# Before the Robot Fails: Trajectory-Level Risk Prediction in Vision-Language-Action Systems

**Anonymous ACL submission**

## Abstract

Robot failures are costly because they do not merely produce wrong outputs; they change the physical world. Vision-language-action systems are therefore usually evaluated after execution, when the failure has already occurred. We ask whether some failures can be detected earlier, while the system is still forming an action trajectory. We model VLA systems as embodied bounded interpreters whose multimodal hidden states and action-policy readouts can expose trajectory-level risk before physical execution. Under this view, failures arise from unstable or ambiguous embodied interpretation over outcome, referent, action, policy, and safety constraints, and can be predicted before action, prevented through intervention, or recovered from after mismatch with the observed world state. In a held-out-instruction VLA harness with 21,389 samples and five failure buckets, a trajectory-level risk score predicts failures with **0.903 AUC** and **0.983 AUPRC**; a learned weighted risk score improves performance to **0.935 AUC** and **0.989 AUPRC**. Prevention can be tuned along a false-block frontier, achieving **0.806 recall** and **0.980 precision** at a 0.10 target false-block rate. Post-action recovery succeeds in **91.6%** of manifold-level failures and reduces mean manifold error by **16.4%**. These results suggest that some robot failures are visible before execution as trajectory-level risk signals, enabling practical failure prediction, prevention, and recovery for VLA systems.

## 1 Introduction

Robot failures are expensive in a way that language-model failures are not. A bad text continuation can usually be discarded; a bad robot action can move an object, damage hardware, violate a safety constraint, or force costly human intervention. This difference matters for vision-language-action (VLA) systems, which increasingly map language, vision, robot state, and action history into executable policies (Brohan et al., 2023; Zitkovich et al., 2023; Driess et al., 2023; Kim et al., 2025; Octo Model Team et al., 2024). As these systems become more general, the central deployment problem is not only whether they can complete tasks, but whether they can recognize when an action is becoming risky before the robot physically commits.

Most evaluations treat failure as an outcome observed after execution. A robot attempts a grasp, moves the wrong object, violates a constraint, or fails to reach the intended postcondition; only then is the trial marked as failed. This after-the-fact view is natural for benchmarking, but it is incomplete for deployment. In real systems, the useful question is often earlier: can the policy detect that its embodied interpretation is unstable, ambiguous, or unsafe before executing the action? If so, the system can clarify, replan, constrain the action space, ask for human help, or switch into a guarded mode rather than discovering the failure through physical contact.

We study this problem through the lens of trajectory-level risk prediction. A VLA model does not directly jump from an instruction to a final action. It processes a multimodal context: language instruction, visual observation, robot state, and previous actions. Transformer-based policies evolve finite hidden-state trajectories before producing action or policy readouts (Vaswani et al., 2017; Brohan et al., 2023; Zitkovich et al., 2023). We model this process as embodied bounded interpretation: the system constructs a finite, capacity-limited interpretation of the task, including the intended outcome, object or referent binding, action assignment, policy support, and safety envelope. Failures arise when one or more of these coordinates becomes ambiguous, unstable, or unsafe.

This view connects recent progress in language-conditioned robotics with safety and reliability concerns. Work such as SayCan grounds language plans in affordances (Ahn et al., 2023); PaLM-E and RT-style models integrate language and embod-

ied perception for robot control (Driess et al., 2023; Brohan et al., 2023; Zitkovich et al., 2023); Open X-Embodiment, Octo, and OpenVLA push toward generalist robot policies trained across diverse embodiments and datasets (Open X-Embodiment Collaboration, 2024; Octo Model Team et al., 2024; Kim et al., 2025). At the same time, safe reinforcement learning, uncertainty estimation, and recovery methods emphasize that autonomous systems need mechanisms for uncertainty, intervention, and correction rather than blind execution (García and Fernández, 2015; Amodei et al., 2016; Achiam et al., 2017; Thananjeyan et al., 2021; Kahn et al., 2017; Lakshminarayanan et al., 2017; Guo et al., 2017). Our contribution is to bring these concerns to the internal trajectory of VLA inference: risk should be detectable not only from external outcomes, but also from pre-action action-policy trajectories.

We propose a trajectory-level risk score that measures referent ambiguity, action ambiguity, weak policy support, unsafe-action mass, and layerwise instability. This score is designed to identify failures before execution, when intervention is still cheap. Prevention then becomes a policy-level operation: block, replan, clarify, constrain, or hand off before the robot changes the world. When execution has already occurred, the same framework supports recovery: compare the observed post-action world state against the intended outcome, classify the mismatch, and select a safe corrective policy. Thus prediction, prevention, and recovery become stages of the same embodied bounded-interpreter safety stack.

We evaluate this view in a held-out-instruction VLA harness with 21,389 samples and five failure buckets. A hand-designed trajectory-level risk score predicts held-out failures with 0.903 AUC and 0.983 AUPRC. A learned weighted risk score improves to 0.935 AUC and 0.989 AUPRC. The prevention threshold can be tuned along a false-block frontier, reaching 0.806 recall and 0.980 precision at a 0.10 target false-block rate. For post-action recovery, a corrective controller succeeds in 91.6% of manifold-level failures and reduces mean manifold error by 16.4%. These results suggest that many robot failures are not only visible after execution; they are partially visible before execution as trajectory-level risk signals.

The goal of this paper is not to claim complete robot safety or real-world deployment readiness. Rather, we show that VLA failures can be treated as trajectory-level phenomena: decomposable into embodied interpretation coordinates, predictable before action, preventable through intervention, and recoverable after world-state mismatch. This shifts the framing from "did the robot fail?" to "when did the failure become visible, and could the system have acted before physical commitment?"

