DRFR:Deviation Representation and Failure Risk Learning for Weakly Supervised Continual Monitoring of Robot Task Semantics

Anonymous authors
Paper under double-blind review

Abstract

Robots executing language-conditioned tasks may produce physically plausible actions that still violate the intended semantics. Continuously detecting such deviations and their associated failure risk is essential for reliable autonomous execution. Yet successful executions vary in strategy and timing, and local deviations can often be corrected by subsequent actions. This makes single-reference or instantaneous detection unreliable. We present a weakly supervised framework for continuous semantic failure detection that uses only task-level success/failure labels to learn process-level risk, while dynamically expanding its reference space with accumulated reliable successes. Our framework comprises three components. First, we introduce Progress-Aware Deviation Representation with Success Memory (PADR). PADR retrieves task-relevant successful trajectories using task-conditioned vision-language representations. It then aligns the current execution prefix with multiple candidate successes through open-ended monotonic alignment, producing task-relevant relative deviation signals. Second, we propose Failure Risk Learning from Execution Deviations (FRLED). By leveraging non-negative evidence injection and a learnable recovery gate, FRLED tracks the accumulation and resolution of deviation evidence and estimates failure risk during execution under task-level success/failure supervision. Finally, we introduce Continual Success Memory Expansion via Reliability and Novelty Gating (CSME). CSME accepts, buffers, or rejects candidate experiences according to task outcome, behavioral reliability, and experience novelty. Newly confirmed success patterns are then incorporated into the reference memory. Experimental results demonstrate that the proposed framework can identify semantic task failures, while further improves detection performance on subsequent tasks.

1 Introduction

Vision-language-action policies use visual observations and natural-language instructions to generate robot actions (Zitkovich et al., 2023; Kim et al., 2024). Reliable execution also requires checking whether those actions continue to fulfill the instruction. A robot may grasp an object successfully but choose the wrong one (Duan et al., 2025; Grislain et al., 2025). Such a mistake can look physically plausible while violating the task semantics. Detecting it during execution matters because later actions may either correct the mistake or carry it toward task failure (Agia et al., 2024; Lin et al., 2025). We study a weakly supervised setting in which each training execution has a task-level outcome label but no stage-level failure annotations.

Existing monitors use different signals to identify failures. They measure departures from successful behavior (Xu et al., 2025), assess task progress or policy features (Agia et al., 2024; Gu et al., 2025), and compare observations with language instructions (Duan et al., 2025; Grislain et al., 2025). Recent work also learns failure scores during execution from final task outcomes (Zhang et al., 2026). Our deviation signal compares the observed prefix with multiple retrieved successes aligned at matching task stages. For continuous semantic

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

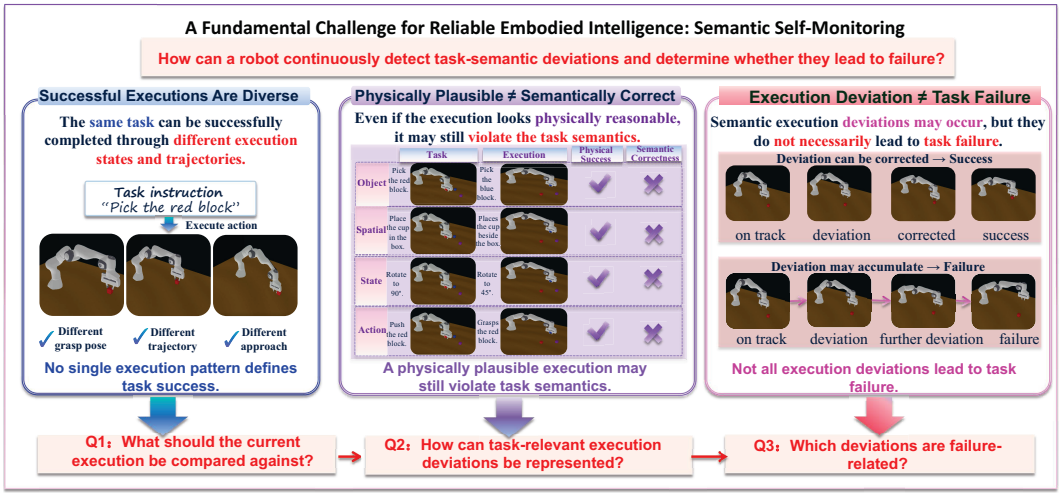

Figure 1: Motivation for continuous task-semantic failure detection.

monitoring, the central challenge is to define a meaningful task-semantic deviation and judge its consequence as the execution unfolds. We therefore ask: How can a robot continuously detect task-semantic deviations and determine whether they lead to failure?

Figure 1 shows three difficulties behind this question. First, the same task admits different successful behaviors, and their stages may progress at different rates (Jia et al., 2024; Memmel et al., 2025). The robot needs an appropriate success reference for its current task and progress (Q1). Second, an action may be feasible yet involve the wrong object, spatial relation, or action sequence (Duan et al., 2025; Grislain et al., 2025). The comparison must capture the resulting semantic deviation (Q2). Third, a local deviation may be corrected by later actions or remain unresolved. For example, a robot may put back the wrong object and grasp the target, or keep manipulating the wrong object. Final task labels alone do not specify which observed deviations contributed to failure (Lin et al., 2025; Zhang et al., 2026). The robot must assess the failure relevance of deviations over time (Q3).

Our key idea is to evaluate an ongoing execution against multiple successful processes at matching progress stages. This provides task-relevant deviation evidence. We then track whether that evidence accumulates or is resolved as new observations arrive. In the example above, continued manipulation adds deviation evidence, while the corrective action can reduce unresolved evidence. The resulting estimate links observed semantic deviations to the risk of final task failure. Newly confirmed successes can expand the reference memory for subsequent executions. The memory starts from outcome-verified successes, and the failure-risk model is initialized on completed executions with task-level outcome labels. A new execution enters trusted memory only after its observed outcome confirms success; the model's reliability estimate alone cannot admit it.

