REAL-WORLD DEBIASING: FROM DEPLOYMENT FAILURES TO CAUSAL REPAIR OF POLICY SHORTCUTS

Anonymous authors
Paper under double-blind review

ABSTRACT

In practice, visual control policies can fail after deployment despite strong source performance. Prior work has identified hidden reliance on task-irrelevant source cues as one cause of such failures, even when the underlying task remains unchanged. Yet diagnosing and repairing this dependence after deployment is difficult: multiple factors may change simultaneously, and target interventions or retraining may be costly or infeasible. We call this broader failure pattern *real-world bias*. We further show that observational source data alone need not identify such harmful dependence, even when source performance is optimal. To address this gap, we introduce RENDER, which uses a few unpaired target failure trajectories to guide causal diagnosis and selective repair in the source simulator. RENDER reenacts candidate nuisance mechanisms to reproduce the observed failures, then tests whether task-preserving interventions change policy decisions and increase task loss. Once harmful dependence is verified, the diagnosed mechanism guides targeted adaptor training using source data while the original policy remains frozen; otherwise, RENDER abstains. The procedure requires no target interventions. Experiments across diverse visual control tasks support our analysis and show that selective repair can improve deployment performance.

1 INTRODUCTION

Visual control policies learned through reinforcement learning or imitation learning often rely on simulators, benchmarks, or fixed demonstration datasets. Their actions should depend on task-relevant state, such as object motion or spatial position. Yet task-irrelevant factors, such as object appearance or environmental lighting, can correlate with appropriate actions during training. A policy may rely on these cues even though they do not change what the task requires.

Imagine deploying a robot policy trained entirely on Earth to the Moon. The policy may achieve nearly perfect source performance, yet fail immediately after deployment. Such a failure is hard to diagnose from either domain. On the source side, strong performance provides no evidence about which Earth-specific regularity the policy exploits. On the target side, many factors change simultaneously, so the observed failures identify neither the responsible mechanism nor the part of the policy to correct. Retraining in the target environment may be too expensive or infeasible, so target feedback may consist of only a few unpaired failure trajectories (Figure 1).

Prior work has documented this failure pattern across sequential decision settings. In Atari Pong, reinforcement-learning agents use the opponent's paddle position as a proxy for the ball's vertical position; hiding the opponent while it continues playing reduces performance to random-play levels (Delfosse et al., 2024). In offline reinforcement learning and imitation learning, spurious previous-action cues can impair learning or fail during closed-loop execution (Gupta et al., 2023; Ross et al., 2011; de Haan et al., 2019; Park et al., 2021). Robot policies can associate tasks with viewpoints or environments and fail when these are recombined (Xing et al., 2025). These studies reveal instances of a problem, addressed through domain randomization, active sampling, causal auditing, counterfactual augmentation, data curation, or human corrections (Tobin et al., 2017; Gupta et al., 2023; Gajcin & Dusparic, 2022; Agia et al., 2025; Ameperosa et al., 2025; Peng et al., 2023). Yet these approaches typically assume a specified bias or intervention, or require bias-specific human feedback. An unresolved deployment challenge is therefore to determine whether harmful task-irrelevant

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.

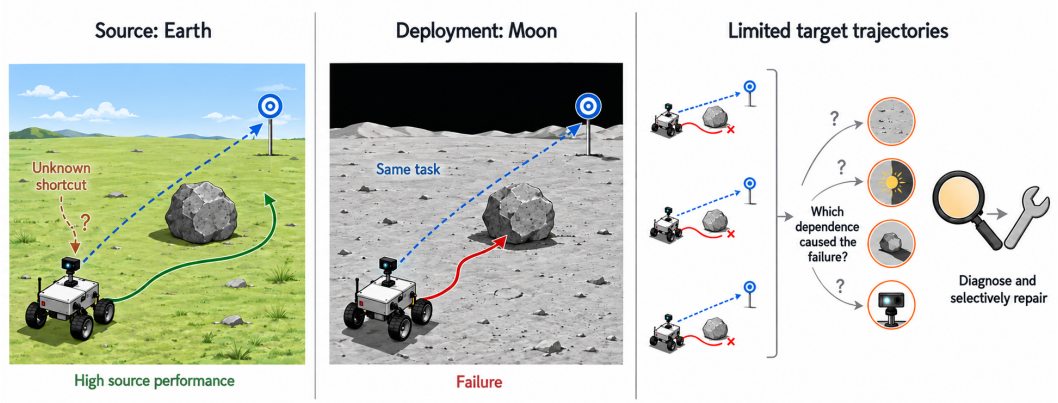


Figure 1: A policy can achieve high source performance by relying on an unknown source specific correlate. Deployment may break this correlation while preserving the task, causing the policy to fail. Given only a few unpaired target failure trajectories, real-world debiasing aims to diagnose the hidden dependence and selectively repair the policy.

dependence explains an already-trained policy's failure, when source evaluation did not reveal the cause and only a few unpaired target failure trajectories are available.

We call this broader failure pattern **real-world bias**. It arises when a policy relies on a task-irrelevant factor that predicts task-appropriate actions in the source environment without causally determining them. This dependence can remain hidden behind strong source performance, yet become harmful when deployment changes the factor while preserving the task.

Therefore, in general deployment settings, two questions remain:

- *Which hidden source dependence explains the observed deployment failures?*
- *How can this dependence be repaired without target interventions or full-policy retraining?*

To answer these questions, we introduce RENDER, a framework for **RE**enactment-based **N**uisance **D**ependence **E**stimation and **R**epair. RENDER summarizes target failures, reenacts candidate mechanisms in the source simulator, verifies harmful dependence through task-preserving interventions, and selectively repairs the diagnosed dependence while keeping the control network frozen. It abstains when diagnosis cannot be verified and requires no target interventions.

Contributions. First, we formalize real-world bias, unifying shortcut failures reported in prior work through a causal account, and explain why source success can conceal harmful dependence: mechanisms indistinguishable from source observations can respond differently to task-preserving interventions. Second, we introduce RENDER, which uses a few unpaired deployment failures to guide source reenactment, causal verification, and targeted adaptor repair, without target interventions or policy updates. Third, we show that identifying a harmful source dependence is insufficient for repair: a policy can harbor several harmful dependencies, and correcting one unrelated to the failure can worsen target performance, so diagnosis must find the dependence deployment exposed.

2 Real-World Bias

In this section, we formalize real-world bias and provide a theoretical analysis of why harmful nuisance dependence can remain hidden behind strong source performance. We show that causal mechanisms indistinguishable from source observations can yield different policy responses and task losses under task-preserving interventions, even when the policy is source-optimal. This analysis clarifies the limits of observational diagnosis and motivates an intervention-based approach to identifying nuisance dependence and verifying its harmful consequences.

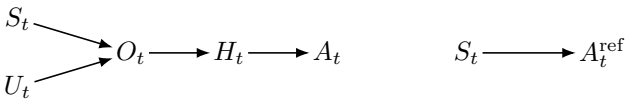
2.1 Setup

Let $\mathcal{S}$ be the task state space. At time t, let $S_t \in \mathcal{S}$ denote the task state, which contains the information relevant to task appropriate behavior, and let U_t denote nuisance information that can

affect observation but is irrelevant to task appropriate behavior. Let $\pi_{\mathrm{ref}}(\cdot \mid S_t)$ denote the reference policy, which specifies the task appropriate action distribution. In reinforcement learning, it can be an optimal policy under the task reward. In imitation learning, it can be the expert policy that generated the demonstrations. The agent observes O_t and acts from the history $H_t = (O_{1:t}, A_{1:t-1})$. Let A_t^{ref} be a reference action and A_t the action selected by the learned policy. With f denoting the observation function and π the learned policy, these relationships are

$$O_t = f(S_t, U_t), \qquad A_t^{\mathrm{ref}} \sim \pi_{\mathrm{ref}}(\cdot \mid S_t), \qquad A_t \sim \pi(\cdot \mid H_t). \tag{1}$$

Thus the learned policy acts through its history, while the reference policy uses S_t directly:



(2)

S_t and U_t jointly generate the observation and may be associated in the source environment, so the task and nuisance paths may both contribute to the learned action.

Let P be the state transition kernel, so that $S_{t+1} \sim P(\cdot \mid S_t, A_t)$, and let $r(S_t, A_t)$ be the expected immediate reward. An intervention I changes nuisance information in the policy input. For example, $I = \mathrm{do}(U_t = u)$ sets the nuisance value to u (Pearl, 2009). A superscript I marks a quantity under the intervention. In particular, H_t^I is the policy history after applying I at the current decision.

Definition 1 (task-preserving intervention). Let $\mathcal{S}_I$ be the task states where an intervention is used. The intervention is task-preserving on $\mathcal{S}_I$ if it may change H_t to H_t^I with $S_t = s$ fixed, H_t^I still contains the information about s needed for decisions, and, for every $s \in \mathcal{S}_I$ and a,

$$\pi_{\mathrm{ref}}^I(\cdot \mid s) = \pi_{\mathrm{ref}}(\cdot \mid s), \qquad P^I(\cdot \mid s, a) = P(\cdot \mid s, a), \qquad r^I(s, a) = r(s, a). \tag{3}$$

In short, a task-preserving intervention may change a nuisance cue, but it does not change the task, remove information needed to solve it, or alter the effect of an action. The information condition is stated separately because hiding an object needed for the decision can leave all three equalities intact. The intervention may still change the policy's action and thus its trajectory and return.

Definition 2 (real-world bias). Consider a policy π that performs well in the source environment, and let μ be the distribution of deployment histories of interest. For a task-preserving intervention I, let $\mathcal{L}_{\mathrm{task}}^I(\pi; H_t)$ and $\mathcal{L}_{\mathrm{task}}(\pi; H_t)$ denote the expected future task loss with and without applying I from time t onward. Define the intervention induced excess task loss

$$\mathcal{B}_I(\pi; \mu) = \mathbb{E}_{H_t \sim \mu}\left[\mathcal{L}_{\mathrm{task}}^I(\pi; H_t) - \mathcal{L}_{\mathrm{task}}(\pi; H_t)\right]. \tag{4}$$

The policy has real-world bias with respect to (I, μ) if $\mathcal{B}_I(\pi; \mu) > 0$.

By Definition 1, I changes the nuisance while preserving the task. Hence, $\mathcal{B}_I(\pi; \mu) > 0$ means that the policy responds to the nuisance change in a way that increases expected task loss. Harmless changes are not real-world bias, and a biased policy may still retain useful task competence.