## 2 Robot Failure Before Execution

Robot failure is usually defined at the end of an episode: the robot drops an object, grasps the wrong item, collides, violates a constraint, or fails to reach the intended postcondition. This outcome-level definition is useful for benchmarking, but it hides the moment at which the failure first becomes predictable. For deployed robots, that earlier moment is often the most important one. If the system can detect that its policy is becoming ambiguous or unsafe before execution, the cost of intervention is low. It can pause, ask for clarification, replan, constrain the action set, or request human assistance. After execution, the same error may require physical recovery, reset, or repair.

This distinction is especially important for VLA systems. A VLA policy receives a multimodal context: a language instruction, visual observation, robot state, and action history. The final action is only the endpoint of an internal computation that must resolve what outcome is intended, which object is being referred to, which action should be taken, whether the policy is sufficiently supported, and whether the action remains inside a safety envelope. A failure can therefore arise before the action is emitted. The system may be uncertain about the referent, split between competing actions, weakly committed to a plan, or assigning nontrivial probability mass to unsafe action regions.

Existing work in language-conditioned robotics already suggests that successful execution depends on more than mapping instructions to actions. Affordance-grounded approaches such as SayCan combine language-model planning with action feasibility estimates (Ahn et al., 2023). Embodied multimodal models such as PaLM-E and RT-style policies integrate language, perception, and robot state into action-conditioned representations (Driess et al., 2023; Brohan et al., 2023; Zitkovich et al., 2023). Generalist policies such as Octo and OpenVLA further scale this idea across diverse robot datasets and embodiments (Octo Model Team et al., 2024; Kim et al., 2025; Open X-Embodiment Collaboration, 2024). These systems make the

pre-action phase increasingly important: as policies become more general, the question is not only whether they can act, but whether they can identify when acting is unreliable.

We therefore treat pre-execution failure prediction as a trajectory-level problem. A risky action is not merely an incorrect final output; it is often preceded by unstable embodied interpretation. For example, if an instruction says "pick up the cup next to the bowl," risk may appear as referent ambiguity between two cups. If the task requires moving an object near a fragile item, risk may appear as unsafe-action mass. If the model alternates between incompatible action plans across internal layers, risk may appear as trajectory instability. These signals are available before the robot changes the world.

This framing differs from standard post-hoc failure classification. Post-hoc classification asks why the robot failed after observing the outcome. Pre-execution prediction asks whether the robot should have acted at all. The latter is more valuable for safety and cost reduction because it allows prevention rather than only diagnosis. It also connects naturally to safe reinforcement learning, uncertainty estimation, and recovery methods, which emphasize that autonomous systems should identify uncertain or unsafe conditions before committing to irreversible actions (García and Fernández, 2015; Kahn et al., 2017; Thananjeyan et al., 2021; Lakshminarayanan et al., 2017; Guo et al., 2017).

We use this idea to define a failure-prevention stack. Before execution, the system estimates trajectory-level risk. If risk is low, it executes normally. If risk is high, it can intervene through clarification, replanning, constrained decoding, guarded execution, or handoff. After execution, if the observed world state does not match the intended outcome, the system enters recovery: it classifies the mismatch, selects a safe corrective policy, and verifies whether the intended postcondition has been restored. Thus prediction, prevention, and recovery are not separate problems. They are different points along the same embodied trajectory.

## 3 VLA Systems as Embodied Bounded Interpreters

We model a VLA system as an embodied bounded interpreter. At physical time step $t$, the system receives a multimodal input containing prior dialogue or task context, the current language instruction, the visual/environment observation, robot state or proprioception, and previous action history. We write this context compactly as

$$X_t = (c, u, E_t, s_t, a_{<t}),$$

where $c$ is prior context, $u$ is the instruction, $E_t$ is the environment observation, $s_t$ is robot state, and $a_{<t}$ is the action history.

A transformer-based VLA policy embeds this context into a finite multimodal sequence and evolves hidden states through a finite number of layers:

$$H^{(0)}(X_t), \ldots, H^{(L)}(X_t).$$

Let $g_t^{(\ell)}$ denote the action-query state at layer $\ell$, such as a special action token, pooled policy state, or final instruction-conditioned readout. For a discrete action or motor-primitive space, the layerwise action logits are

$$\eta_t^{(\ell)} = g_t^{(\ell)} W_A,$$

with action distribution

$$\pi_t^{(\ell)} = (\eta_t^{(\ell)}).$$

For continuous control, the same role may be played by policy parameters such as a mean and covariance, or by a deterministic control head. In either case, the system induces an action or policy trajectory:

$$\mathcal{T}_{\mathrm{act},t} = (\pi_t^{(0)}, \ldots, \pi_t^{(L)}).$$

This is the embodied analogue of a token-logit trajectory in a language model.

The interpretation performed by a VLA system is not purely linguistic. It must resolve an embodied task. We define the embodied interpretation state as a tuple over five coordinates:

$$I_{\mathrm{VLA}}(X_t) = (O, R, A, \Pi, \Sigma).$$

Here, $O$ is the intended outcome, $R$ is the referent or object-role binding, $A$ is the action assignment, $\Pi$ is policy or plan support, and $\Sigma$ is the safety envelope. A successful action requires these coordinates to be mutually compatible: the selected policy should pursue the intended outcome, act on the correct referent, instantiate the correct action, and remain inside the safety envelope.

This decomposition turns robot failure into an interpretable object. An outcome failure occurs

when the achieved world state does not match the intended postcondition. A referent failure occurs when the policy acts on the wrong object or object role. An action-assignment failure occurs when the selected motor primitive or action class is incompatible with the task. A policy failure occurs when the plan is weak, unstable, or inconsistent over the trajectory. A safety failure occurs when the policy assigns too much support to unsafe actions. These failures may be observed after execution, but their precursors can appear before execution in the action trajectory.