We make three contributions:

- To compare diverse executions at the right task stage, we propose Progress-Aware Deviation Representation with Success Memory (PADR). It retrieves task-relevant successful trajectories and aligns them with the current execution prefix to construct multiple progress-matched references and extract semantic deviations.
- To determine whether observed deviations remain relevant to failure, we propose Failure Risk Learning from Execution Deviations (FRLED). It models evidence injection and recovery to estimate failure risk throughout execution using only task-level outcome labels.
- To keep the success references useful across tasks, we propose Continual Success Memory Expansion via Reliability and Novelty Gating (CSME). It uses task out-

come, behavioral reliability, and experience novelty to accept, buffer, or reject candidate experiences before updating the success memory.

Section 2 reviews related work, Section 3 presents the method, and Section 4 reports the experiments.

2 Related Work

Success references and progress alignment. Successful executions may follow different strategies, as illustrated by the diverse demonstrations in D3IL (Jia et al., 2024). STRAP uses visual features and dynamic time warping to retrieve sub-trajectories for policy adaptation (Memmel et al., 2025). GVL estimates task progress by reasoning about the temporal order of video frames (Ma et al., 2025). RoboProcessBench evaluates whether vision-language models understand execution phase, local progress, and temporal order (Xia et al., 2026). FAIL-Detect learns monitoring signals from successful rollouts (Xu et al., 2025), while PRM-as-a-Judge derives dense task-progress measures from execution videos (Ji et al., 2026). PADR uses successful experience to construct comparison references: it retrieves several task-relevant trajectories and aligns the observed prefix to their corresponding progress stages. The aligned states provide multiple local references for extracting task-semantic deviations.

Semantic and runtime failure monitoring. Vision-language methods compare robot behavior with task instructions. AHA explains manipulation failures (Duan et al., 2025), while I-FailSense detects semantic mismatches with a post-trained VLM (Grislain et al., 2025). Thoduka et al. use action and object cues for video-based detection (Thoduka et al., 2025); Guardian reasons about planning and execution errors (Pacaud et al., 2025); and FailBench evaluates VLM judgments across datasets (Navasardyan et al., 2026). Runtime monitors assess action consistency and progress (Sentinel) (Agia et al., 2024) or policy features (SAFE) (Gu et al., 2025). PADR instead measures deviations from retrieved, progress-aligned successes.

Outcome supervision and alarm calibration. Foresight and FRLED both predict online failure from task-level outcomes. Foresight uses action-conditioned world-model latents, whereas FRLED learns risk from progress-aligned success deviations and models their accumulation and recovery (Zhang et al., 2026). Given compute and time constraints, we do not report a matched numerical comparison; this remains future work. ARMADA uses failure detection to request intervention and collect adaptation data (Yu et al., 2025). FAIL-Detect, SAFE, and Foresight use conformal prediction to calibrate alarms (Xu et al., 2025; Gu et al., 2025; Zhang et al., 2026); FRLED learns the risk score and does not claim conformal coverage guarantees.

Recovery and continual references. PRM-as-a-Judge 1.5 extends progress assessment with measures of failure proximity and setback recovery (Ji et al., 2026). FailSafe pairs failure cases with executable corrective actions (Lin et al., 2025). RoboFAC trains a multimodal model for failure analysis and corrective reasoning (Ye et al., 2025), while FLARE equips robot policies with retry and reset mechanisms for recovery (Zhao et al., 2026a). Retrieve-then-Steer stores successful deployment segments and retrieves them to guide later action generation (Zhao et al., 2026b). In our framework, recovery informs the interpretation of failure evidence. After task completion, CSME uses the verified outcome, behavioral reliability, and experience novelty to decide which executions can expand the success references used for subsequent detection.

3 Method

Our framework detects task-semantic failure by comparing an ongoing execution with multiple plausible successes and tracking whether the resulting deviations remain unresolved. Figure 2 shows the complete process. PADR constructs success references that match the

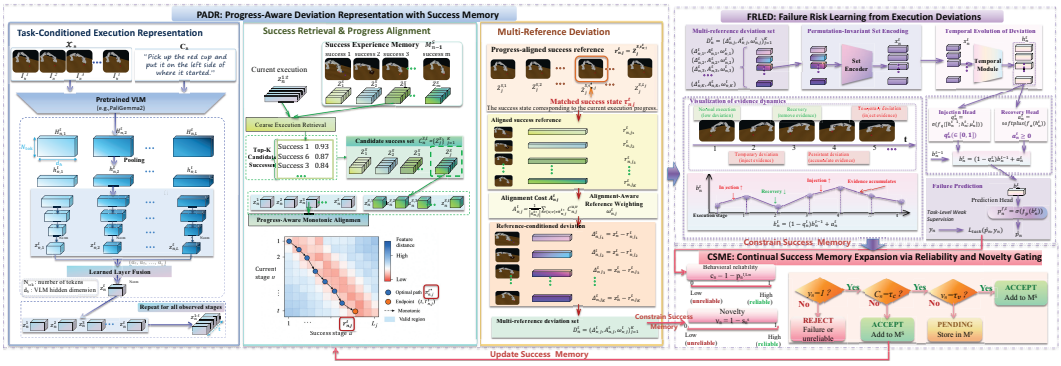

Figure 2: Overview of the framework. The task instruction and observed execution prefix enter PADR, which retrieves and aligns trajectories from the trusted success memory. PADR's multi-reference deviations enter FRLED to produce failure risk at each stage. After the task outcome is known, CSME uses that outcome, behavioral reliability, and experience novelty to update the memory used by future executions.

task and current execution progress. FRLED uses deviations from these references to estimate failure risk throughout execution. After the final task outcome is observed, CSME decides which experiences can improve the success memory for subsequent executions.

3.1 Problem Formulation

We consider executions arriving sequentially. Execution n consists of a task instruction c_n, visual observations $X_n = \{I_n^t\}_{t=1}^{L_n}$, and a final outcome $y_n \in \{0, 1\}$:

$$T_n = (c_n, X_n, y_n). \tag{1}$$

Here, I_n^t is the observation at stage t, L_n is the total number of stages, and $y_n = 1$ denotes failure. During execution, the outcome and future observations are unavailable. Training provides task-level outcomes but no stage-level labels for deviation onset or recovery.