Definition 2 characterizes the harmful dependence underlying this deployment problem. We study its diagnosis and repair when the responsible nuisance mechanism is unknown, target feedback consists of a few unpaired failure trajectories, and target interventions are unavailable.

Examples of real-world bias. Here we present several previously reported cases in which nuisance cues change while the task remains unchanged. In driving, a dashboard cue linked to the expert's previous action can become misleading under the learned policy's control (de Haan et al., 2019). In robot manipulation, changes in viewpoint or background can disrupt associations learned from demonstrations (Xing et al., 2025). In visual control, score displays or object colors can change without altering the reference policy or dynamics (Peng et al., 2023; Park et al., 2021). Reliance on these cues can turn such changes into altered actions and increased task loss.

2.2 Why Source Success Can Hide Bias

Source evaluation observes behavior under the associations present during training, and a task-preserving intervention breaks those associations while keeping the task. High source return therefore need not establish robustness to the intervention (Peters et al., 2016; Schölkopf et al., 2021).

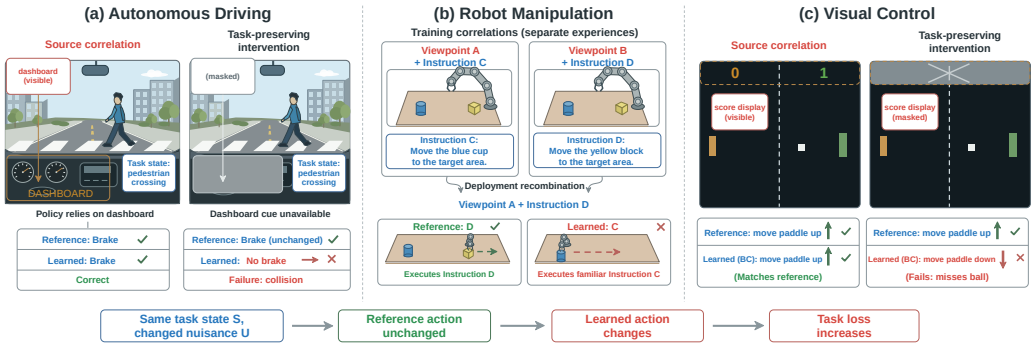


Figure 2: Real-world bias. In each setting, the task state and reference action distribution are preserved while a nuisance cue changes. A biased policy changes its action and incurs greater task loss. The examples illustrate dashboard reliance in driving, viewpoint reliance in robot manipulation, and score-display reliance in visual control (de Haan et al., 2019; Xing et al., 2025; Park et al., 2021).

To make this precise, let M specify the causal mechanisms in the setup and P^M_{src} its source record distribution under a protocol that allows action exploration but no nuisance interventions. For fixed π, I, and μ, the excess task loss $\mathcal{B}^M_I(\pi;\mu)$ is *identifiable* from source records if all models satisfying the task assumptions and matching P^M_{src} agree on its value (Pearl, 2009).

Proposition 1 (Source equivalence can conceal real-world bias)**.** *The setup admits two causal models M_0, M_1 with the same task mechanisms, the same source law of (S_t, U_t), a common policy π that is source optimal in both, a common initial source history distribution μ, and an intervention I that is task-preserving in both and sets the nuisance to a value that occurs in the source, such that*

$$P^{M_0}_{\mathrm{src}} = P^{M_1}_{\mathrm{src}}, \qquad \mathcal{B}^{M_0}_I(\pi;\mu) = 0 < \mathcal{B}^{M_1}_I(\pi;\mu). \tag{5}$$

Source records therefore do not identify real-world bias in this model class, even when they record task states, nuisance values, and reference actions.

In the construction (Appendix A.1), M_0 produces the observed cue from the task state and M_1 from a separate mechanism that agrees with it in the source. The policy reads the cue and is defined on every input, so the records leave open not its behavior but the input the intervention produces. The intervened value occurs in the source, but never with a different task state; only such combinations separate the models. More source records never contain one; simulating I produces them.

Proposition 2 (Interventional access identifies real-world bias)**.** *Suppose the source simulator draws $H_t \sim \mu$ with its state and rolls out π from it with and without I. Let the realized task losses Y^0, Y^I satisfy $\mathbb{E}[Y^0 \mid H_t] = \mathcal{L}_{\mathrm{task}}(\pi; H_t)$ and $\mathbb{E}[Y^I \mid H_t] = \mathcal{L}^I_{\mathrm{task}}(\pi; H_t)$. Then $\mathcal{B}_I(\pi;\mu) = \mathbb{E}[Y^I - Y^0]$ is identifiable. With finite variances, the mean of n i.i.d. pairs $Y^I_i - Y^0_i$ is unbiased and consistent, and pairing changes its variance, relative to independent rollouts, by $-2\,\mathrm{Cov}(Y^I, Y^0)/n$.*

Together the propositions locate the missing information: how the policy's input responds when the nuisance changes at a fixed task state. RENDER obtains it by intervention, and asks of the result whether the change moves the policy's decision and whether it increases task loss.

3 Method

In this section, we present RENDER, a four-step procedure that uses target failures to guide diagnosis and repair in the source simulator (Figure 3). Using target failure trajectories containing only observations, actions, rewards, and termination information, we identify candidate nuisance mechanisms and test them through source interventions and rollouts. Upon confirming harmful dependence, we construct source supervision from the diagnosed mechanism to train an adaptor and any required outcome predictor, keeping the policy π frozen throughout. Appendix B gives details.

3.1 Step 1: Read target failures as causal clues

First, we use n target failure trajectories $\mathcal{D}^{\mathrm{fail}} = \{\tau_i\}_{i=1}^n$ to narrow the candidate mechanisms. Each trajectory is mapped to a prespecified observable signature $\phi(\tau)$, typically its return, episode length,

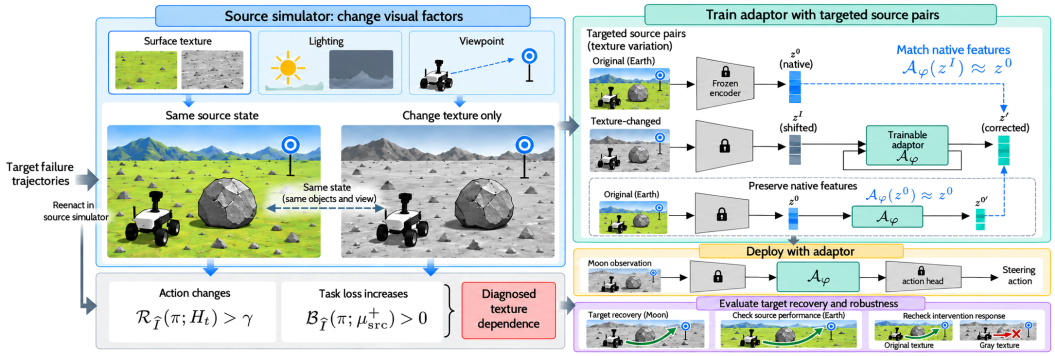


Figure 3: The four steps of RENDER, illustrated by texture dependence in the Earth to Moon example. Step 4 shows the feature correction variant.

reward timing, and empirical action distribution. It may also include observable events such as a safety violation, but not hidden target state. Different mechanisms leave different signatures, such as earlier failures or shifted action frequencies, so their distribution narrows the candidates.

3.2 Step 2: Reenact candidate causes in the source simulator

Next, we reenact candidate mechanisms using an intervention library $\mathcal{M}$ fixed before target evaluation. Each family $m \in \mathcal{M}$ represents one mechanism, such as appearance, illumination, viewpoint, or action interface, and $\mathcal{I}_m$ contains the values tested for that mechanism. We first remove interventions that change the task or hide information it needs. For each remaining $I \in \mathcal{I}_m$, let $P^{\pi,I}_{\text{src}}$ be the source trajectory distribution under the frozen policy π and intervention I. We collect fresh source episodes and retain their failures under the prespecified failure rule. Let $\Phi_{\text{tgt,sel}}$ and $\Phi^I_{\text{src,sel}}$ denote the empirical distributions of target and source failure signatures on the selection split, retaining observed frequencies. The matching score $d_{\text{sel}}(I)$ uses a scaled energy distance D_ϕ:

$$\tau^I_j \sim P^{\pi,I}_{\text{src}}, \qquad d_{\text{sel}}(I) = D_\phi\big(\Phi_{\text{tgt,sel}}, \Phi^I_{\text{src,sel}}\big)\,. \tag{6}$$

Only candidates with enough failures are eligible. Selection chooses the smallest-distance intervention within each family and then the best family. Both stay fixed while confirmation on disjoint episodes accepts or rejects the selected mechanism. Acceptance requires sufficient failure recurrence, a close signature match, stability under resampling, and separation from alternative families. The distance and recurrence thresholds are calibrated on source pseudo targets with known mechanisms, so a correct match fails either with probability at most 5% (Appendix B.3). We denote the confirmed family and intervention by $\widehat{m}$ and $\widehat{I}$. If confirmation fails, RENDER returns `none_of_the_above`.

3.3 Step 3: Test for real-world bias under the selected intervention

We then test the selected mechanism on paired source histories. Matching alone does not show that the candidate cue changes the policy or harms the task, so we test these two claims separately.

Policy response. The first test asks whether the selected intervention changes the policy output. For each source history H_t, we construct $H^{\widehat{I}}_t$ with the same underlying task history and a different nuisance value. Let I_0 denote a placebo replay that reconstructs the factual view without changing the nuisance, producing $H^{I_0}_t$. The raw policy change $\Delta_I(\pi; H_t)$ is a discrepancy d_A between the factual and intervened action distributions. The response score $\mathcal{R}_{\widehat{I}}(\pi; H_t)$ subtracts the placebo change and truncates at zero:

$$\begin{aligned} \Delta_I(\pi; H_t) &= d_A\big(\pi(\cdot \mid H_t), \pi(\cdot \mid H^I_t)\big)\,, \qquad I \in \{\widehat{I}, I_0\}, \\ \mathcal{R}_{\widehat{I}}(\pi; H_t) &= \big[\Delta_{\widehat{I}}(\pi; H_t) - \Delta_{I_0}(\pi; H_t)\big]_+\,. \end{aligned} \tag{7}$$

Here $[x]_+ = \max\{x, 0\}$. For deterministic policies, the distributions are point masses on the selected actions. A response above the placebo calibrated threshold γ marks an affected decision. Let

μ_{src} be the source audit distribution and μ^+_{src} its conditional distribution over affected histories:

$$\mu^+_{\mathrm{src}} = \mu_{\mathrm{src}}\big(\cdot \mid \mathcal{R}_{\widehat{I}}(\pi; H_t) > \gamma\big)\,. \tag{8}$$

Task harm. The second test asks whether the response raises task loss, as Definition 2 requires. On histories from μ^+_{src}, we compare factual and intervened continuations by their excess task loss

$$\mathcal{B}_{\widehat{I}}(\pi; \mu^+_{\mathrm{src}}) = \mathbb{E}_{H_t \sim \mu^+_{\mathrm{src}}}\Big[\mathcal{L}^{\widehat{I}}_{\mathrm{task}}(\pi; H_t) - \mathcal{L}_{\mathrm{task}}(\pi; H_t)\Big]\,. \tag{9}$$

The response score selects affected decisions; the excess task loss measures their consequences without weighting losses by response magnitude. Under task preservation, a positive excess loss meets Definition 2 on these audited source histories; action sensitivity alone does not. If either test fails, RENDER abstains; otherwise, $\widehat{I}$ guides repair construction in Step 4.