To expose these precursors, we define trajectory-level probe masses over action or policy regions. Let $\Omega_O, \Omega_R, \Omega_A, \Omega_\Pi, \Omega_\Sigma$ denote action classes or policy regions compatible with the intended outcome, referent, action, plan, and safety envelope. For discrete actions, the support assigned to a region is

$$M_\Omega(\eta) = \sum_{a \in \Omega} (\eta)_a.$$

For continuous policies, the same quantity is the probability mass assigned by the policy distribution to a region of action space. The embodied interpreted state at layer $\ell$ is then

$$m^{(\ell)}_{\mathrm{VLA},t} = \big(M_{\Omega_O}, M_{\Omega_R}, M_{\Omega_A}, M_{\Omega_\Pi}, M_{\Omega_\Sigma}\big),$$

where each mass is computed from the corresponding layerwise action or policy readout. The sequence of these states forms an embodied interpreted trajectory.

This construction is deliberately bounded. It does not assume that the robot has a perfect world model, exact symbolic planner, or complete safety proof. It only assumes that the policy's finite internal trajectory may contain measurable signals about the stability of the embodied interpretation. If referent support is split across objects, action support is diffuse, policy support is weak, unsafe mass is high, or interpreted states change sharply across layers, then the trajectory contains pre-action evidence of risk.

The bounded-interpreter view therefore gives a practical criterion for failure prediction. A VLA system should not be evaluated only by its final action. It should also be evaluated by whether its pre-action trajectory exposes uncertainty, ambiguity, instability, or safety violation before the robot acts. The next section turns this idea into a risk score for prediction and prevention.

## 4 Trajectory-Level Risk Prediction

If VLA failure is often preceded by unstable embodied interpretation, then risk should be measurable before execution. We define risk at the trajectory level rather than only at the final action. The goal is not to prove that every failure is predictable, but to identify cases where the model's own action/policy trajectory already contains warning signals: ambiguous referents, diffuse action support, weak policy commitment, unsafe-action mass, or large layerwise instability.

Let $m^{(\ell)}_{\mathrm{VLA},t}$ be the embodied interpreted state at layer $\ell$. We define a scalar risk score

$$\rho^{(\ell)}_{\mathrm{VLA}} = \lambda_R \rho^{(\ell)}_R + \lambda_A \rho^{(\ell)}_A + \lambda_\Pi \rho^{(\ell)}_\Pi + \lambda_\Sigma \rho^{(\ell)}_\Sigma + \lambda_\Delta \Delta^{(\ell)}.$$

The terms measure different ways in which embodied interpretation can become unreliable. Referent risk $\rho^{(\ell)}_R$ is high when the model assigns similar support to competing objects or object roles. Action risk $\rho^{(\ell)}_A$ is high when no action class dominates. Policy risk $\rho^{(\ell)}_\Pi$ is high when support for the intended plan is weak. Safety risk $\rho^{(\ell)}_\Sigma$ is high when probability mass enters unsafe action regions. Trajectory instability $\Delta^{(\ell)}$ is high when the interpreted state changes sharply across adjacent layers.

For example, for two competing referents $R_1$ and $R_2$, we use

$$\rho^{(\ell)}_R = 1 - \left| M_{\Omega_{R_1}}(\eta^{(\ell)}_t) - M_{\Omega_{R_2}}(\eta^{(\ell)}_t) \right|.$$

This term is large when the policy cannot strongly distinguish between the two referents. Action ambiguity is defined as

$$\rho^{(\ell)}_A = 1 - \max_A M_{\Omega_A}(\eta^{(\ell)}_t),$$

so risk increases when action support is diffuse. Policy weakness and safety risk are defined as

$$\rho^{(\ell)}_\Pi = 1 - M_{\Omega_\Pi}(\eta^{(\ell)}_t), \qquad \rho^{(\ell)}_\Sigma = M_{\Omega_{\Sigma_{\mathrm{bad}}}}(\eta^{(\ell)}_t).$$

Finally, trajectory instability is

$$\Delta^{(\ell)} = \left\| m^{(\ell+1)}_{\mathrm{VLA},t} - m^{(\ell)}_{\mathrm{VLA},t} \right\|_2.$$

A high value of $\rho^{(\ell)}_{\mathrm{VLA}}$ indicates that at least one necessary component of embodied execution is ambiguous, weakly supported, unsafe, or unstable. Given a threshold $\delta$, the system predicts pre-action failure risk when

$$\rho^{(\ell)}_{\mathrm{VLA}} > \delta.$$

The threshold may be chosen for different deployment preferences. A conservative robot may use a low threshold to block more risky actions, while a throughput-oriented system may use a higher threshold to reduce false blocks. This tradeoff is central for real deployment: prevention is valuable only if it catches failures without halting safe actions too aggressively.

Risk prediction enables prevention. If risk is below threshold, the system executes the policy normally. If risk exceeds threshold, the system can intervene before physical commitment. The intervention depends on the source of risk. High referent ambiguity suggests clarification; high action ambiguity suggests replanning or additional perception; high unsafe mass suggests constrained action selection; high trajectory instability suggests guarded execution or human handoff. In all cases, the goal is the same: prevent the robot from discovering the failure by acting in the world.

This framework also supports learned risk weighting. The hand-designed score treats the terms as interpretable components, but the weights $\lambda$ can be tuned on validation data or learned by a lightweight classifier. A learned score preserves the decomposition while allowing the system to discover which trajectory signals are most predictive in a given environment. In our experiments, we evaluate both a hand-designed risk score and a learned weighted version.

The prevention problem is therefore not simply binary classification. It is a deployment tradeoff between failure recall and false blocking. A useful risk score should support a frontier: at stricter thresholds, it should catch more failures while accepting more blocked safe actions; at looser thresholds, it should reduce false blocks while still catching severe failures. We report this frontier explicitly in Section 6.