Let M_{n-1}^S be the trusted success memory available before execution n. At stage t, the detector estimates the risk of final failure from the observed prefix:

$$\hat{p}_n^t = p_\theta\big(y_n = 1 \mid c_n, X_n^{1:t}, M_{n-1}^S\big)\,. \tag{2}$$

The three modules resolve successive parts of this inference: PADR determines what successful processes are comparable and represents their differences from the current execution; FRLED determines how those differences affect failure risk over time; CSME updates the references after the outcome is observed.

3.2 PADR: Progress-Aware Deviation Representation with Success Memory

A fixed success template cannot account for different valid ways to complete the same task. Even two similar executions may reach corresponding states at different times. PADR addresses both variations by retrieving several relevant successes and locating the comparable progress stage in each one. Its architecture is shown in Fig. 3.

Task-conditioned representation and retrieval. Let Φ_{VLM} be a pretrained vision-language encoder and F_{rep} a feature mapping. PADR represents observation I_n^t under instruction c_n as

$$z_n^t = F_{\text{rep}}\big(\Phi_{\text{VLM}}(I_n^t, c_n)\big) \in \mathbb{R}^d. \tag{3}$$

The observed prefix is $Z_n^{1:t} = \{z_n^u\}_{u=1}^t$. For a trusted successful execution j, the memory retains its instruction and complete feature sequence $Z_j^S = \{z_j^{S,v}\}_{v=1}^{L_j}$. Keeping complete sequences preserves both distinct success modes and the order of stages within each mode.

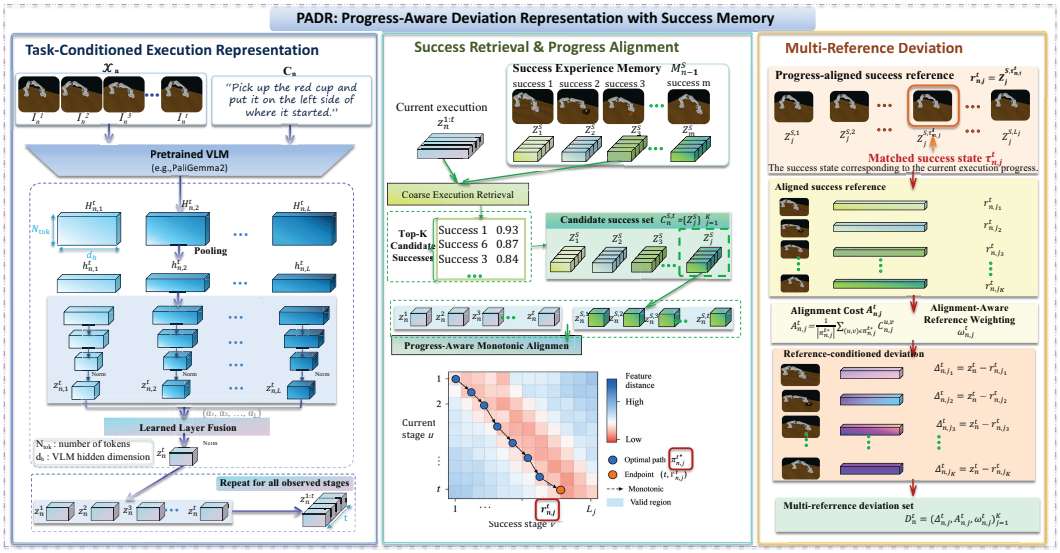

Figure 3: PADR architecture. Task-conditioned visual features encode the observed execution. Retrieval selects candidate successes from memory, and open-end monotonic alignment finds a comparable progress stage in each candidate. The aligned references produce a multi-reference deviation set.

Let $\bar{z}_n^t$ and $\bar{z}_j^S$ be pooled descriptors of the observed prefix and success j. PADR retrieves the K most similar successful trajectories:

$$C_n^{S,t} = \mathrm{TopK}_{j \in M_{n-1}^S} \mathrm{sim}(\bar{z}_n^t, \bar{z}_j^S). \tag{4}$$

Here, $C_n^{S,t}$ is the candidate success set, and sim measures similarity between instruction-conditioned execution features. Retrieval selects plausible processes; it does not yet assume that their current stages are synchronized.

Progress alignment and deviation. For each candidate $j \in C_n^{S,t}$, PADR aligns the observed prefix with the ordered success trajectory:

$$\pi_{n,j}^{t*} = \arg\min_{\pi \in \mathcal{P}_{n,j}^t} \sum_{(u,v)\in\pi} d_{\mathrm{feat}}(z_n^u, z_j^{S,v}). \tag{5}$$

The set $\mathcal{P}_{n,j}^t$ contains monotonic paths, and d_{feat} is the feature distance between matched stages. The path reaches the current observation t but may end at any stage of success j. This open-end alignment compares an ongoing execution with the appropriate part of a completed success. It follows the monotonic matching principle of dynamic time warping.

Let $\tau_{n,j}^t$ be the success-stage index at which $\pi_{n,j}^{t*}$ ends. The aligned reference and its feature residual are

$$r_{n,j}^t = z_j^{S,\tau_{n,j}^t}, \qquad \Delta_{n,j}^t = z_n^t - r_{n,j}^t. \tag{6}$$

Let $A_{n,j}^t$ be the normalized alignment cost and $\omega_{n,j}^t$ its non-negative reference weight, where $\sum_{j\in C_n^{S,t}} \omega_{n,j}^t = 1$. PADR outputs

$$D_n^t = \left\{(\Delta_{n,j}^t, A_{n,j}^t, \omega_{n,j}^t)\right\}_{j\in C_n^{S,t}}. \tag{7}$$

Thus, D_n^t retains the deviation from each plausible success alongside the quality of that comparison. It provides FRLED with richer evidence than a single instantaneous deviation score.