3.4 Step 4: Train an adaptor for the diagnosed bias

Finally, we train an adaptor on task-preserving source variations of the diagnosed cue while π stays frozen. It corrects features from matched native views or decisions from source rollout outcomes.

Feature correction. For visual RL policies such as Atari, matched source views provide native feature targets that the trained action head already uses. Write the frozen policy as $\pi_\theta(\cdot \mid H_t) = h_\theta(\cdot \mid e_\theta(H_t))$, with encoder e_θ and action head h_θ. A matched pair gives native features $z^0 = e_\theta(H_t)$ and intervened features $z^I = e_\theta(H^I_t)$ at the same task state and history. With a trainable correction $r_\varphi(z)$, the frozen head reads the corrected feature; a zero correction recovers π:

$$\mathcal{A}_\varphi(z) = z + r_\varphi(z), \qquad \pi_R(\cdot \mid H_t) = h_\theta(\cdot \mid \mathcal{A}_\varphi(e_\theta(H_t)))\,. \tag{10}$$

The objective restores intervened features and keeps native ones, with preservation weight $\lambda_{\mathrm{src}} \geq 0$:

$$\mathcal{L}_{\mathrm{feat}}(\varphi) = \mathbb{E}_{\mathrm{paired}}\left\|\mathcal{A}_\varphi(z^I) - z^0\right\|_2^2 + \lambda_{\mathrm{src}}\mathbb{E}_{\mathrm{native}}\left\|\mathcal{A}_\varphi(z^0) - z^0\right\|_2^2. \tag{11}$$

Decision correction from source outcomes. For IL policies such as CARLA's behavior cloned controller, a cue correlated with expert actions can become unreliable under the policy's own control. Source rollout outcomes compare the decisions obtained by varying that cue. For each admitted view v of the history H_t, the frozen policy supplies a candidate distribution $p_v(a \mid H_t)$. A predictor fitted to source outcomes assigns a score $s_\psi(H_t, v)$, larger for better outcomes. We combine candidates as

$$\bar{p}_\psi(a \mid H_t) = \sum_v w_v(H_t) p_v(a \mid H_t), \qquad w_v(H_t) = \frac{\exp[\beta s_\psi(H_t, v)]}{\sum_{v'} \exp[\beta s_\psi(H_t, v')]}. \tag{12}$$

Here $\beta \geq 0$ controls how strongly predicted outcomes affect the vote; $\beta = 0$ gives equal weights. In CARLA, scores come from collisions and the weighted speed distribution drives a gated runtime adaptor; in ForkRoad, continuation returns give audited labels for a gated residual adaptor.

3.5 Evaluating a repair

After model selection, we compare the fixed policies π_R and π on matched target and source episodes. With J_e denoting expected return or success rate in environment $e \in \{\mathrm{tgt}, \mathrm{src}\}$, define

$$\Delta_{\mathrm{tgt}} = J_{\mathrm{tgt}}(\pi_R) - J_{\mathrm{tgt}}(\pi), \qquad \Delta_{\mathrm{src}} = J_{\mathrm{src}}(\pi_R) - J_{\mathrm{src}}(\pi). \tag{13}$$

For dependence reduction, both policies receive the same factual, intervened, and placebo histories from μ_{src}. With the original affected indicator $M_\pi(H_t) = \mathbf{1}\{\mathcal{R}_{\widehat{I}}(\pi; H_t) > \gamma\}$, define

$$\Delta_{\mathrm{dep}} = \mathbb{E}_{H_t \sim \mu_{\mathrm{src}}}\big[M_\pi(H_t)\big(\mathcal{R}_{\widehat{I}}(\pi; H_t) - \mathcal{R}_{\widehat{I}}(\pi_R; H_t)\big)\big]\,. \tag{14}$$

Positive Δ_{tgt} means target recovery, Δ_{src} source retention, and positive Δ_{dep} reduced response.

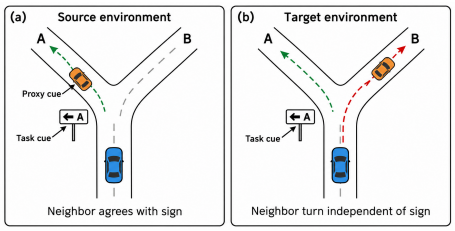
4 Experiments

In this section, we evaluate RENDER on route selection in ForkRoad, visual game control in Atari, and pedestrian crossing in CARLA, and ablate mechanism selection. Each setting changes a cue available during source training while preserving the task; repairs use source data and a frozen base policy. Every fitted arm uses five repair seeds on one frozen policy, and tabulated standard errors are over those seeds, or over evaluation episodes for the frozen policy. Diagnosis uses 30 (CARLA) to 160 (ForkRoad) target failures, split between selection and confirmation (Appendices C and D).

4.1 CONTROLLED VALIDATION IN FORKROAD

In ForkRoad, a car must read a road sign and take the matching branch at a junction. A leading car's turn agrees with the sign in the source setting and so supplies a shortcut, but is independent of it in the target setting. We induce a known dependence by randomizing the learner's sign observation during pretraining, then restore the sign for diagnosis and evaluation.

RENDER identifies the leading car's turn as the harmful cue and trains an action adaptor from source continuation outcomes. As an ablation, Randomize-all keeps every step of the repair but randomizes all ten task-preserving cues instead of the diagnosed one, on the same paired episodes. The diagnosed repair achieves nearly complete target recovery with little source loss and almost no remaining response to the cue (Figure 4), and for every seed all three joint lower bounds clear fixed margins (Appendix C.4.1). Randomize-all also passes, but recovers less and keeps a measurable response: it must learn invariance to nine irrelevant cues and covers the harmful one by chance.



Metric	Frozen	Rand.-all	RENDER
Target (%) ↑	48.6 ± 1.4	84.9 ± 1.9	$\mathbf{97.7 \pm 0.9}$
Source (%) ↑	100.0 ± 0.0	$\mathbf{100.0 \pm 0.0}$	99.7 ± 0.2
Response (%) ↓	48.8 ± 1.4	5.9 ± 1.1	$\mathbf{0.1 \pm 0.0}$

Figure 4: ForkRoad. *Left*: the sign determines the correct road; the leading car's turn agrees with it in the source domain and is independent in the target. *Right*: deployment metrics for the frozen policy, the Randomize-all ablation and RENDER, with the better adaptor in bold.

4.2 TARGETED FEATURE ADAPTORS FOR PIXEL POLICIES IN ATARI

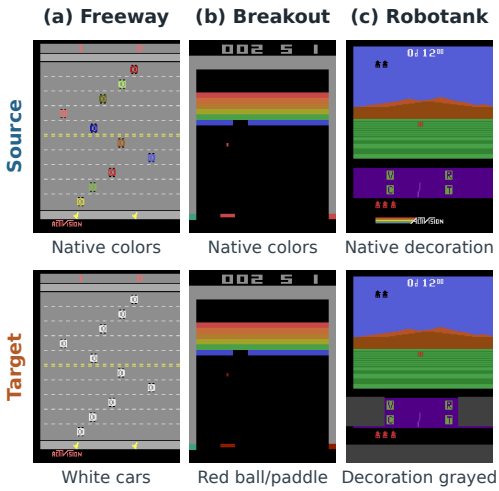
We consider three tasks that need decisions from moving scenes: crossing lanes of traffic in Freeway, keeping a ball in play in Breakout, and aiming at opponents from a tank in Robotank. The policies are trained on the games' native displays, where stable visual properties can enter their decision rules. At deployment we turn the cars white, recolor the paddle and ball red, and mask the logo strip and the two flanks of Robotank's instrument panel with a neutral gray, simulating an occluded dashboard; at matched states these changes preserve object geometry and task dynamics (Figure 5).

Freeway and Breakout use the released pixel PPO baselines of Blüml et al. (2026); Robotank uses an existing Sample Factory policy, run through the same code. We train a feature adaptor on matched native and intervened source views with the encoder and action head frozen, so deployment requires only pixels. As in ForkRoad, Randomize-all trains the same adaptor but randomizes every task-preserving cue instead of the diagnosed one. We further compare with ILA (Yoneda et al., 2022) and TARL (Xu et al., 2025), which adapt a deployed policy from unlabeled target observations, and with continuing PPO in the native game for two million decisions, with and without UCB-DrAC's generic augmentations (Raileanu et al., 2021). Every arm uses the same frozen checkpoints, seeds and evaluation episodes.

The diagnosed repair is the only arm that raises target return in all three games (Figure 5), recovering about two fifths of what the Robotank occlusion costs, and source return does not fall except for a small cost in Freeway. Randomize-all recovers less everywhere and ends below the frozen policy in Breakout. ILA and TARL recover little and do not raise target return consistently across the three games, and ILA also loses source return. Continued PPO leaves Freeway near the frozen level and deepens the dependence in Robotank; its one target gain, UCB-DrAC on Breakout, costs about twice as much source return as it gains on the target and changes half of the native greedy actions.