## 5 Empirical Harness

We evaluate whether VLA failures can be detected before execution in a controlled embodied manipulation harness. The harness is designed to isolate the failure modes predicted by the embodied bounded-interpreter view: outcome failure, referent failure, action-assignment failure, policy failure, and safety-envelope failure. Each example contains a language instruction, an embodied scene state, a robot-state/action context, a target policy label, and failure annotations. Tasks involve object selection, receptacle placement, drawer-like state changes, spatial relations, and safety-sensitive action choices. The setting is not intended to replace real robot deployment; it is a controlled testbed for asking whether pre-action policy trajectories contain usable warning signals.

We use a held-out-instruction split rather than a random example split. This makes the evaluation harder: test examples contain instruction templates not seen during training, so prediction cannot rely only on memorized task strings. The dataset contains 21,389 samples: 13,773 for training, 2,828 for validation, and 4,788 for test. Failures are grouped into five buckets aligned with the interpretation coordinates introduced above. The VLA policy is trained for 8 epochs and reaches a final validation loss of 0.042.

We run three experiments. First, we test whether a trajectory-level risk score predicts failure before execution. Second, we test whether the same score can be thresholded into a prevention policy with controllable false-block rates. Third, we test whether post-action failures can be repaired by a recovery controller after observing world-state mismatch.

## 6 Failure Prediction and Prevention

The first experiment asks whether failure is visible before action execution. For each test example, we compute a trajectory-level risk score from pre-action policy readouts. The hand-designed score combines referent ambiguity, action ambiguity, weak policy support, unsafe-action mass, and trajectory instability. We also evaluate a learned weighted score trained on validation data over the same interpretable components. Both scores are computed before the robot executes the action.

Table 2 shows the main prediction results. The hand-designed score achieves 0.903 AUC and 0.983 AUPRC on held-out test instructions. At the validation-selected threshold, it reaches 0.862 recall and 0.959 precision, with a false-block rate of 0.222. The learned weighted score improves the AUC to 0.935 and AUPRC to 0.989, while increasing precision and reducing the false-block rate.

These results support the main hypothesis: many failures are not only visible after execution, but are already signaled in the pre-action trajectory. The strongest single component is policy support, but the learned weighted score performs best overall. This suggests that VLA failure prediction benefits

| Quantity | Value |
|---|---|
| Total samples | 21,389 |
| Train / validation / test | 13,773 / 2,828 / 4,788 |
| Split type | Held-out instruction |
| Failure buckets | 5 |
| Training epochs | 8 |
| Final validation loss | 0.042 |

Table 1: Controlled VLA harness used for trajectory-level failure prediction, prevention, and recovery.

| Metric | Hand | Learned |
|---|---|---|
| AUC | 0.903 | 0.935 |
| AUPRC | 0.983 | 0.989 |
| Recall | 0.862 | 0.878 |
| Precision | 0.959 | 0.968 |
| False-block rate | 0.222 | 0.175 |

Table 2: Pre-execution failure prediction on held-out test instructions. The learned score uses validation-tuned weights over the same trajectory-risk components.

from combining multiple trajectory signals rather than relying on one uncertainty proxy.

The second experiment converts prediction into prevention. A prevention policy blocks or intervenes when risk exceeds a threshold. Different deployments require different thresholds: a household assistant, warehouse robot, and medical robot would not tolerate the same rate of false blocking. We therefore report a prevention frontier rather than a single operating point.

Table 3 shows that the risk score exposes a useful operating frontier. At a strict target false-block rate of 0.01, the system catches 50.7% of failures with 99.7% precision. At a 0.10 false-block target, it catches 80.6% of failures with 98.0% precision. At a more conservative 0.20 target, recall rises to 85.2% while precision remains 96.3%. Thus the same trajectory signal can be tuned either for high throughput or for stronger prevention.

## 7 Post-Action Recovery

The third experiment evaluates recovery after execution. Prevention will never catch every failure: some risks remain below threshold, and some systems may choose permissive thresholds to avoid excessive blocking. After action, the robot observes a new world state and can compare the achieved state against the intended outcome. If the outcome mismatch is recoverable, the robot should select a safe corrective policy rather than treating the episode as terminal.

We evaluate a corrective controller on corrupted or failed states. The primary metric is manifold-level recovery: whether the corrective action restores the task-relevant state manifold. This is different from exact next-action imitation. After a failure, the world is no longer in the nominal state, so the correct recovery behavior may differ from the expert action that would have been taken without corruption.

| Target FBR | Recall | Precision | Actual FBR |
|---|---|---|---|
| 0.01 | 0.507 | 0.997 | 0.010 |
| 0.05 | 0.732 | 0.989 | 0.049 |
| 0.10 | 0.806 | 0.980 | 0.098 |
| 0.20 | 0.852 | 0.963 | 0.195 |

Table 3: Prevention frontier. Thresholds trade off failure recall against false blocking of safe actions.

Table 4 shows that recovery succeeds in 91.6% of manifold-level failures and reduces mean manifold error from 0.472 to 0.406, a relative reduction of 16.4%. The verification rate is lower, at 0.269, which separates two parts of recovery: moving the state back toward feasibility and explicitly verifying that the intended outcome has been restored. This distinction is important for deployment. A recovery policy may repair the task state, but a complete safety stack should also verify the corrected postcondition before continuing.

A secondary exact-next-state metric worsens under recovery. We do not treat this as a failure of the recovery objective. Exact imitation is the wrong target after the world has changed. Once a robot has entered a corrupted state, the relevant question is not whether it reproduces the nominal expert next action, but whether it safely restores the task-relevant condition.

## 8 Discussion and Conclusion

The experiments support the trajectory-level view of VLA reliability. Failure is not merely a terminal label assigned after the robot acts. In many cases, failure is preceded by measurable risk in the pre-action trajectory: weak policy support, referent ambiguity, diffuse action assignment, unsafe mass, or instability. This matters because the cost of intervention is much lower before physical execution. A robot can clarify, replan, constrain its action set, or hand off before it moves hardware or changes the scene.