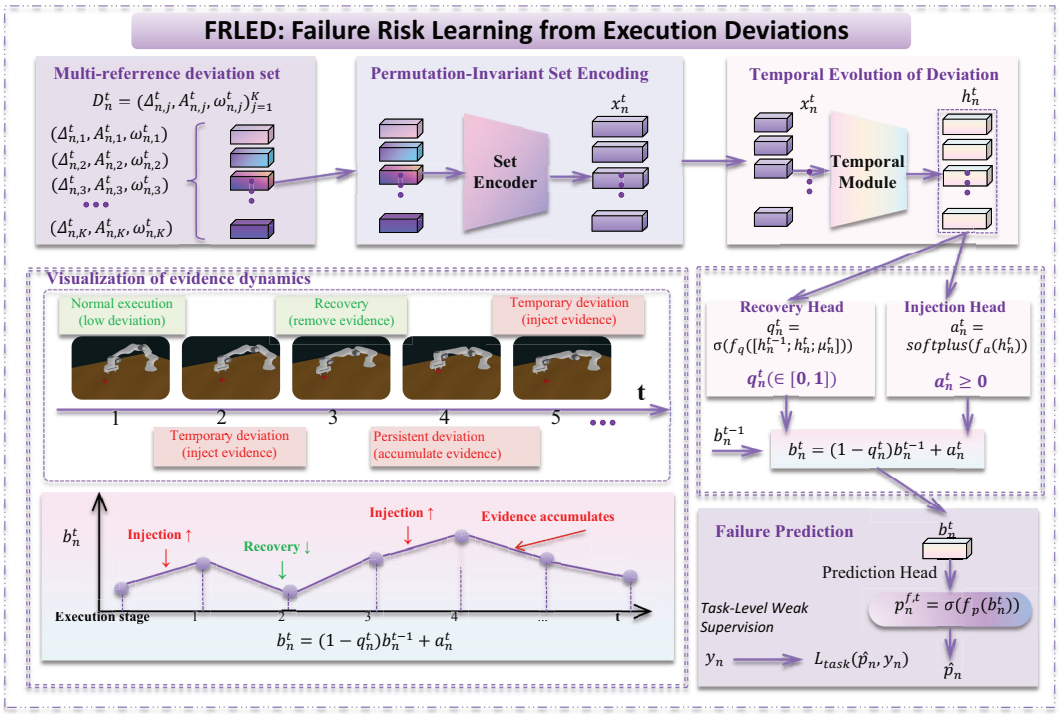

Figure 4: FRLED architecture. A permutation-invariant encoder summarizes the alternative deviations supplied by PADR. A causal temporal model predicts non-negative evidence injection and a recovery gate. Their recurrence maintains unresolved evidence for stage-wise failure prediction.

3.3 FRLED: Failure Risk Learning from Execution Deviations

An observed deviation need not determine the final outcome: later actions may correct it, while a persistent mismatch may increasingly support failure. FRLED therefore maintains an unresolved evidence state rather than classifying each deviation in isolation. Figure 4 shows its set encoding, temporal reasoning, and evidence update.

The elements of D_n^t correspond to alternative success references and have no intrinsic order. FRLED first encodes them with a permutation-invariant set encoder and then summarizes their observed temporal history:

$$x_n^t = \mathrm{SetEnc}(D_n^t), \qquad h_n^t = F_{\mathrm{temp}}(x_n^{1:t}). \tag{8}$$

Here, x_n^t is the stage-level deviation feature, while h_n^t depends only on stages observed through t.

Let μ_n^t summarize the current deviation set. FRLED predicts newly supported evidence a_n^t and the fraction q_n^t of previous evidence to resolve:

$$a_n^t = \mathrm{softplus}(f_a(h_n^t)), \qquad q_n^t = \sigma\big(f_q([h_n^{t-1}; h_n^t; \mu_n^t])\big). \tag{9}$$

The softplus function ensures $a_n^t \geq 0$, and the sigmoid bounds q_n^t in $[0, 1]$. With $b_n^0 = 0$ and $h_n^0 = 0$, FRLED updates unresolved evidence as

$$b_n^t = (1 - q_n^t) b_n^{t-1} + a_n^t. \tag{10}$$

The first term carries forward earlier evidence that remains active; the second adds evidence supported by the current execution. A subsequently corrected mismatch can therefore lose influence, whereas continuing deviations keep supplying evidence.

A prediction head maps the unresolved state to the failure risk available at stage t:

$$p_n^{f,t} = \sigma(f_p(b_n^t)). \tag{11}$$

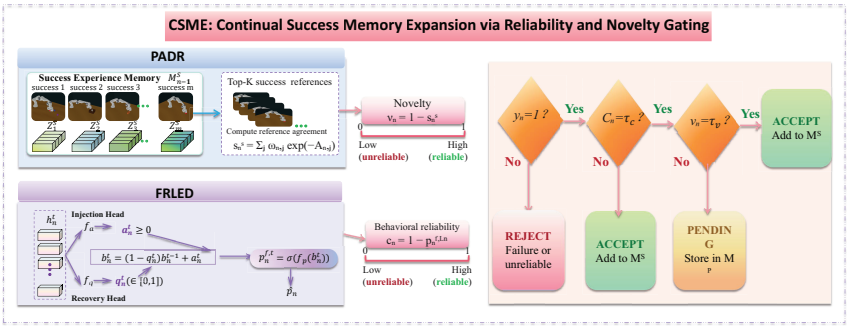

Figure 5: CSME architecture. The final task outcome, FRLED-based behavioral reliability, and PADR-based experience novelty enter an admission gate. Accepted successes join trusted memory, unfamiliar candidates are buffered, and rejected executions cannot become future success references.

We supervise its final-stage prediction $\hat{p}_n = p_n^{f,L_n}$ using the task outcome:

$$\mathcal{L}_{\text{task}} = -y_n \log \hat{p}_n - (1 - y_n) \log(1 - \hat{p}_n). \tag{12}$$

This objective learns evidence injection and recovery through final outcomes, while $p_n^{f,t}$ remains available throughout execution. The stage-wise injection and recovery values also expose how the model's evidence changes over the observed process.

3.4 CSME: Continual Success Memory Expansion via Reliability and Novelty Gating

PADR needs references that reflect both established and newly observed ways of completing a task. CSME evaluates candidate experiences after execution using two signals: FRLED's assessment of unresolved evidence and PADR's assessment of similarity to existing successes. As shown in Fig. 5, these signals complement the observed task outcome in the memory admission decision.