4.3 DRIVING UNDER ACTION HISTORY SHIFTS IN CARLA

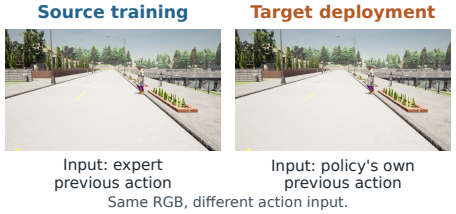
In CARLA, the policy drives a route and must yield when a pedestrian steps into the road; idle episodes (no pedestrian) test for unnecessary stopping. It selects speed and steering from a camera image and a previous action input, taken from the expert during behavior cloning; because consecutive expert commands are correlated, that input predicts the next action without reading the scene. At deployment it carries the policy's own previous command (Figure 6), which can sustain motion



Method	Source ↑	Target ↑	Response (%) ↓
Freeway, white cars			
Frozen	32.85 ± 0.08	16.85 ± 0.13	34.4
ILA	27.48 ± 0.72	17.77 ± 0.38	23.9 ± 1.1
TARL	32.61 ± 0.08	17.83 ± 0.08	31.4 ± 0.7
Native fine-tune	32.57 ± 0.11	16.46 ± 0.83	32.2 ± 1.1
UCB-DrAC fine-tune	32.42 ± 0.16	16.14 ± 0.44	31.0 ± 0.9
Randomize-all	$\mathbf{32.89 \pm 0.02}$	20.38 ± 0.88	21.1 ± 0.9
RENDER	32.22 ± 0.22	$\mathbf{26.48 \pm 0.66}$	$\mathbf{8.3 \pm 0.1}$
Breakout, red ball and paddle			
Frozen	185.40 ± 19.10	55.00 ± 16.43	27.9
ILA	54.24 ± 6.05	37.21 ± 3.60	21.4 ± 0.3
TARL	$\mathbf{209.29 \pm 9.95}$	65.55 ± 6.61	31.2 ± 0.8
Native fine-tune	171.30 ± 11.50	42.99 ± 12.15	27.5 ± 1.7
UCB-DrAC fine-tune	131.13 ± 24.42	83.44 ± 12.87	19.7 ± 1.5
Randomize-all	191.10 ± 9.42	41.61 ± 16.91	26.4 ± 0.9
RENDER	200.08 ± 4.02	$\mathbf{125.18 \pm 6.67}$	$\mathbf{18.4 \pm 0.4}$
Robotank, occluded instrument panel			
Frozen	66.80 ± 1.11	19.70 ± 1.19	59.9
ILA	50.32 ± 2.94	17.78 ± 0.64	$\mathbf{27.5 \pm 1.0}$
TARL	67.35 ± 0.58	17.11 ± 0.68	50.7 ± 1.1
Native fine-tune	63.78 ± 0.97	9.26 ± 1.05	81.7 ± 4.1
UCB-DrAC fine-tune	65.72 ± 0.42	14.06 ± 3.10	80.1 ± 2.6
Randomize-all	$\mathbf{67.85 \pm 0.30}$	25.05 ± 2.48	42.8 ± 1.3
RENDER	67.63 ± 0.70	$\mathbf{38.12 \pm 3.96}$	31.6 ± 3.9

Figure 5: Atari. *Left*: source and target views at matched task states. The shifts recolor the cars and the paddle and ball, and gray the instrument panel's logo strip and flanks, leaving the radar, damage indicators and reserve-tank count visible; the Robotank mask acts on the 84×84 policy input and is drawn on the RGB frame for display. *Right*: returns in both domains and action response, how often the native and shifted views give different greedy actions on held-out source states. Shading marks our method and bold the best value within a game; the frozen base is unmarked, and its response, measured at fixed states, has no standard error.

as a pedestrian enters the road. RENDER selects the previous action family from target failures and confirms its harm (Appendix C.2.3); a runtime adaptor then learns from source collisions.



Controller	Crossing (%) ↑	Idle (%) ↑
Frozen	50.0 ± 4.5	100.0 ± 0.0
Equal weights	50.8 ± 4.5	$\mathbf{100.0 \pm 0.0}$
RENDER	$\mathbf{89.7 \pm 0.7}$	97.6 ± 1.0
Oracle gate	89.7 ± 2.7	100.0 ± 0.0

Figure 6: CARLA previous action inputs and deployment success. *Left*: the same RGB frame shows the change from the expert's previous action in training to the policy's own action at deployment. *Right*: success on crossing and idle episodes, with our method shaded and bold marking the better of Equal weights and RENDER; the oracle gate reads privileged state and is a reference. The two controls use the retained predictor of repair seed 0, and their errors are over episodes.

RENDER improves crossing success while largely retaining idle completion (Figure 6), for all five repair seeds on every evaluation seed. Two controls share the episodes and the seed 0 predictor: equal weights keeps our gate, and the oracle gate keeps our scores. With equal weights ($\beta = 0$) the vote over the two cue views reduces to the frozen policy's own preference and crossing success stays at the frozen level, so the collision-outcome scores are what change the decision. The oracle gate replaces our runtime gate with simulator knowledge of whether a pedestrian is crossing ahead; our gate matches that reference on crossing episodes, and its only idle failures are stalls, most frequent for the seed whose predictor ranks collisions best.

4.4 Ablation Study: Why Diagnosis Needs Target Failures

In this ablation study, we test whether target failures are needed to decide which dependence to repair. A policy can carry several harmful dependencies, and a source audit reveals all of them without indicating which one a deployment triggered. We therefore deploy one policy under shifts that trigger different dependencies, plus a shift outside the library, and compare RENDER with *Harm-only*, which skips Steps 1 and 2 and selects the candidate with the largest lower bound on source harm among those passing Step 3, with *Return-only*, which runs Steps 1 and 2 on the returns

of target failures alone, and with *Oracle*, given the true mechanism. All rules keep RENDER's Step 3 audit and share five diagnosis seed blocks.

Table 1: Mechanism selection on three Breakout targets over five diagnosis seed blocks. Counts give the blocks in which a rule selected the true mechanism, another one, or abstained, which deploys the frozen policy. Gains over the frozen policy are block averages with the standard error over five repair seeds.

Target	Rule	True	Other	Abstain	Target gain ↑	Source gain ↑
T1 red paddle	Harm-only	0	5	0	-20.1 ± 2.7	-1.9 ± 5.0
	Return-only	1	0	4	$+14.0 \pm 1.3$	$+2.9 \pm 0.8$
	RENDER	**5**	0	0	$\mathbf{+70.2 \pm 6.7}$	$\mathbf{+14.7 \pm 4.0}$
	Oracle	5	0	0	$+70.2 \pm 6.7$	$+14.7 \pm 4.0$
T2 white bricks	Harm-only	0	5	0	-8.0 ± 10.7	-1.9 ± 5.0
	Return-only	4	0	1	$+48.8 \pm 6.0$	$+3.1 \pm 7.8$
	RENDER	**5**	0	0	$\mathbf{+61.0 \pm 7.5}$	$\mathbf{+3.9 \pm 9.8}$
	Oracle	5	0	0	$+61.0 \pm 7.5$	$+3.9 \pm 9.8$
T3 action delay	Harm-only	–	5	0	-0.1 ± 0.4	-1.9 ± 5.0
	Return-only	–	0	**5**	**0.0**	**0.0**
	RENDER	–	0	**5**	**0.0**	**0.0**
	Oracle	–	0	5	0.0	0.0

The Breakout policy of Section 4.2 is harmed both by a red paddle and ball (T1) and by white bricks (T2), and neither repair transfers to the other shift (Appendix D.6); T3 delays each action by one decision, which no candidate reproduces, so abstaining is correct. RENDER matches Oracle in every case (Table 1). Oracle knows the true mechanism and applies the same Step 3 audit and repair, so a few target failures lose nothing against knowing the cause, and the remaining gap to source return comes from the repair rather than the diagnosis. Harm-only never finds the true mechanism, because the source audit ranks a gray background and a blue paddle above both, and its repairs do not raise target return. Return-only accepts no wrong mechanism but mostly abstains on T1, where the red paddle and dimming leave nearly the same failure returns. Source harm shows which dependencies a policy has; how it fails on the target shows which one the deployment triggered.

Appendix D.7 reports a much harder ablation study in Robotank: it tests whether RENDER still identifies the true mechanism, or abstains outside the library, when candidates from different families fail as often, with similar returns and source harm. RENDER again matches Oracle.

Beyond these ablations, repeating Steps 1 to 3 on five seed blocks per target in every benchmark confirms the true mechanism in 27 of 30 runs and never accepts a wrong one (Appendix D.5).

5 Discussion and Conclusion

Our results suggest that deployment failure should not be seen only as a generic distribution shift: a target failure reveals that the policy no longer behaves appropriately but not which changed factor caused this, and strong source performance does not rule out harmful nuisance dependence. Source records alone cannot settle this question, but our experiments show that a few unpaired target failures can guide source interventions that do. RENDER therefore separates failure matching from causal verification and treats diagnosis and repair as separate claims, since identifying a harmful dependence does not guarantee that a local adaptor can remove it without harming useful behavior. Hence we evaluate repair by target recovery, source preservation, and reduced cue dependence.

The main limitation is intervention coverage. RENDER can verify harmful dependence only for mechanisms that can be reenacted and tested in the source environment. Its conclusions also depend on whether source reenactment captures the mechanisms relevant to the target failure. Current experiments mainly consider individual, interpretable nuisance factors, whereas real deployments may involve unseen or interacting mechanisms. Extending the framework toward automatically generated and compositional interventions is therefore a natural direction for future work.

AI USE STATEMENT

We used generative AI tools to polish the manuscript text and to write, execute, and debug the experiment code. The authors reviewed all AI assisted code and text, checked every reported result against the saved experimental outputs, and take responsibility for the final content and claims.

ETHICS STATEMENT

The work uses simulated driving and game environments and does not involve human subjects or personal data. The proposed diagnosis and repair methods are research prototypes and should not be interpreted as safety certification for real-world autonomous systems.

REPRODUCIBILITY STATEMENT

Appendix B specifies the diagnosis and repair procedures and how every decision threshold is set. Appendix C gives the environments, frozen policies, intervention libraries and task audits, data collection, training settings, seed ranges, and statistical protocol. Appendix D reports results across the repair seeds and the diagnosis seed blocks, including the negative trials.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. CUPID: curating data your robot loves with influence functions. *CoRR*, abs/2506.19121, 2025.

Ezra Ameperosa, Jeremy A. Collins, Mrinal Jain, and Animesh Garg. Rocoda: Counterfactual data augmentation for data-efficient robot learning from demonstrations. In *ICRA*, pp. 13250–13256. IEEE, 2025.

Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *CoRR*, abs/1907.02893, 2019.

Jannis Blüml, Cedric Derstroff, Bjarne Gregori, Elisabeth Dillies, Quentin Delfosse, and Kristian Kersting. Do object channels improve robustness in deep reinforcement learning? *Trans. Mach. Learn. Res.*, 2026, 2026.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *NeurIPS*, pp. 11693–11704, 2019.

Quentin Delfosse, Sebastian Sztwiertnia, Mark Rothermel, Wolfgang Stammer, and Kristian Kersting. Interpretable concept bottlenecks to align reinforcement learning agents. In *NeurIPS*, 2024.