The prevention frontier is the most deployment-relevant result. There is no universal threshold for robot intervention. Different settings require different tradeoffs between catching failures and blocking safe actions. A useful risk signal should there-

| Recovery metric | Value |
|---|---|
| Manifold recovery success rate | 0.916 |
| Verification rate | 0.269 |
| Mean corrupt manifold error | 0.472 |
| Mean recovered manifold error | 0.406 |
| Relative manifold error reduction | 0.164 |

Table 4: Post-action recovery results. Recovery improves the task-relevant manifold state after failure.

fore provide a controllable operating curve, not only a single accuracy number. Our results show that trajectory-level risk can support this tradeoff under held-out instructions.

Recovery completes the picture. Some failures will escape prediction, and some risky actions will be allowed under permissive thresholds. For those cases, the system must treat the observed post-action world state as a new input: detect mismatch, classify the failure, select a safe corrective policy, and verify the outcome. The recovery experiment shows that task-manifold repair is possible even when exact next-state imitation is not the right objective.

Overall, this paper reframes VLA failure as a trajectory-level phenomenon. A robot failure is expensive because it occurs in the world, but it may become visible before the world changes. By modeling VLA systems as embodied bounded interpreters, we can decompose failures into outcome, referent, action, policy, and safety components; predict risk before execution; prevent some failures through thresholded intervention; and recover from others after observed mismatch. The goal is not to claim complete robot safety. The goal is to move from post-hoc failure diagnosis toward pre-action failure prevention and post-action repair.

## Limitations

This study uses a controlled VLA harness rather than deployment on physical robots. The held-out-instruction split tests generalization beyond memorized templates, but the environment remains synthetic and bounded. The results therefore support the claim that trajectory-level risk can predict, prevent, and repair failures in a controlled embodied setting; they do not establish complete safety for open-world robotics. Future work should evaluate the same trajectory-risk stack on physical manipulation systems, richer continuous-control policies, and longer-horizon tasks with human-in-the-loop intervention.

## Ethics Statement

The motivation of this work is to reduce costly and unsafe robot failures by detecting risk before action execution and supporting recovery after mismatch. The main ethical risk is overclaiming: a high pre-action prediction score in a controlled harness should not be treated as a guarantee of real-world safety. Any deployed robotic system using such methods should include conservative constraints, physical safety mechanisms, human override, and post-action verification. The proposed framework is intended as an additional layer for risk detection and intervention, not as a replacement for established robot safety practices.

## References


Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and 26 others. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, and 33 others. 2023. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems*.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, and 3 others. 2023. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*,

volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR.

Javier García and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. 2017. Uncertainty-aware reinforcement learning for collision avoidance. In *IEEE International Conference on Robotics and Automation*, pages 5738–5745. IEEE.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2025. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. 2024. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*.

Open X-Embodiment Collaboration. 2024. Open x-embodiment: Robotic learning datasets and RT-X models. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. 2021. Recovery RL: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag Sanketi, Grecia Salazar, and 35 others. 2023. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR.

‘

# A Formal Details and Proofs

**VLA input and policy trajectory.** At physical time step $t$, the VLA policy receives a multimodal context $X_t = (c, u, E_t, s_t, a_{<t})$, consisting of prior context, language instruction, visual/environment observation, robot state, and action history. A transformer policy maps this context to a finite sequence of hidden states $H^{(0)}(X_t), \ldots, H^{(L)}(X_t)$. Let $g_t^{(\ell)}$ denote the layer-$\ell$ action-query state, obtained from an action token, pooled policy state, or instruction-conditioned readout. For a discrete action space, the layerwise action logits are $\eta_t^{(\ell)} = g_t^{(\ell)} W_A$, and the corresponding policy is $\pi_t^{(\ell)} = \mathrm{softmax}(\eta_t^{(\ell)})$. For continuous control, the same role may be played by policy parameters such as a mean and covariance. In either case, the VLA system induces a finite action-policy trajectory $\mathcal{T}_{\mathrm{act},t} = (\pi_t^{(0)}, \ldots, \pi_t^{(L)})$.

**Embodied interpretation variables.** We decompose embodied interpretation into five coordinates: intended outcome $O$, referent or object-role binding $R$, action assignment $A$, policy or plan support $\Pi$, and safety envelope $\Sigma$. This gives an interpretation tuple $I_{\mathrm{VLA}}(X_t) = (O, R, A, \Pi, \Sigma)$. The corresponding failure modes are outcome failure, referent failure, action-assignment failure, policy failure, and safety-envelope failure. The tuple is not intended as a complete semantics of robotics; it is a practical decomposition of the state needed for safe embodied execution.

**Trajectory risk.** The trajectory-level risk score combines five interpretable signals: referent ambiguity, action ambiguity, weak policy support, unsafe-action mass, and layerwise instability. In the experiments, these components are computed from pre-action policy readouts and combined either by a hand-designed score or by validation-learned weights. The hand-designed score has the form $\rho_{\mathrm{VLA}} = \lambda_R \rho_R + \lambda_A \rho_A + \lambda_\Pi \rho_\Pi + \lambda_\Sigma \rho_\Sigma + \lambda_\Delta \Delta$. Here, $\rho_R$ increases when support is split

across candidate referents, $\rho_A$ increases when action support is diffuse, $\rho_\Pi$ increases when intended-policy support is weak, $\rho_\Sigma$ increases with unsafe-action mass, and $\Delta$ measures movement between adjacent interpreted states. A threshold on $\rho_{\mathrm{VLA}}$ converts prediction into prevention.