For the completed execution $Z_n = Z_n^{1:L_n}$, CSME defines behavioral reliability as

$$\rho_n = 1 - p_n^{f,L_n}. \tag{13}$$

Let $\nu_n \in [0, 1]$ be experience novelty computed from PADR's progress-aligned agreement with the trusted memory; larger values indicate less represented success patterns. Given thresholds τ_ρ and τ_ν, CSME assigns the completed execution to one of three states:

$$g_n = \begin{cases} \text{Accept}, & y_n = 0,\ \rho_n \geq \tau_\rho,\ \nu_n < \tau_\nu, \\ \text{Buffer}, & y_n = 0,\ \rho_n \geq \tau_\rho,\ \nu_n \geq \tau_\nu, \\ \text{Reject}, & \text{otherwise}. \end{cases} \tag{14}$$

Reliable successes consistent with existing experience enter the trusted memory. Reliable but unfamiliar successes are buffered for corroboration by later successful executions. Failed or unreliable executions are excluded from the success references.

PADR uses only M_{n-1}^S while execution n is in progress. CSME updates the memory after y_n becomes available, so newly admitted experiences can influence execution $n+1$ but cannot serve as their own references.

4 Experiments

4.1 Experimental setup

We evaluate task-semantic failure detection on a CALVIN-based semantic-misalignment dataset. A successful sample pairs a robot trajectory with its matching instruction. A failure sample pairs a valid trajectory with a mismatched instruction. We use two camera

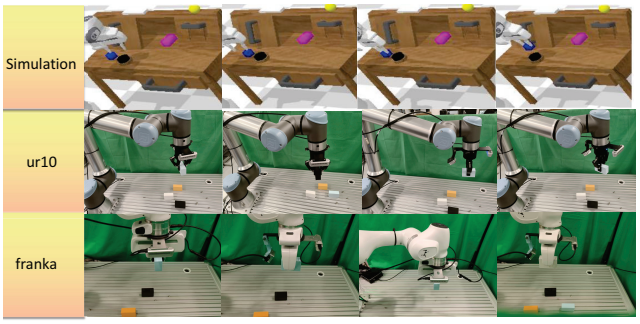

Figure 6: Representative operation scenes in simulation and on the UR10 and Franka robots. Each row shows four frames from one scene.

Table 1: Two-view semantic-misalignment detection on CALVIN. Published results provide reference accuracies; our result uses our validation split. All values are percentages.

Method	Accuracy ↑
PaliGemma2-mix-3B (zero-shot)	52.71
Qwen2.5-VL-7B (zero-shot)	69.46
I-FailSense (published)	91.18
PADR + FRLED (ours)	93.57

views and sample four stages from each trajectory. The training split contains 5,124 samples per class. Its successful trajectories form the initial reference memory. The validation split contains 1,011 samples per class and is excluded from memory construction. Representative simulation scenes and real-robot scenes captured with UR10 and Franka are shown in Fig. 6.

We also evaluate transfer to AHA/RLBench, SMF-DROID, and a real-robot dataset. The CALVIN-trained model is evaluated on each set without target-domain adaptation.

We adapt a PaliGemma 2 3B backbone with LoRA for 20 epochs (6,420 optimization steps), then freeze it for downstream training. Features from layers 13, 17, and 25 yield a $3\times4\times2304$ representation per trajectory. PADR retrieves up to $K = 20$ successful references, and the FRLED hidden dimension is 256. We train the downstream model for 15 epochs and select checkpoints by validation AUROC. Failure is the positive class. We report AUROC and AUPRC from continuous scores; accuracy, failure precision, recall, and F1 use a threshold of 0.5.

4.2 Semantic failure detection

Table 1 places our two-view result alongside published results on CALVIN semantic-misalignment detection. The published rows are from Grislain et al. (2025); our row is measured on the validation split described above. PADR + FRLED achieves 93.57% accuracy, 95.23% AUROC, and 94.67% AUPRC. Its failure precision, recall, and F1 are 94.10%, 92.98%, and 93.53%, respectively. It detects 940 of the 1,011 failure samples, while incorrectly flagging 59 of the 1,011 successful samples.

4.3 Cross-domain transfer

Table 2 reports results without retraining or updating the success memory on the target datasets. AUROC remains between 82.71% and 83.48% across the three evaluation sets.

4.4 Component analysis

We ablate task-conditioned retrieval, progress alignment, the number of successful references, and explicit recovery. All variants use the same two-view features and validation

Table 2: Zero-shot transfer from CALVIN. All values are percentages; recall refers to the failure class.

Target dataset	AUROC ↑	F1 ↑	Recall ↑
AHA/RLBench	82.71	81.53	81.20
SMF-DROID	81.48	82.86	81.91
Real robot	77.15	76.52	77.67

Table 3: Ablations on the CALVIN validation split. All values are percentages.

Variant	AUROC ↑	AUPRC ↑	F1 ↑	Recall ↑
Full model ($K = 20$)	95.23	94.67	93.53	92.98
No task-conditioned retrieval	94.81	94.12	92.15	91.34
Fixed-stage matching	94.56	93.98	91.78	93.12
Single reference ($K = 1$)	93.42	92.76	89.42	87.65
No explicit recovery	94.28	93.85	91.23	92.15

split. The backbone remains frozen, and the downstream heads are trained from the same initialization.

Replacing progress alignment with fixed-stage matching reduces failure F1 by 1.75 percentage points. Using one reference reduces it by 4.11 points, while removing explicit recovery reduces it by 2.30 points. These results support progress-matched multi-reference comparison and recovery modeling for final-outcome classification. As K increases from 1 to 5, 10, 15, and 20, failure F1 rises from 89.42% to 91.56%, 92.67%, 93.12%, and 93.53%. We therefore use $K = 20$, which gives the highest validation F1 among the evaluated settings. We do not evaluate a frame-wise zero-shot VLM or a temporal classifier without success-reference alignment. The former lacksexecution history; the latter lacks progress-matched success references. These are design-based observations, not empirical comparisons.