Lauro Langosco di Langosco, Jack Koch, Lee D. Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12004–12019. PMLR, 2022.

Thomas G. Dietterich, George Trimponias, and Zhitang Chen. Discovering and removing exogenous state variables and rewards for reinforcement learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1261–1269. PMLR, 2018.

Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provable RL with exogenous distractors via multistep inverse dynamics. *CoRR*, abs/2110.08847, 2021.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi S. Jaakkola. Learning task informed abstractions. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3480–3491. PMLR, 2021.

Jasmina Gajcin and Ivana Dusparic. Reccover: Detecting causal confusion for explainable reinforcement learning. In *EXTRAAMAS@AAMAS*, volume 13283 of *Lecture Notes in Computer Science*, pp. 38–56. Springer, 2022.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nat. Mach. Intell.*, 2(11):665–673, 2020.

Gunshi Gupta, Tim G. J. Rudner, Rowan Thomas McAllister, Adrien Gaidon, and Yarin Gal. Can active sampling reduce causal confusion in offline reinforcement learning? In *CLeaR*, volume 213 of *Proceedings of Machine Learning Research*, pp. 386–407. PMLR, 2023.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR (Poster)*. OpenReview.net, 2019.

Daniel Kumor, Junzhe Zhang, and Elias Bareinboim. Sequential causal imitation learning with unobserved confounders. In *NeurIPS*, pp. 14669–14680, 2021.

Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *NIPS*, pp. 4066–4076, 2017.

Shuaijun Liu, Feiyang You, Chengyu Wu, Shuyang Hao, Chenglong Zhang, Jingyao Cai, Xingwei Chen, and Ningxin Su. Beyond appearance shifts: Task-semantic action calibration for vla models, 2026. URL https://arxiv.org/abs/2609.23650.

Jongjin Park, Younggyo Seo, Chang Liu, Li Zhao, Tao Qin, Jinwoo Shin, and Tie-Yan Liu. Object-aware regularization for addressing causal confusion in imitation learning. In *NeurIPS*, pp. 3029–3042, 2021.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009. ISBN 9780521749190. URL https://books.google.co.id/books?id=6blRkgEACAAJ.

Andi Peng, Aviv Netanyahu, Mark K. Ho, Tianmin Shu, Andreea Bobu, Julie Shah, and Pulkit Agrawal. Diagnosis, feedback, adaptation: A human-in-the-loop framework for test-time policy adaptation. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27630–27641. PMLR, 2023.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: Identification and confidence intervals. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 78(5):947–1012, 2016. doi: 10.1111/rssb.12167.

Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. In *NeurIPS*, pp. 5402–5415, 2021.

Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with checklist. In *ACL*, pp. 4902–4912. Association for Computational Linguistics, 2020.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 15 of *JMLR Proceedings*, pp. 627–635. JMLR.org, 2011.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *CoRR*, abs/1911.08731, 2019.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proc. IEEE*, 109 (5):612–634, 2021.

Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals. *CoRR*, abs/2210.01790, 2022.

Miguel Suau, Matthijs T. J. Spaan, and Frans A Oliehoek. Bad habits: Policy confounding and out-of-trajectory generalization in RL. *Reinforcement Learning Journal*, 4:1711–1732, 2024.

Raphael Suter, Đorđe Miladinovic, Bernhard Schölkopf, and Stefan Bauer. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6056–6065. PMLR, 2019.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pp. 23–30. IEEE, 2017.

Victor Veitch, Alexander D'Amour, Steve Yadlowsky, and Jacob Eisenstein. Counterfactual invariance to spurious correlations in text classification. In *NeurIPS*, pp. 16196–16208, 2021.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. In *NeurIPS*, 2020.

Youguang Xing, Xu Luo, Junlin Xie, Lianli Gao, Hengtao Shen, and Jingkuan Song. Shortcut learning in generalist robot policies: The role of dataset diversity and fragmentation. *CoRR*, abs/2508.06426, 2025.

Shoukai Xu, Zihao Lian, Mingkui Tan, Liu Liu, Zhong Zhang, and Peilin Zhao. Test-time adapted reinforcement learning with action entropy regularization. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Takuma Yoneda, Ge Yang, Matthew R. Walter, and Bradly C. Stadie. Invariance through latent alignment. In *Robotics: Science and Systems*, 2022.

Junzhe Zhang, Daniel Kumor, and Elias Bareinboim. Causal imitation learning with unobserved confounders. In *NeurIPS*, 2020.

Appendices of
Real-World Debiasing: From Deployment Failures to Causal Repair of Policy Shortcuts

TABLE OF CONTENTS

A Theoretical analysis and proofs **14**
- A.1 Identifiability of excess task loss under intervention . . . 14
- A.2 Deployment guarantee . . . 15

B Method details **15**
- B.1 Observable failure signatures . . . 16
- B.2 Task preservation . . . 16
- B.3 Reenactment selection and confirmation . . . 16
- B.4 Paired response and harm . . . 17
- B.5 Outcome scoring and decision composition . . . 17
- B.6 Gated action residual . . . 18
- B.7 Feature correction and training objective . . . 18
- B.8 Runtime decision adaptor . . . 19

C Experimental setup **19**
- C.1 Environments and frozen policies . . . 19
- C.2 Intervention libraries and task audits . . . 20
- C.3 Data collection and training . . . 22
- C.4 Evaluation protocols and statistical reporting . . . 24

D Main experiment results and analysis **25**
- D.1 Outcome label audit . . . 25
- D.2 Return differences and success intervals . . . 25
- D.3 Published adaptation baselines on Atari . . . 27
- D.4 Behavioral confirmation . . . 27
- D.5 Diagnosis over five seed blocks . . . 29
- D.6 Target failures select the deployed mechanism . . . 30
- D.7 Equally harmful candidates in Robotank . . . 31

E Related work **32**

A THEORETICAL ANALYSIS AND PROOFS

A.1 IDENTIFIABILITY OF EXCESS TASK LOSS UNDER INTERVENTION

Identification criterion. For a fixed policy π, intervention I, and history distribution μ, the query is $\mathcal{B}_I^M(\pi;\mu)$ in an admissible causal model M. Identification from the source distribution requires every pair of models that agree on that distribution to agree on the query (Pearl, 2009). Proposition 1 shows that the task assumptions alone do not impose this agreement. Proposition 2 gives a sufficient form of interventional access.

Proof of Proposition 1. Consider a one-decision task with a finite state space $\mathcal{S}$ containing at least two elements. Let $S \sim \nu$, where ν has full support, and let $\ell(s,a) \geq 0$ be a bounded task loss. Choose an optimal rule g with $\ell(s,g(s)) = 0$ for every s, and states $s \neq s'$ such that

$$\ell(s, g(s')) > 0. \tag{15}$$

Both models use reference action $g(s)$, reward $r(s,a) = -\ell(s,a)$, and the same transition to a terminal state under every action. Let $\mathcal{U} = \mathcal{S}$ and impose $U = S$ almost surely in the source. Observations have two coordinates, $O = (s,c)$, with mechanisms

$$f_0(s,u) = (s,s), \qquad f_1(s,u) = (s,u). \tag{16}$$

Fix the same deterministic policy in both models: $\pi(s,c) = g(c)$ for every input, including inputs absent from the source. Since $U = S$ in the source, both mechanisms produce (S,S) and the policy incurs zero loss. The models therefore agree on the source law of states, nuisances, observations, actions, rewards, reference actions, and terminations, under any collection protocol that does not intervene on the nuisance.

Now apply $I = \mathrm{do}(U = s')$, with s' from Equation 15, and use the common source history law μ. The value s' has positive source probability. The intervention leaves the task state, reference action, reward, and terminal transition unchanged. The first observation coordinate still reveals S in both models, so the intervention also preserves the information needed to solve the task, and it is task-preserving in both models. In M_0 the policy still chooses $g(S)$. In M_1 it chooses $g(s')$. Therefore,

$$\mathcal{B}_I^{M_0}(\pi;\mu) = 0, \qquad \mathcal{B}_I^{M_1}(\pi;\mu) = \mathbb{E}_{S\sim\nu}\big[\ell(S, g(s'))\big] \geq \nu(s)\,\ell(s, g(s')) > 0. \tag{17}$$

The same source distribution thus corresponds to different excess losses, and even the distinction between zero and positive excess loss is not identified. □

The missing information concerns the observation mechanism under intervention. The policy is already specified on every input. Although every nuisance value occurs in the source, the combinations $u \neq s$ do not: source observations are (s,s), whereas intervention can produce (s,s'). More data from this source distribution cannot distinguish the two mechanisms.

A special case shows the role of support. In a finite one-decision task, suppose a source optimal policy is evaluated on an intervened observation that has positive source probability, and a common decoder recovers its task state in both domains. That observation then corresponds to the same state in the source, where the policy already selects an optimal action, so the intervention adds no loss there. ForkRoad instantiates the perfectly correlated source setting: the neighboring turn predicts the correct action in the source, and the intervention breaks that association.

Proof of Proposition 2. Fix π, I, and μ. Assume that the simulator samples histories and their states with the intended joint law, and generates factual and intervened losses with conditional means $\mathbb{E}[Y^0 \mid H_t] = \mathcal{L}_{\mathrm{task}}(\pi; H_t)$ and $\mathbb{E}[Y^I \mid H_t] = \mathcal{L}^I_{\mathrm{task}}(\pi; H_t)$. The tower property and Definition 2 give

$$\mathbb{E}[Y^I - Y^0] = \mathbb{E}_{H_t\sim\mu}\big[\mathcal{L}^I_{\mathrm{task}}(\pi; H_t) - \mathcal{L}_{\mathrm{task}}(\pi; H_t)\big] = \mathcal{B}_I(\pi;\mu). \tag{18}$$

The left side depends only on the laws of Y^I and Y^0, so the two rollout marginals identify the query. Their coupling affects estimation variance but not identification.