### A.1 Multimodal Bounded Interpreter Theorem

**Theorem.** Let a VLA policy receive a finite multimodal input $X_t = (c, u, E_t, s_t, a_{<t})$, where $c$ is prior context, $u$ is the instruction, $E_t$ is the environment observation, $s_t$ is robot state, and $a_{<t}$ is action history. Suppose the policy embeds $X_t$ into a finite sequence $H^{(0)}(X_t) \in \mathbb{R}^{N\times d}$, applies $L$ finite transformer layers, and reads out an action or policy from a finite-dimensional action-query state $g_t^{(L)}$. Then the VLA policy is a bounded embodied interpreter.

**Proof.** The multimodal input $X_t$ contains finitely many embedded elements: language tokens, visual patches or scene tokens, robot-state tokens, and previous-action tokens. Therefore the initial hidden sequence is finite, $H^{(0)}(X_t) \in \mathbb{R}^{N\times d}$. At each layer, the transformer applies a finite-dimensional map $H^{(\ell+1)} = F^{(\ell)}_{\theta,\mathrm{VLA}}(H^{(\ell)})$. Because the number of layers $L$ is finite, repeated application of these maps yields a finite hidden-state trajectory $H^{(0)}, H^{(1)}, \ldots, H^{(L)}$.

Let $g_t^{(\ell)}$ be the action-query state at layer $\ell$. For a discrete action space, the model computes layerwise action logits $\eta_t^{(\ell)} = g_t^{(\ell)} W_A$ and a policy $\pi_t^{(\ell)} = \mathrm{softmax}(\eta_t^{(\ell)})$. For continuous control, the readout instead produces finite-dimensional policy parameters, such as a mean and covariance. In either case, the policy output is a finite readout from $g_t^{(\ell)}$. Thus inference consists of finite multimodal state evolution followed by finite action or policy readout. The model can interpret the embodied task only within the limits of its finite context, finite hidden dimension, finite layer count, and finite readout. Therefore it is bounded. Since the finite state is conditioned on language, visual observation, robot state, and action history, the bounded state is embodied. Hence the VLA policy is a bounded embodied interpreter.

### A.2 Embodied Interpretation Support Theorem

**Theorem.** Let the embodied interpretation state be $I_{\mathrm{VLA}}(X_t) = (O, R, A, \Pi, \Sigma)$, where $O$ is intended outcome, $R$ is referent or object-role binding, $A$ is action assignment, $\Pi$ is policy support, and $\Sigma$ is the safety envelope. If the VLA policy successfully executes $X_t$, then its final policy readout must assign nonzero support to action or policy regions compatible with the required coordinates of $I_{\mathrm{VLA}}(X_t)$.

**Proof.** Consider any required embodied coordinate $Y \in \{O, R, A, \Pi, \Sigma\}$. Let $\Omega_Y(X_t)$ be the action or policy region compatible with that coordinate in context $X_t$. For a discrete policy, the final support assigned to this region is $M_{\Omega_Y}(\eta_t^{(L)}) = \sum_{a\in\Omega_Y} \mathrm{softmax}(\eta_t^{(L)})_a$. For a continuous policy, the same argument applies with probability mass integrated over $\Omega_Y$.

Assume, for contradiction, that the model successfully executes the task but $M_{\Omega_Y}(\eta_t^{(L)}) = 0$. Then the final policy assigns zero probability mass to every action compatible with $Y$. If $Y$ is an outcome coordinate, the model assigns no support to actions that can realize the intended outcome. If $Y$ is a referent coordinate, it assigns no support to actions on the correct object or role. If $Y$ is an action coordinate, it assigns no support to the required action class. If $Y$ is a policy coordinate, it assigns no support to the intended plan region. If $Y$ is a safety coordinate, it assigns no support to safe actions. In each case, successful execution requiring that coordinate would be impossible under zero support. This contradicts the assumption of success. Therefore final policy support for each required compatible coordinate must be nonzero.

### A.3 Embodied Trajectory Divergence Theorem

**Theorem.** Let $X_1$ and $X_2$ be two embodied contexts that share the same instruction but differ in prior context, environment, robot state, or action history. Suppose $I_{\mathrm{VLA}}(X_1) \neq I_{\mathrm{VLA}}(X_2)$. If the VLA policy successfully executes both contexts, then their embodied interpreted trajectories differ.

**Proof.** Since $I_{\mathrm{VLA}}(X) = (O, R, A, \Pi, \Sigma)$, the assumption $I_{\mathrm{VLA}}(X_1) \neq I_{\mathrm{VLA}}(X_2)$ means that at least one coordinate differs. Thus, for some $Y \in \{O, R, A, \Pi, \Sigma\}$, $Y(X_1) \neq Y(X_2)$. The compatible action or policy region for that coordinate is therefore context-dependent. Let $\Omega_Y(X_1)$ and $\Omega_Y(X_2)$ denote the compatible regions for the two contexts.

By the support theorem, successful execution

on $X_1$ requires final support for $\Omega_Y(X_1)$, and successful execution on $X_2$ requires final support for $\Omega_Y(X_2)$. Because the required coordinate differs, the policy must distinguish the two contexts in a $Y$-relevant direction. Otherwise the final interpreted state would not separate the two required action or policy regions.

Let $m^{(L)}_{\mathrm{VLA},t}(X)$ denote the final interpreted state obtained by reading probe masses for $O, R, A, \Pi, \Sigma$ from the final policy readout. Since $Y$ differs between $X_1$ and $X_2$, successful execution requires $m^{(L)}_{\mathrm{VLA},t}(X_1) \neq m^{(L)}_{\mathrm{VLA},t}(X_2)$. Now suppose, for contradiction, that the full embodied interpreted trajectories are identical. If the trajectories are identical, then all corresponding layerwise interpreted states are identical, including the final states. That implies $m^{(L)}_{\mathrm{VLA},t}(X_1) = m^{(L)}_{\mathrm{VLA},t}(X_2)$, contradicting the final-state divergence above. Therefore the embodied interpreted trajectories must differ.