CSME maintains reliable success references while admitting novel valid strategies. Outcome labels exclude failures, reliability filters uncertain executions, and novelty buffers underrepresented successes for confirmation. Its contribution remains to be quantified through a fixed-versus-updated memory comparison.

PADR's inference cost grows with the number and length of retrieved references. Caching alignment states trades memory for lower latency; limiting K and reference length, or clustering the success memory, can further reduce deployment cost.

FRLED outputs a continuous failure-risk score. Conformal prediction could calibrate a separate alarm threshold, but we do not use it here and make no coverage guarantee. CSME's thresholds τ_ρ and τ_ν govern memory admission, not failure alarms. We keep them fixed during evaluation but do not assess their sensitivity; a threshold sweep remains future work.

5 Conclusion

We investigated how a robot can continuously identify task-semantic deviations and assess their failure risk during execution. To learn from task-level outcomes, we proposed a weakly supervised framework that compares the observed process with multiple successful executions at matching progress stages. It then tracks whether deviation evidence accumulates or is resolved, and screens verified successes before adding them to the reference memory.Experiments show improved semantic failure detection on the evaluated manipulation tasks and useful detection performance across different environments. Ablations further show that multiple aligned references and explicit recovery modeling both contribute to the result. These findings support the central idea of the paper: failure risk is better judged from how a task-relevant deviation evolves than from a single observation of that deviation. As new successful executions are confirmed, the reference memory can represent additional valid ways to complete a task and support subsequent monitoring.

Author Contributions

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

References

Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In CoRL, volume 270 of Proceedings of Machine Learning Research, pp. 689–723. PMLR, 2024.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A Vision-Language-Model for detecting and reasoning over failures in robotic manipulation. In International Conference on Learning Representations. OpenReview.net, 2025.

Clemence Grislain, Hamed Rahimi, Olivier Sigaud, and Mohamed. Chetouani. I-FailSense: Towards general robotic failure detection with vision-language models. CoRR, abs/2509.16072, 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 NeurIPS, 2025.

Yuheng Ji, Yuyang Liu, Huajie Tan, Xuchuan Huang, Fanding Huang, Yijie Xu, Cheng Chi, Yuting Zhao, Huaihai Lyu, Peterson Co, Mingyu Cao, Qiongyu Zhang, Zhe Li, Enshen Zhou, Pengwei Wang, Zhongyuan Wang, Shanghang Zhang, and Xiaolong Zheng. PRM-as-a-Judge: A dense evaluation paradigm for fine-grained robotic auditing. CoRR, abs/2603.21669, 2026.

Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In ICLR. OpenReview.net, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model., 2024.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. FailSafe: Reasoning and recovery from failures in Vision-Language-Action models., 2025.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are In-Context value learners. In ICLR. OpenReview.net, 2025.

Marius Memmel, Jacob Berg, Bingqing Chen, Abhishek Gupta, and Jonathan Francis. STRAP: Robot sub-trajectory retrieval for augmented policy learning. In ICLR. OpenReview.net, 2025.

Zaruhi Navasardyan, Tatul Danielyan, and Hrant Davtyan. Failbench: How reliable are vlms at judging robot task success?, 2026.

Paul Pacaud, Ricardo Garcia, Shizhe Chen, and Cordelia Schmid. Guardian: Detecting robotic planning and execution errors with vision-language models, 2025.

Santosh Thoduka, Sebastian Houben, Juergen Gall, and Paul G. Plöger. Enhancing video-based robot failure detection using task knowledge. In 2025 European Conference on Mobile Robots (ECMR), 2025. doi: 10.1109/ECMR65884.2025.11162998.

Dayu Xia, Yue Shi, Yao Mu, Huiting Ji, Chaofan Ma, Yingjie Zhou, Hua Chen, Yang Liu, Jiezhang Cao, and Guangtao Zhai. RoboProcessBench: Benchmarking process-aware understanding in vision-language robotic manipulation, 2026.

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Tree 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. CoRR, abs/2503.08558, 2025.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. Robo-FAC: A comprehensive framework for robotic failure analysis and correction., 2025.

Wenye Yu, Jun Lv, Zixi Ying, Yang Jin, Chuan Wen, and Cewu Lu. Armada: Autonomous online failure detection and human shared control empower scalable real-world deployment and adaptation, 2025.

Haoran Zhang, Yifu Lu, Boyang Wang, Xuhui Kang, Yen-Ling Kuo, Zezhou Cheng, Mengdi Wang, and Odest Chadwicke Jenkins Jenkins. Foresight: Failure detection for long-horizon robotic manipulation with action-conditioned world model latents, 2026.

Ganlong Zhao, Zijia Tang, Xingping Chen, Zhanghui Kuang, Ye Tian, and Guanbin Li. Flare: A failure-aware framework for autonomous correction and recovery in visual-language robotic manipulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22391–22401, 2026a.

Jianchao Zhao, Huoren Yang, Yusong Hu, Yuyang Gao, Qiguan Ou, Cong Wan, SongLin Dong, Zhiheng Ma, and Yihong Gong. Retrieve-then-steer: Online success memory for test-time adaptation of generative vlas, 2026b.

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

A Appendix

A Implementation Details

This appendix specifies the representation, alignment, evidence update, and memory admission operations used in the main text. All online predictions use only the task instruction, the observed execution prefix, and success memory available before the current execution.