Let $(Y_i^I, Y_i^0)_{i=1}^n$ be i.i.d. pairs from a fixed coupling with these marginals, and let $D_i = Y_i^I - Y_i^0$. If both losses have finite variance, $\widehat{\mathcal{B}}_I = n^{-1}\sum_i D_i$ is unbiased and converges almost surely to $\mathcal{B}_I(\pi;\mu)$ by the strong law of large numbers. Its variance is

$$\mathrm{Var}(\widehat{\mathcal{B}}_I) = \frac{\mathrm{Var}(Y^I) + \mathrm{Var}(Y^0) - 2\,\mathrm{Cov}(Y^I, Y^0)}{n}.$$

Relative to two independent sample means with n rollouts each, pairing changes the variance by $-2\,\mathrm{Cov}(Y^I, Y^0)/n$, so it lowers the variance exactly when this covariance is positive. □

Connection to the causal audit. Step 3 uses the paired mean with $\mu = \mu^+_{\mathrm{src}}$, the distribution of first affected source histories. The selected intervention $\widehat{I}$ and the threshold γ are fixed before Step 3 collects its episodes from a disjoint seed block. Each affected trajectory supplies one history, so the pairs are independent, and the branching decision uses only the history up to that decision, so each continuation keeps its conditional law given H_t. ForkRoad and Atari branch by restoring simulator snapshots. CARLA reaches the branch point by replaying the recorded controls, so its estimate is the harm under this replay protocol.

A.2 Deployment guarantee

A joint deployment certificate combines target recovery, source preservation, and reduced intervention response. Let π_R be the repaired policy of Section 3.5, and let J_{tgt} and J_{src} denote expected task performance on the prespecified evaluation distributions. The source audit fixes one decision per trajectory using the original policy and keeps the indicator $M_\pi(H_t) = \mathbf{1}\{\mathcal{R}_{\widehat{I}}(\pi; H_t) > \gamma\}$ unchanged for the repaired policy. The application sets three margins: ϵ_{tgt} is the minimum target gain, ϵ_{src} is the largest allowed source loss, and ϵ_{dep} is the minimum reduction in average response. The deployment conditions are

$$J_{\mathrm{tgt}}(\pi_R) - J_{\mathrm{tgt}}(\pi) \geq \epsilon_{\mathrm{tgt}}, \tag{19}$$

$$J_{\mathrm{src}}(\pi_R) - J_{\mathrm{src}}(\pi) \geq -\epsilon_{\mathrm{src}}, \tag{20}$$

$$\mathbb{E}_{H_t\sim\mu_{\mathrm{src}}}\big[M_\pi(H_t)\big(\mathcal{R}_{\widehat{I}}(\pi; H_t) - \mathcal{R}_{\widehat{I}}(\pi_R; H_t)\big)\big] \geq \epsilon_{\mathrm{dep}}. \tag{21}$$

Unaffected trajectories contribute zero to the last expectation, which retains the prevalence of the diagnosed response in the evaluation population. Its sample form is Equation 25.

Proposition 3 (Selective deployment guarantee)**.** *Let $\delta \in (0,1)$. Suppose the lower confidence bounds for the three quantities in Equations 19 to 21 have simultaneous coverage at least $1-\delta$ over all claims and candidates considered for deployment, and a candidate is deployed only when all three bounds reach their margins. Then, with probability at least $1-\delta$, every deployed candidate satisfies all three conditions.*

Proof of Proposition 3. Let $\mathcal{E}$ be the simultaneous coverage event. On $\mathcal{E}$, every true quantity is at least its lower confidence bound. Every deployed candidate has all three bounds at or above their margins, so all three true quantities satisfy the corresponding conditions. Since $\Pr(\mathcal{E}) \geq 1-\delta$, the claim follows. □

The experiments obtain the lower bounds by paired bootstrap. ForkRoad evaluates the three conditions jointly; Atari reports return differences and action response, and CARLA reports deployment success.

B Method details

This appendix specifies failure matching, the causal audit, and the three repair implementations of Section 3. Appendix C gives their data and training protocols, Appendix C.4 the statistical conventions, and Appendix D the results.

B.1 OBSERVABLE FAILURE SIGNATURES

For a trajectory $\tau = (O_t, A_t, R_t)_{t=1}^{T}$, let $G = \sum_t R_t$ and $h_k = T^{-1} \sum_t \mathbf{1}\{A_t = k\}$ for K action categories. Given nonnegative event weights w_t fixed by the application, define

$$q_p(\tau) = \min \left\{ \frac{t}{T} : \sum_{j=1}^{t} w_j \geq p \sum_{j=1}^{T} w_j \right\}, \qquad p \in \{0.25, 0.50, 0.75, 1\}. \tag{22}$$

All four times are set to one when no event occurs. With prespecified observable event indicators e, such as collision or stall, the signature is

$$\phi(\tau) = (G, T, q_{0.25}, q_{0.50}, q_{0.75}, q_1, h_1, \ldots, h_K, e). \tag{23}$$

ForkRoad sets $w_t = 1$ when the reward departs from its ordinary step cost. CARLA has no reward, so it omits the return and sets $w_t = 1$ at each commanded deceleration. Both use Equation 22. Atari sets $w_t = \mathrm{round}([R_t]_+)$. Its q_1 follows Equation 22, while for $p < 1$ it takes the linearly interpolated p quantile of the list in which each event time t/T appears w_t times. Continuous actions are summarized by fixed bins or moments. Target failures and source reenactments use the same extractor, with no hidden target state, and coordinate scales are estimated from source rollouts.

B.2 TASK PRESERVATION

An admitted intervention must preserve the task state, the reference decision, the executed control, the reward rule, and the task transition under the same action. It must also leave the observation information needed for task decisions. Information preservation is checked separately, because an occlusion can leave rewards and transitions unchanged while removing a necessary visual cue.

For an isolated observation or input write, the check examines the implementation and the content that remains available to the policy. Other interventions use paired source queries that restore the same simulator and random state, execute the same action, and compare the task state, reference decision, reward, and next task state. Stochastic comparisons use matched seeds. State coverage, discrepancy measures, tolerances, and sample sizes are fixed before target matching, and every discrepancy must have a simultaneous one sided upper confidence bound below its tolerance. Candidates that fail the audit or lack sufficient coverage are removed before mechanism selection. Appendix C.2 gives the checks for each benchmark; CARLA's world candidates receive only a closed-form check of the pedestrian conflict.

B.3 REENACTMENT SELECTION AND CONFIRMATION

A fixed failure rule determines which episodes enter signature matching. Target failures and source candidate rollouts are divided into selection and confirmation splits before ranking. Matching uses failed episodes, while failure recurrence uses all source episodes for a candidate. Each split must contain the prespecified minimum number of failures.

For split $q \in \{\mathrm{sel}, \mathrm{conf}\}$, let X_q, X'_q be independent draws from the empirical target failure signature distribution, and $Y_{I,q}, Y'_{I,q}$ independent draws from the empirical source failure signature distribution under I. These distributions retain sample frequencies, including repeated signatures. Pooled source selection rollouts determine coordinate scales σ with fixed positive floors. Writing $\oslash$ for coordinatewise division, the scaled energy distance D_ϕ on split q is

$$\begin{aligned} d_q(I) = 2\mathbb{E} \left\| (X_q - Y_{I,q}) \oslash \sigma \right\|_2 - \mathbb{E} \left\| (X_q - X'_q) \oslash \sigma \right\|_2 \\ - \mathbb{E} \left\| (Y_{I,q} - Y'_{I,q}) \oslash \sigma \right\|_2 . \end{aligned} \tag{24}$$

Selection first retains the smallest-distance eligible representative in each family and then selects the best family. The representatives and the winning family stay fixed during confirmation. Acceptance requires four checks: the one sided 95% Wilson lower bound on failure recurrence reaches $r_{\mathrm{reen}}(c)$, the confirmation distance is at most $d_{\max}(c)$, where c is the selected candidate, the bootstrap support reaches $b_{\min}$, and the selected distance divided by the smallest alternative-family distance is at most $\rho_{\max}$. A representative with too few confirmation failures receives infinite distance. If no alternative has finite distance, the separation ratio is set to zero, and acceptance then rests on absolute fit and

recurrence. Bootstrap support is the fraction of resamples in which the frozen winning family keeps the smallest distance; each resample draws target and source confirmation failures and compares the frozen representatives.

Source calibration precedes target matching. Each library candidate that produces enough source failures serves in turn as a matched pseudo target whose correct family is known. A matched trial is usable when it selects its own family with sufficient stability and separation. For each usable candidate c, the distance and recurrence thresholds are set so that a correct match with c fails either check with probability at most 5%. Two disjoint subsamples of c's calibration failures, one of the target split size and one of the size of the library batch the target is compared with, give the distance that a correct match reaches by sampling alone; $d_{\max}(c)$ is its 95% quantile. For recurrence, c's calibration failure count is resampled and then a library batch is drawn at the resampled rate; $r_{\text{reen}}(c)$ is the 5% quantile of the resulting lower bound. A candidate without a usable matched trial has no thresholds and cannot be accepted. Negative trials use held out mechanisms outside the library, and split nulls use two different held out mechanisms for their two splits; both test that the complete rule rejects answers the library does not contain, and calibration fails if it accepts any of them. All benchmarks use $b_{\min} = 0.5$ and $\rho_{\max} = 0.8$.

Confirmation only accepts or rejects the frozen selection, and a failed check returns `none_of_the_above`. An accepted reenactment is not yet evidence of real-world bias; the response and harm audit supplies it. Appendix C.2 reports the calibrated values and outcomes for each benchmark.

B.4 Paired response and harm

All three benchmarks audit the greedy action, so the distributions in Equation 7 are point masses and d_A, their total variation, is 1 when the greedy action changes and 0 otherwise. The placebo I_0 reconstructs the factual input through the same intervention route and measures replay error. Let $H_{i,t}$ be the history at time t in trajectory i, let $E_i(\pi) = \max_t \mathcal{R}_{\widehat{I}}(\pi; H_{i,t})$, and let $t_i^*(\pi) = \arg\max_t \mathcal{R}_{\widehat{I}}(\pi; H_{i,t})$. A trajectory is affected when $E_i(\pi) > \gamma$. The threshold γ is the 95% quantile of the same maximum statistic over source placebo trials, calibrated separately for each intervention route (Appendix C.2.3). Source intervention calibration also sets each candidate's minimum affected prevalence: the candidate's affected count on the calibration histories is resampled, an audit of the audit's size is drawn at the resampled rate, and the threshold is the 5% quantile of the audit's prevalence lower bound. The audit compares the selected candidate with its own threshold. The prevalence and harm lower bounds split the 5% error by Bonferroni allocation, so each uses the one sided level 0.975.

For harm estimation, each affected trajectory contributes its history at the first affected decision $t_i^+(\pi) = \min\{t : \mathcal{R}_{\widehat{I}}(\pi; H_{i,t}) > \gamma\}$ to μ_{src}^+. Paired continuations branch there, apply the selected intervention from that decision on, follow their own termination conditions, and stop at the application horizon. The lower bound on mean excess task loss must exceed zero or the application's larger loss margin.