### A.4 Failure Decomposition Theorem

**Theorem.** Let the intended embodied interpretation be $I_{\mathrm{VLA}}(X_t) = (O, R, A, \Pi, \Sigma)$, and let the predicted or realized embodied interpretation be $\widehat{I}_{\mathrm{VLA}}(X_t) = (\widehat{O}, \widehat{R}, \widehat{A}, \widehat{\Pi}, \widehat{\Sigma})$. If $\widehat{I}_{\mathrm{VLA}}(X_t) \neq I_{\mathrm{VLA}}(X_t)$, then the failure decomposes into at least one of outcome, referent, action-assignment, policy, or safety-envelope failure.

**Proof.** Two ordered tuples are unequal if and only if at least one coordinate differs. Therefore $\widehat{I}_{\mathrm{VLA}}(X_t) \neq I_{\mathrm{VLA}}(X_t)$ implies that at least one of the following holds: $\widehat{O} \neq O$, $\widehat{R} \neq R$, $\widehat{A} \neq A$, $\widehat{\Pi} \neq \Pi$, or $\widehat{\Sigma} \neq \Sigma$. Each coordinate corresponds to a failure type. If $\widehat{O} \neq O$, the achieved or predicted outcome differs from the intended outcome, giving outcome failure. If $\widehat{R} \neq R$, the robot has selected the wrong object or object role, giving referent failure. If $\widehat{A} \neq A$, the selected action assignment is wrong. If $\widehat{\Pi} \neq \Pi$, the policy or plan differs from the intended executable support. If $\widehat{\Sigma} \neq \Sigma$, the policy has violated or misrepresented the safety envelope. Therefore any mismatch in embodied interpretation decomposes into at least one of the five failure types.

### A.5 Failure Prediction Theorem

**Theorem.** Let $\rho_{\mathrm{VLA}}$ be a pre-action risk score composed of referent ambiguity, action ambiguity, weak policy support, unsafe-action mass, and trajectory instability. If $\rho_{\mathrm{VLA}} > \delta$ before execution, then the trajectory contains a pre-action warning signal for likely failure.

**Proof.** The risk score is a weighted combination of terms that each measure degradation in a component required for stable embodied execution. Referent ambiguity is high when the policy assigns similar support to competing objects or object roles. This indicates that $R$ is not stably resolved. Action ambiguity is high when no action class dominates. This indicates that $A$ is not stably resolved. Policy weakness is high when the intended plan region has low support. This indicates that $\Pi$ is weak. Unsafe-action mass is high when the policy assigns probability to actions outside the safety envelope. This indicates possible violation of $\Sigma$. Trajectory instability is high when interpreted states change sharply across layers. This indicates that the embodied interpretation has not stabilized.

If the total score exceeds threshold, then at least one of these risk components is large enough to make the weighted sum exceed the intervention boundary. Thus before physical execution, the policy already contains evidence of unresolved referent, action, policy, safety, or stability state. Such evidence is a warning signal for likely execution failure. The theorem does not imply deterministic failure; it implies that the pre-action trajectory contains enough risk evidence to justify prediction or intervention.

### A.6 Failure Prevention Theorem

**Theorem.** If a VLA system detects $\rho_{\mathrm{VLA}} > \delta$ before execution, then it can prevent immediate physical commitment by applying an intervention policy such as clarification, replanning, constrained action selection, guarded execution, or human handoff.

**Proof.** By the failure prediction theorem, $\rho_{\mathrm{VLA}} > \delta$ indicates that the pre-action trajectory contains a warning signal. Because this signal is detected before action execution, the robot has not yet changed the physical world. Therefore the system can choose not to execute the original policy.

The type of intervention can be selected from the dominant risk component. If referent ambiguity is high, the system can ask for clarification or gather more perceptual evidence. If action ambiguity is high, it can replan or defer action selection. If policy support is weak, it can search for a stronger policy or call a higher-level planner. If unsafe-action mass is high, it can constrain the action set to safe regions. If trajectory instability is high, it

can switch to guarded execution or human handoff. Each intervention blocks or modifies the action before physical commitment. Therefore pre-action risk detection enables failure prevention.

### A.7 Failure Recovery Theorem

**Theorem.** Let a robot execute an action under intended outcome $O_t$, observe a post-action outcome $\widehat{O}_{t+1}$, and detect mismatch $d_O(\widehat{O}_{t+1}, O_t) > \epsilon_O$. If there exists a safe corrective policy $\pi_{\text{rec}} \in \Sigma$ such that after $k$ recovery steps $d_O(\widehat{O}_{t+k}, O_t) \leq \epsilon_O$, then the failure is recoverable.

**Proof.** The inequality $d_O(\widehat{O}_{t+1}, O_t) > \epsilon_O$ means the achieved post-action state differs from the intended outcome by more than the allowed tolerance. Thus a post-action failure has occurred.

Now assume there exists a corrective policy $\pi_{\text{rec}}$ inside the safety envelope $\Sigma$. Executing this corrective policy for $k$ steps produces a later observed outcome $\widehat{O}_{t+k}$. By assumption, $d_O(\widehat{O}_{t+k}, O_t) \leq \epsilon_O$. Therefore the corrected state satisfies the intended outcome within tolerance. Because $\pi_{\text{rec}} \in \Sigma$, the correction remains safe. Thus the failure is not terminal: it can be repaired by safe corrective action. A VLA system supports recovery when it can detect the mismatch, choose such a corrective policy, execute it safely, and verify the corrected state.

### A.8 Recovery as Trajectory Repair

**Claim.** Post-action recovery can be viewed as trajectory repair rather than continuation of the original nominal policy.

**Proof.** Before failure, the robot follows a policy trajectory conditioned on $X_t$. After execution, the environment changes and the robot observes a new state. If the observed state violates the intended postcondition, then the original trajectory is no longer the correct reference, because the world has moved to a different state. The recovery problem therefore takes the failed post-action state as a new input.

