefit of enabling guidance after a failure alarm.

C.4 Progress-Guided Backtracking in Navigation

Self-Monitoring Agent (Ma et al., 2019a) and Regretful Agent (Ma et al., 2019b) study a different problem: instruction following through discrete viewpoint transitions on a navigation graph. Their actions select navigable viewpoints, and backtracking revisits graph nodes. DoOver studies failure monitoring and recovery during continuous visuomotor manipulation with FM/DDPM policies. Here, action chunks change robot–object configurations; returning the end effector to an earlier pose need not restore the object state, so recovery must replan from the resulting physical configuration.

The supervision and use of progress also differ. Self-Monitoring Agent supervises progress with distance-to-goal targets. The monitor combines visual and recurrent features with attention over the navigation instruction to estimate instruction completion. Regretful Agent builds on this monitor: its Regret Module uses progress differences to learn weights over forward and rollback representations, while its Progress Marker stores estimates at visited viewpoints and incorporates their differences from the current estimate into candidate-direction features. Both methods use progress to choose navigation actions and search the graph.

DoOver uses the temporal position of each observation–action frame in expert manipulation demonstrations as supervision in addition to the target actions. Normalized timestamps $t/(T-1)$ turn this recorded ordering into progress targets without computing distances to a goal or annotating intermediate task completion. A lightweight head learns these targets jointly with action prediction and reuses the policy's visual features. At execution, temporal stagnation in its predictions provides a failure alarm that activates additional replanning computation from the current physical state.