Repair evaluation is a separate measurement. It compares both policies at the original policy's maximum-response decision. For n fresh evaluation trajectories, the dependence reduction statistic is

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{E_i(\pi) > \gamma\} \left[\mathcal{R}_{\widehat{I}}(\pi; H_{i,t_i^*(\pi)}) - \mathcal{R}_{\widehat{I}}(\pi_R; H_{i,t_i^*(\pi)})\right]. \tag{25}$$

Both policies receive the same factual and intervened histories at this decision, and unaffected trajectories contribute zero. The first affected decision branches the causal continuations; the maximum-response decision serves the fixed-history comparison of the two policies.

B.5 Outcome scoring and decision composition

For an admitted view v, let $p_v(a \mid H_t)$ be the frozen policy's candidate action distribution. ForkRoad uses a point mass on the greedy action; CARLA uses the full speed distribution. An outcome predictor with parameters ψ, fitted to executed source outcomes, supplies a score $s_\psi(H_t, v)$, with larger values favoring better predicted outcomes. It is trained by supervised regression or classification without a policy consistency penalty. The application fixes the score scale and $\beta \geq 0$ for the

weighted vote in Equation 12, and equal weighting is $\beta = 0$. ForkRoad turns the vote into audited labels for a separate adaptor with parameters φ. CARLA applies the weighted distribution inside a fixed runtime gate, with $\varphi = \psi$.

B.5.1 ACTION CONTINUATION LABELS

For ForkRoad source history i, taken at the original policy's maximum-response decision, let H_i^v set the neighboring-turn input to view v, and let $a_{i,v} = \arg\max_a \pi_\theta(a \mid H_i^v)$. For each distinct candidate action, we restore the same source snapshot, execute that action once, and follow the frozen policy to the end of the episode. The undiscounted continuation return supplies the label. The return predictor $Q_\psi(x_i, a)$ receives an action one-hot vector and the observed state x_i with the diagnosed coordinate set to zero, so its score depends on the task context and the candidate action rather than on the neighboring-turn value.

The predictor fits standardized returns (Appendix C.3.1), and its predictions are converted back to return units for voting. Within a state, candidate scores are centered and divided by their standard deviation, with a floor of 10^{-3}; if their range is at most 10^{-3}, all scores are set to zero. ForkRoad uses $\beta = 2$. Let a_i^* be the action with the largest vote; the label is retained only when its vote mass is at least 0.9. Appendix D.1 reports the independent label audit.

B.6 GATED ACTION RESIDUAL

Each retained state supplies the same label distribution for every admitted view:

$$q_i(a) = \mathbf{1}\{a = a_i^*\}. \tag{26}$$

Let $\ell_\theta(H)$ be the frozen DQN value vector divided by the fixed temperature. A learned gate $g_\varphi(H) \in [0, 1]$ multiplies an action residual $r_\varphi(H)$:

$$\widetilde{p}_{\theta,\varphi}(a \mid H) = \mathrm{softmax}[\ell_\theta(H) + g_\varphi(H) r_\varphi(H)]_a \,. \tag{27}$$

Execution is greedy. The gate and residual receive the same scaled observation as the base policy and have hidden widths 32 and 64. The residual output layer starts at zero, so initialization preserves the base scores. This learned gate is distinct from CARLA's fixed runtime disagreement gate.

Let $\mathcal{D}_{\mathrm{src}}^{\mathrm{safe}}$ contain source observations at which the original action does not respond to the admitted nuisance values, and let $p_\theta = \mathrm{softmax}(\ell_\theta)$. With $|\mathcal{A}|$ actions, training minimizes

$$\begin{aligned} \mathcal{L}_{\mathrm{act}}(\varphi) = \mathbb{E}_{i,v}\left[-\log \widetilde{p}_{\theta,\varphi}(a_i^* \mid H_i^v) + \frac{\lambda_{\mathrm{res}}}{|\mathcal{A}|} \|r_\varphi(H_i^v)\|_2^2 \right] \\ + \lambda_{\mathrm{src}} \mathbb{E}_{H \sim \mathcal{D}_{\mathrm{src}}^{\mathrm{safe}}} D_{\mathrm{KL}}(p_\theta(\cdot \mid H) \,\|\, \widetilde{p}_{\theta,\varphi}(\cdot \mid H)) \,. \end{aligned} \tag{28}$$

The common label encourages agreement across nuisance views, the KL term preserves the original action distribution on unaffected source observations, and the residual penalty limits the correction. The base policy and the outcome predictor stay frozen during adaptor fitting. Execution requires only the base policy, the gate, and the residual.

B.7 FEATURE CORRECTION AND TRAINING OBJECTIVE

For Atari, the frozen encoder e_θ produces 512 dimensional native and intervened features $z^0 = e_\theta(H_t)$ and $z^I = e_\theta(H_t^I)$ from matched source states and action histories. Let μ and σ be the native training feature mean and standard deviation, with each entry of σ at least 0.1. Writing $\odot$ for coordinatewise multiplication, all three games use

$$\mathcal{A}_\varphi(z) = \mathrm{ReLU}\big[z + \sigma \odot r_\varphi^{\mathrm{net}}((z - \mu) \oslash \sigma)\big] \,. \tag{29}$$

The residual network r_φ^{net} has widths $512, 64, 512$, with a ReLU between its linear layers and a zero-initialized output layer. Since the encoder output is nonnegative, initialization preserves features and logits exactly. The complete correction in Equation 10 is $r_\varphi(z) = \mathcal{A}_\varphi(z) - z$. The encoder, action head, and value head are frozen; the adaptor has $66{,}112$ trainable parameters.

With $d_\sigma(u, v) = \|(u - v) \oslash \sigma\|_2^2/512$, the objective is

$$\mathcal{L}_{\text{feat}}(\varphi) = \mathbb{E}_{(z^0, z^I)}\Big[4d_\sigma(\mathcal{A}_\varphi(z^0), z^0) + d_\sigma(\mathcal{A}_\varphi(z^I), z^0) + \frac{10^{-4}}{2} \sum_{v \in \{0,I\}} d_\sigma(\mathcal{A}_\varphi(z^v), z^v)\Big]. \tag{30}$$

The three terms preserve native features, align intervened features with their native counterparts, and limit the correction; this is Equation 11 with $\lambda_{\text{src}} = 4$, distances scaled by σ, and a small penalty on the correction. Supervision comes from the original encoder on the matched native history. Object geometry and action labels do not enter the loss.

B.8 Runtime decision adaptor

CARLA queries the frozen policy on the same current RGB image with previous speed $v \in \{0, 6\}$ and previous steering zero. For feature z_v and collision logit $h_\psi(z_v)$, Equation 12 uses $s_\psi(H_t, v) = -h_\psi(z_v)$, $\beta = 1$, and the full speed distributions. The predictor is the only trained repair component, so $\varphi = \psi$; Appendix C.3.3 specifies its labels and training.

The fixed gate fires when the two views give different greedy speed bins and their feature differs from the factual query by more than 10^{-6}. Inside the gate, the vote selects the greedy speed bin; outside it, the factual speed is kept. Steering always comes from the factual query, and the next cue records the executed action. Logit scores keep the separation that bounded success probabilities would compress. Equal weighting sets $\beta = 0$. The deployed adaptor uses no simulator state.

C Experimental setup

The following protocols specify the environments, intervention audits, training data, and evaluation procedures for Section 4. Each task uses one frozen base checkpoint, and the five repair seeds vary repair construction and fitting as specified below.

C.1 Environments and frozen policies

C.1.1 ForkRoad

The corridor has length $\Delta = 4$ and source correlation $\rho = 1$. The greedy DQN has two hidden layers of width 128 and is trained for 2,000 source episodes with seed 0. The visible sign is randomized during training, so the neighboring turn is the reliable cue; the sign is valid during diagnosis and evaluation. This controlled construction supplies a known shortcut for validating diagnosis and repair. Failure means no positive terminal reward. Signatures contain return, length, four event times, three action frequencies, and indicators for a goal reward and a barrier penalty. Target records contain no simulator snapshots or corrective actions.

C.1.2 Atari

Freeway and Breakout use the released pixel PPO checkpoints of Blüml et al. (2026). Their first convolution receives four grayscale frames of size 84×84. Robotank uses an existing Sample Factory policy with the same 512 dimensional feature interface.

C.1.3 CARLA

CARLA 0.9.15 runs Town01_Opt in synchronous mode. The camera supplies 256×144 RGB images with a 70 degree field of view. The vehicle starts 60 m before a crossing and completes the episode 15 m beyond it. The pedestrian walks at 1.5 m/s and starts crossing when the vehicle reaches a trigger distance sampled uniformly from 18 to 26 m. Two bystanders remain beside the road. Main evaluations use ClearNoon weather at 21 validated sites, with two episodes per site and condition for each evaluation seed.

The policy has six convolutional blocks, a 256 dimensional shared feature, nine target speed bins from 0 to 8 m/s, and a steering head. A fixed proportional controller converts the selected speed into throttle and brake. Two layers of width 64 encode the preceding target speed and steering.

C.2 Intervention libraries and task audits

C.2.1 ForkRoad

The task audit checks every nonfactual feature value at 887 source histories. It admits 44 resampling or constant interventions from ten families and rejects changes to agent position, junction position, committed route, and the sign. There are 160 target failures from 324 episodes, divided equally between selection and confirmation. Each candidate receives 240 source episodes per split and must produce at least 30 failures. Stability uses 2,000 bootstrap resamples.

An independent source calibration repeats the full selection procedure with three matched mechanisms, three coherent held out task-changing mechanisms, and three split nulls. Each pseudo target split uses 80 failures sampled from 320 source episodes. All matched trials pass and all six negative trials are rejected; the smallest negative distance is 39.18. For neighboring-turn resampling, the thresholds at the size of the decision library are $d_{\max} = 0.111$ and $r_{\text{reen}} = 0.390$. Placebo replay gives $\gamma = 0$, and the affected prevalence threshold of this candidate is 0.414.

Selection chooses neighboring-turn resampling. It produces 112/240 selection failures and 127/240 confirmation failures (recurrence lower bound 0.476), with distances 0.0797 and 0.0305 and stability 1.0. No other family produces 30 failures in a split (at most 2 of 240), so stability and separation hold trivially here, in the target and in every calibration trial, and the distance check alone rejects the negatives. The causal audit affects 168/300 source trajectories, with lower bounds of 0.503 for prevalence and 23.00 for mean excess task loss from 5,000 paired bootstrap resamples. The latter equals its estimate because every affected history has the same excess loss.