Let the failed trajectory produce action $a_t$ and observed state $\widehat{E}_{t+1}$. The recovery controller maps the new recovery context to a corrective policy, $(\widehat{E}_{t+1}, O_t, \widehat{I}_{\text{VLA}}) \mapsto \pi_{\text{rec}}$. This is not required to match the original expert next action. It is required to restore the intended outcome safely. Therefore recovery is trajectory repair after world-state mismatch, not imitation of the nominal path.

## B Dataset and Training Details

The empirical harness is a controlled VLA manipulation setting with language-conditioned tasks over objects, receptacles, drawers, spatial relations, and safety-sensitive actions. Each example contains an instruction, embodied state, robot-state/action context, target policy label, and failure annotation. The split is held out by instruction template, so test examples use instructions unseen during training.

| Quantity | Value |
|---|---|
| Total samples | 21,389 |
| Train samples | 13,773 |
| Validation samples | 2,828 |
| Test samples | 4,788 |
| Split type | Held-out instruction |
| Failure buckets | 5 |
| Training epochs | 8 |
| Final validation loss | 0.042 |

Table 5: Dataset and training summary.

| Bucket | Description |
|---|---|
| Outcome | Achieved state misses the intended postcondition. |
| Referent | Policy selects the wrong object or object role. |
| Action | Action class or motor primitive is incompatible with the task. |
| Policy | Plan support is weak, reversed, or unstable. |
| Safety | Policy enters unsafe or forbidden action regions. |

Table 6: Failure buckets used in the controlled VLA harness.

| Epoch | Train loss | Validation loss |
|---|---|---|
| 1 | 0.844 | 0.133 |
| 2 | 0.194 | 0.077 |
| 3 | 0.123 | 0.060 |
| 4 | 0.100 | 0.053 |
| 5 | 0.086 | 0.046 |
| 6 | 0.077 | 0.047 |
| 7 | 0.071 | 0.044 |
| 8 | 0.065 | 0.042 |

Table 7: Training history for the compact VLA policy.

## C Risk Prediction and Prevention Details

The hand-designed risk score uses fixed interpretable components. The learned score uses the same components plus final action entropy, with weights fit on validation data. Both scores are evaluated on held-out test instructions.

| Metric | Validation | Test |
|---|---|---|
| AUC | 0.917 | 0.903 |
| AUPRC | 0.986 | 0.983 |
| Recall at threshold | 0.845 | 0.862 |
| Precision at threshold | 0.981 | 0.959 |
| False-block rate | 0.097 | 0.222 |

Table 8: Hand-designed trajectory risk score.

| Component | Learned weight |
|---|---|
| Referent risk | 6.765 |
| Action ambiguity | -1.574 |
| Policy weakness | 7.850 |
| Unsafe-action mass | -0.051 |
| Trajectory instability | -2.319 |
| Final action entropy | -1.729 |
| Intercept | -0.835 |

Table 9: Validation-learned weights for the weighted risk score.

## D Recovery Details

Recovery is evaluated after post-action mismatch. The primary metric is manifold-level recovery: whether the corrective controller restores the task-relevant state manifold. This is distinct from exact next-state imitation, which is not the correct objective after the world has been perturbed.

## E Embodied Context Divergence

We also test whether differences in embodied context correspond to differences in final-layer action readouts. Context pairs are constructed under held-out test instructions. We measure context distance and final-layer action divergence.

| Metric | Value |
|---|---|
| Validation threshold | 0.259 |
| Validation recall | 0.873 |
| Validation precision | 0.982 |
| Validation false-block rate | 0.097 |
| Test AUC | 0.935 |
| Test AUPRC | 0.989 |
| Test recall | 0.878 |
| Test precision | 0.968 |
| Test false-block rate | 0.175 |
| Positive base rate | 0.857 |
| AUPRC lift over base | 1.154 |

Table 10: Learned weighted risk score.

| Score | AUC | AUPRC |
|---|---|---|
| Learned weighted risk | 0.935 | 0.989 |
| Policy support only | 0.930 | 0.987 |
| Full trajectory risk | 0.903 | 0.983 |
| Safety mass only | 0.500 | 0.857 |
| Action ambiguity only | 0.500 | 0.857 |
| Trajectory instability only | 0.500 | 0.857 |
| Entropy only | 0.500 | 0.857 |
| Random | 0.496 | 0.856 |

Table 11: Baseline comparison on held-out test instructions.

| Target FBR | Recall | Precision | Actual FBR |
|---|---|---|---|
| 0.01 | 0.507 | 0.997 | 0.010 |
| 0.05 | 0.732 | 0.989 | 0.049 |
| 0.10 | 0.806 | 0.980 | 0.098 |
| 0.20 | 0.852 | 0.963 | 0.195 |

Table 12: Prevention frontier. Thresholds trade off failure recall against false blocking of safe actions.

| Metric | Value |
|---|---|
| Manifold recovery success rate | 0.916 |
| Verification rate | 0.269 |
| Mean corrupt manifold error | 0.472 |
| Mean recovered manifold error | 0.406 |
| Mean manifold error reduction | 0.066 |
| Relative manifold error reduction | 0.164 |

Table 13: Post-action manifold recovery results.

| Exact-state metric | Value |
|---|---|
| Corrupt exact next-state error | 0.044 |
| Recovered exact next-state error | 0.135 |
| Exact error reduction | -0.090 |

Table 14: Secondary exact-next-state metric. Recovery worsens this metric because it optimizes task-manifold repair rather than nominal next-action imitation.

| Metric | Value |
|---|---|
| Context pairs | 1,200 |
| Mean context distance | 1.296 |
| Mean action divergence | 1.379 |
| Spearman correlation | 0.157 |
| Spearman $p$-value | $4.92 \times 10^{-8}$ |

Table 15: Embodied context/action divergence. Context differences are significantly associated with action-readout divergence.