A.1 Task-Conditioned Representation

PADR represents each observation in the context of its task instruction. Let $\mathcal{S}$ denote the selected layers of the vision-language model. For observation I_n^t and instruction c_n, the model produces token-level features

$$\{H_{n,\ell}^t\}_{\ell\in\mathcal{S}} = \Phi_{\mathrm{VLM}}(I_n^t, c_n), \qquad H_{n,\ell}^t \in \mathbb{R}^{N_{\mathrm{tok}}\times d_h}. \tag{15}$$

Here, N_{tok} is the number of tokens and d_h is the hidden dimension. We pool the tokens from each selected layer and project them into a shared d-dimensional space:

$$h_{n,\ell}^t = \frac{1}{N_{\text{tok}}} \sum_{r=1}^{N_{\text{tok}}} H_{n,\ell,r}^t, \qquad z_{n,\ell}^t = \text{Norm}\big(g_\ell(h_{n,\ell}^t)\big)\,. \tag{16}$$

The projection g_ℓ is learned. Layer features are fused with learned weights:

$$\alpha_\ell = \frac{\exp(\gamma_\ell)}{\sum_{m\in\mathcal{S}} \exp(\gamma_m)}, \qquad z_n^t = \text{Norm}\left(\sum_{\ell\in\mathcal{S}} \alpha_\ell z_{n,\ell}^t\right). \tag{17}$$

Thus, z_n^t is the task-conditioned stage representation used in Eq. equation 3. The sequence $Z_n^{1:t} = \{z_n^u\}_{u=1}^t$ retains stage order for progress alignment. After adapting the vision-language backbone, we freeze it during downstream detector training and learn the projection and fusion parameters.

A.2 Success Retrieval and Open-End Alignment

Candidate retrieval. For coarse retrieval, we average the observed prefix and each stored success trajectory:

$$\bar{z}_n^t = \text{Norm}\left(\frac{1}{t}\sum_{u=1}^{t} z_n^u\right), \qquad \bar{z}_j^S = \text{Norm}\left(\frac{1}{L_j}\sum_{v=1}^{L_j} z_j^{S,v}\right). \tag{18}$$

The descriptor $\bar{z}_n^t$ contains observed stages only. The descriptor $\bar{z}_j^S$ can use the complete trajectory because success j was confirmed before execution n. We rank stored successes by cosine similarity:

$$s_{n,j}^t = \frac{(\bar{z}_n^t)^\top \bar{z}_j^S}{\|\bar{z}_n^t\|_2 \|\bar{z}_j^S\|_2}, \qquad C_n^{S,t} = \text{TopK}_{j\in M_{n-1}^S}\, s_{n,j}^t. \tag{19}$$

Retrieval selects plausible successful processes. Their individual stages are matched in the next step.

Progress alignment. For candidate success j, we compute a cost between each observed stage $u \le t$ and stored success stage $v \le L_j$:

$$C_{n,j}(u,v) = 1 - \frac{(z_n^u)^\top z_j^{S,v}}{\|z_n^u\|_2 \|z_j^{S,v}\|_2}. \tag{20}$$

Because the features are normalized, $C_{n,j}(u,v) \ge 0$. Let $G_{n,j}(u,v)$ be the minimum cumulative cost of a path ending at (u,v). We use

$$G_{n,j}(u,v) = C_{n,j}(u,v) + \min\left\{G_{n,j}(u-1,v), G_{n,j}(u,v-1), G_{n,j}(u-1,v-1)\right\}. \tag{21}$$

The path begins at $(1,1)$, with $G_{n,j}(1,1) = C_{n,j}(1,1)$; invalid boundary cells have infinite cost. The three transitions allow the executions to advance at different rates while preserving their temporal order.

The current prefix must end at its observed stage t, but its match may end at any stage of stored success j:

$$\tau_{n,j}^t = \arg\min_{1\le v\le L_j} G_{n,j}(t,v). \tag{22}$$

Backtracking from $(t, \tau_{n,j}^t)$ gives the path $\pi_{n,j}^{t*}$ in Eq. equation 5. The endpoint is open on the stored success trajectory; it does not use future observations from the current execution. Temporal monotonicity constrains the order of matched observations, not the physical motion of the robot. A successful strategy that revisits a state can therefore still appear as successive stages of its stored trajectory.

We normalize the path cost by its length:

$$A^t_{n,j} = \frac{1}{|\pi^{t*}_{n,j}|} \sum_{(u,v)\in\pi^{t*}_{n,j}} C_{n,j}(u,v). \tag{23}$$

A smaller $A^t_{n,j}$ indicates a better overall comparison. The aligned reference and its current-stage residual are

$$r^t_{n,j} = z^{S,\tau^t_{n,j}}_j, \qquad \Delta^t_{n,j} = z^t_n - r^t_{n,j}. \tag{24}$$

Multiple references. We assign relative weights within the retrieved set:

$$\omega^t_{n,j} = \frac{\exp(-A^t_{n,j}/\tau_r)}{\sum_{k\in C^{S,t}_n}\exp(-A^t_{n,k}/\tau_r)}, \qquad \tau_r > 0. \tag{25}$$

The weights sum to one. They indicate which retrieved references fit better relative to the other candidates. The absolute cost $A^t_{n,j}$ remains in the deviation set, so uniformly poor matches are not hidden by weight normalization. PADR passes

$$D^t_n = \left\{(\Delta^t_{n,j}, A^t_{n,j}, \omega^t_{n,j})\right\}_{j\in C^{S,t}_n} \tag{26}$$

to FRLED.

For a fixed retrieved candidate, adding one observed stage requires one additional row of Eq. equation 21. This costs $O(L_j)$ time per stage for success j. If retrieval introduces a new candidate at stage t, its alignment can be initialized in $O(tL_j)$ time.

For K retrieved references with lengths $L_1, \ldots, L_K$, recomputing open-end alignment for a prefix of length t costs $O(t\sum_{j=1}^K L_j)$. Reusing dynamic-programming rows reduces the cost per new stage to $O(\sum_{j=1}^K L_j)$; a newly retrieved reference requires $O(tL_j)$ initialization. For references of average length $\bar{L}$, these costs are $O(Kt\bar{L})$ and $O(K\bar{L})$, respectively. Coarse retrieval adds $O(|M^S|d)$ per stage, where $|M^S|$ is the memory size and d is the feature dimension.

A.3 Deviation Encoding and Sequential Evidence

Permutation-invariant set encoding. The elements of D^t_n represent alternative successful processes; their listing order has no meaning. We first encode each element:

$$s^t_{n,j} = \phi_\Delta\left([\Delta^t_{n,j}; \|\Delta^t_{n,j}\|_2; A^t_{n,j}]\right), \tag{27}$$

where ϕ_Δ is a learned projection. The residual gives the direction of the difference, its norm gives its magnitude, and the alignment cost describes the quality of the full process comparison.

We combine reference fit with residual magnitude using

$$\beta^t_{n,j} = \frac{\exp\left(\log\omega^t_{n,j} - \|\Delta^t_{n,j}\|_2/\tau_\beta\right)}{\sum_{k\in C^{S,t}_n}\exp\left(\log\omega^t_{n,k} - \|\Delta^t_{n,k}\|_2/\tau_\beta\right)}, \qquad \tau_\beta > 0. \tag{28}$$

Because Eq. equation 25 uses a softmax, every $\omega^t_{n,j}$ is positive and the logarithm is defined. The weights $\beta^t_{n,j}$ emphasize plausible local comparisons. The alignment cost remains in $s^t_{n,j}$, allowing the encoder to represent cases in which every candidate fits poorly.

The weighted mean and dispersion are

$$\mu^t_n = \sum_{j\in C^{S,t}_n} \beta^t_{n,j} s^t_{n,j}, \tag{29}$$

$$v^t_n = \sum_{j\in C^{S,t}_n} \beta^t_{n,j}(s^t_{n,j} - \mu^t_n) \odot (s^t_{n,j} - \mu^t_n). \tag{30}$$

Here, μ_n^t summarizes the dominant comparison and v_n^t represents disagreement among plausible references. Reference-weight entropy gives another measure of that disagreement:

$$H_n^t = -\sum_{j \in C_n^{S,t}} \omega_{n,j}^t \log \omega_{n,j}^t. \tag{31}$$

We implement the set encoder in Eq. equation 8 as

$$x_n^t = \phi_x([\mu_n^t; v_n^t; H_n^t]). \tag{32}$$

The sums and shared encoders make this representation invariant to permutations of the retrieved references. The temporal order of execution stages is retained separately in $\{x_n^u\}_{u=1}^t$.

Causal evidence update. A causal temporal model computes $h_n^t = F_{\text{temp}}(x_n^{1:t})$. It cannot attend to stages after t. FRLED uses h_n^t to produce the non-negative injection a_n^t and recovery gate q_n^t in Eq. equation 9. The summary μ_n^t in that equation is precisely the weighted mean defined by Eq. equation 29.

The unresolved evidence state follows Eq. equation 10: $b_n^t = (1 - q_n^t) b_n^{t-1} + a_n^t$, with $b_n^0 = 0$. Unrolling this recurrence gives

$$b_n^{L_n} = \sum_{u=1}^{L_n} a_n^u \prod_{v=u+1}^{L_n} (1 - q_n^v). \tag{33}$$

An empty product equals one. Equation equation 33 makes the roles of the two terms explicit: evidence introduced at stage u affects the final state according to how much of it subsequent gates retain. This permits a temporary mismatch to lose influence while persistent deviations continue to contribute evidence.

The prediction at each observed stage is $p_n^{f,t} = \sigma(f_p(b_n^t))$. For training, we set $\hat{p}_n = p_n^{f,L_n}$ and apply the task-level binary cross-entropy in Eq. equation 12, where $y_n = 1$ denotes failure. Gradients from the final outcome reach both the injection and recovery heads through the recurrence. No stage-level onset or recovery label is used. At inference, $p_n^{f,t}$ is computed from the observed prefix.

A.4 Continual Success Memory Admission

The initial trusted memory M_0^S contains executions whose task outcomes confirm success. During execution n, PADR reads only M_{n-1}^S. CSME makes admission decisions after the outcome y_n becomes available.

Reliability and novelty. Behavioral reliability is the complement of FRLED's final failure prediction:

$$\rho_n = 1 - p_n^{f,L_n}. \tag{34}$$

The observed outcome remains a separate admission condition; a large ρ_n alone cannot establish success.

For each reference $j \in C_n^{S,L_n}$, let $A_{n,j}^{L_n}$ and $\omega_{n,j}^{L_n}$ be the alignment cost and weight for the completed execution. CSME computes reference agreement and experience novelty:

$$s_n^S = \sum_{j \in C_n^{S,L_n}} \omega_{n,j}^{L_n} \exp(-A_{n,j}^{L_n}), \qquad \nu_n = 1 - s_n^S. \tag{35}$$

Because the costs are non-negative and the weights sum to one, $\nu_n \in [0, 1]$. A large ν_n indicates that existing trusted successes provide a weak match. It does not by itself indicate failure.

Three-state admission. With reliability threshold τ_ρ and novelty threshold τ_ν, CSME applies

$$g_n = \begin{cases} \text{Accept}, & y_n = 0,\ \rho_n \geq \tau_\rho,\ \nu_n < \tau_\nu, \\ \text{Buffer}, & y_n = 0,\ \rho_n \geq \tau_\rho,\ \nu_n \geq \tau_\nu, \\ \text{Reject}, & \text{otherwise.} \end{cases} \tag{36}$$

This is the rule in Eq. equation 14. An accepted execution joins trusted memory. A buffered execution is stored in pending memory M^P and is not used as a PADR reference. A rejected execution enters neither memory.

Confirmation of unfamiliar successes. Buffered executions have verified successful outcomes, but their strategies are not yet represented by the trusted memory. To check whether an unfamiliar strategy recurs, we compare distinct buffered executions using the same feature distance and monotonic alignment as PADR. Let $\mathrm{sim}_{\mathrm{traj}}(Z_i, Z_j)$ denote their resulting trajectory similarity. For $Z_i \in M^P$, define

$$\Gamma_i = \left\{ Z_j \in M^P : j \neq i,\ \mathrm{sim}_{\mathrm{traj}}(Z_i, Z_j) \geq \tau_P \right\}, \tag{37}$$

where τ_P is the confirmation threshold. The condition $j \neq i$ ensures that an execution cannot confirm itself. When $|\Gamma_i| + 1 \geq N_{\min}$, with $N_{\min} \geq 2$, the supported successful executions can be promoted from pending to trusted memory. Thus, a new success mode is incorporated after repeated outcome-verified observations.

All admission and promotion operations occur after execution n has finished. Its prediction uses M^S_{n-1}; any resulting memory update is available from execution $n+1$ onward. This read–verify–update order prevents the current execution from serving as its own success reference.