C.2.2 Atari

Freeway uses 60 target failures, 30 per split, and 60 source episodes per candidate and split, against a library of seven candidates. Breakout uses twice these budgets, 120 target failures and 120 source episodes, against a library of eleven candidates. In Breakout two of the five matched trials are not usable, one on stability and separation and one because it selects another family, so those candidates receive no thresholds. Freeway selects the car color family at energy distance 0.118, against 3.779 for the next eligible candidate; Breakout selects the controlled object color family at 0.065, against 1.241. Their confirmation distances are 0.110 and 0.092, below the selected candidates' $d_{\max}$ of 0.278 and 0.249. Their recurrence lower bounds are 0.957 and 0.886, reaching r_{reen} of 0.957 and 0.875; bootstrap stability is 1.0 and 0.988, and the separation ratios are 0.017 and 0.104.

All 30 audited source histories are affected in both games, with prevalence lower bound 0.886. Branching at the first affected decision gives mean excess task loss 16.2 in Freeway (lower bound 15.9) and 155.4 in Breakout (lower bound 128.1). Each game's four negative trials, a held out camera shift, a held out action delay, and two split nulls, are rejected. Appendix D.6 applies the harm audit to every Breakout candidate.

Robotank uses 60 target failures, 30 per split, and six candidates in three families: HUD clock digits (two RAM values), brightness, and instrument-panel decoration (logo strip, panel flanks, or both). Selection chooses the decoration family at energy distance 0.169, against 6.9 for brightness and 7.1 for the HUD. The confirmation distance 0.110 is below $d_{\max} = 0.345$, the recurrence lower bound 0.957 reaches $r_{\text{reen}} = 0.957$, bootstrap stability is 1.0, and the separation ratio is 0.018. The held out camera shift is rejected at distance 9.49. All 30 audited source histories are affected, with prevalence lower bound 0.886 and mean excess task loss 48.7 (lower bound 45.8). With the clock digits as the target shift instead, the procedure selects the HUD family and passes Steps 1–3 (Appendix D.5).

Freeway uses the validated RAM closure $\{20, 77, \ldots, 86\}$ and Breakout writes RAM 62. Robotank's decoration mask sets rows 76–84 (logo strip) and rows 55–68 outside columns 22–64 (panel flanks) to gray level 64 in all four 84×84 input frames. The radar, damage indicators, reserve-tank count, and battlefield stay pixel-identical. This check passes on 4,096 cached states and 512 paired

same-action transitions, and the paired collection compares RAM outside the HUD addresses, frame number, lives, and game-over flag at every decision.

Pixel PPO collection checks RAM outside the admitted addresses, object geometry, rewards, frame number, lives, and end flags. Robotank's HUD candidates reuse their passed RAM audit.

C.2.3 CARLA

The frozen learner drives in both domains. In the source, its previous action input carries the expert's previous command, as in behavior cloning; in the target, it carries the learner's own previous command. Crossing and idle episodes each have probability 0.5 and use the 21 validated sites. An episode fails on a collision, a stall that reaches the 300 step limit, or a lane departure. The signature of Equation 23 omits the return and contains the episode length, four commanded-deceleration event times, the frequencies of the nine speed bins, and indicators for pedestrian collision, other collision, stall, and lane departure. A commanded deceleration occurs when the selected speed bin falls.

The library has fifteen candidates in eleven families. The previous action family writes the learner's input with three values: its own previous command, a constant stopped cue $(0, 0)$, and a constant cruising cue $(6, 0)$. These writes keep the current RGB image, the physical state, and the executed action history, so they change the supplied cue but not the vehicle's actual history. The ten world families vary crossing markings, a parked occluder, weather, sun altitude, ego color, camera exposure, pedestrian appearance, bystander count, pedestrian speed, and pedestrian presence.

The physical task check is computed in closed form for the scripted scenario: it tests that the set of ego positions at which a pedestrian lies in the ego's path is unchanged. It rejects slowing the pedestrian to 0.6 m/s and removing the pedestrian, and admits the remaining thirteen. World candidates receive no separate information check. For the selected input-write family, information preservation holds by construction, because the physical scene and the recorded image are unchanged.

Steps 1 and 2 use 30 target failures, 15 per split, from 160 target episodes; all are pedestrian collisions. Each admitted candidate receives 100 fresh source episodes per split and must produce at least eight failures; nine candidates are eligible. The three previous action values are the only candidates with 15 failures in both calibration splits and serve as the matched pseudo targets. Four mechanisms are held out: a three step actuator delay, a pedestrian speed of 2.5 m/s, a camera tilt from -8 to $+4$ degrees, and a pedestrian who starts crossing when the vehicle is 14 m away. The camera and late pedestrian values were fixed by a 30 episode source pilot for producing failures. The faster pedestrian produces none, which leaves three negative trials and three split nulls.

Calibration gives distance thresholds of 0.0022 for the learner's own command, 3.11 for the stopped cue, and 0.0016 for the cruising cue. The stopped cue's failures mix pedestrian collisions and stalls, which widens its null; the other two produce only pedestrian collisions whose signatures differ by about one step in episode length. All matched trials pass and all six negatives are rejected. The delay and camera trials select the stopped cue and fail on distance (846.6 and 1988.8). The late pedestrian's failures are pedestrian collisions like the target's; it selects the cruising cue at distance 0.113, far above that candidate's threshold, and also fails on stability (0.05) and separation (2.27). Each split null fails on stability or separation.

Selection picks the previous action family, represented by the cruising cue, at energy distance 6.1×10^{-5}. The learner's own command, the value used at deployment, follows at 3.8×10^{-4}, and the nearest other family, the occluder, is at 0.144. Confirmation gives distance 0.0014, below the cruising cue's threshold of 0.0016 at the size of the decision library, recurrence lower bound 0.204 against its threshold of 0.134, stability 0.911 over 2,000 resamples, and separation ratio 0.029.

Step 3 calibrates one threshold γ per intervention route on 20 source histories before target data are read. The policy acts greedily, so the response at a decision is 1 if the greedy speed bin changes and 0 otherwise. An input write reuses the recorded frame, so its placebo is exact and $\gamma = 0$. A world candidate is rendered again by replaying the recorded controls. The replay is not frame exact, and in 35 to 40% of episodes it alone changes the greedy bin at some decision, so the 95% placebo quantile gives $\gamma = 1$. Since a response cannot exceed 1, Step 3 would abstain on any world candidate here; the selected candidate is an input write and uses the exact route.

The cruising cue's minimum affected prevalence is 0.087. The causal audit applies the cruising cue to 40 fresh source histories: 20 are affected, all crossing episodes, with prevalence lower bound

0.352. Both continuations replay the recorded controls up to the first affected decision (median step 112, as the pedestrian approaches) and then drive on their own; the intervened continuation keeps the cue fixed from that decision on. The factual continuation fails in 3 of the 20 affected histories and the intervened one in 9, all pedestrian collisions, so the mean excess task loss is 0.30 with a lower bound of 0.10 from 5,000 paired bootstrap resamples, and the diagnosis passes Steps 1–3.

A constant cue does not repair the dependence. Over 42 crossing and 42 idle episodes, a forced cruising cue leaves crossing success at the frozen policy's 0.500, while a forced stopped cue lowers crossing collisions from 0.500 to 0.167 at the cost of 0.238 crossing and 0.119 idle stalls. The adaptor of Appendix B.8 therefore compares the two constant values of the diagnosed family at each decision and intervenes only when its gate fires.

C.3 Data collection and training

C.3.1 Action residual and outcome predictor

The ForkRoad return predictor is an MLP with two hidden layers of width 64 that fits standardized returns with squared error. The retained experiment uses 320 source states, 640 state-action examples, and 350 Adam updates at learning rate 2×10^{-3}.

The confirmatory adaptor run uses 320 construction states, 640 labeled views, and 895 unaffected observations from 240 source episodes. Adam performs 2,000 updates at learning rate 2×10^{-3} with batches of 256 from each set, $\lambda_{\mathrm{src}} = 1$, and $\lambda_{\mathrm{res}} = 10^{-4}$.

C.3.2 Feature adaptor and external baselines

Freeway fits the white-car value and Breakout the red paddle and ball value, the representatives the diagnosis selects. Each game supplies 8,192 training pairs from four source seeds, 4,096 validation pairs from two separate seeds, and 4,096 confirmation pairs from another two seeds, with 2,048 decisions per seed. The native frozen policy acts with probability 0.95 and otherwise takes a uniform random action. Paired environments execute identical actions. Collection costs 16,384 source decisions and the same number of intervened decisions per game.

Adam performs 2,000 updates at learning rate 10^{-3} with batches of 128 pairs and gradient norm clipped at 5. Validation feature loss selects among checkpoints every 100 updates, including the identity initialization. Collection and fitting use one CPU thread per process and no PPO updates. Five repair seeds shift the training and validation collection seed blocks and the optimization seed; the confirmation states and the twenty evaluation seeds are shared, and repair seed 0 reproduces the original single fit. RENDER selects update 2,000 in all Freeway seeds and updates 700 to 2,000 in Breakout.

Randomize-all. This arm uses the same collection budget, adaptor, loss, and checkpoint rule. Its shadow environment redraws every admitted cue at each full game reset and every 256 decisions, each cue drawn independently from its own admitted values and all of them applied to the same view. Freeway varies the car color (native or one of five palette values including white), a HUD mask, and halved brightness. Breakout varies the paddle and ball color (native or one of six palette values including red), red bricks, a HUD mask, a gray background, and halved brightness. Colors are RAM writes before each raw frame, and the other cues transform the raw screen before grayscale conversion. RAM outside the admitted addresses, object geometry, rewards, lives, and frame counters are checked against the native environment at every decision. The three pairs after a mid game redraw are dropped because their frame stacks mix two appearances. Validation uses the arm's own views, so the selected update ranges from 100 to 1,500 across Breakout seeds. For the color cues, this comparison changes both the number of randomized cues and the range of fitted values; the two Breakout controls of Appendix D.4 vary each one alone.

ILA. ILA follows the official implementation. The whole encoder is adapted with RMSprop at learning rate 10^{-4}, the paper's DrQ-v2 value. The critic is a linear, LayerNorm, and tanh trunk followed by a two layer ReLU MLP of width 1,024 with orthogonal initialization. Training uses the WGAN loss, weight clipping at 0.01, five critic updates per encoder update, batch size 256, and 2,000 encoder updates, and the final checkpoint is used, since ILA specifies no selection rule. The

source buffer holds frozen features of RENDER's native collection for the same repair seed. The target buffer holds 8,192 observations in Freeway and Breakout and 20,000 in Robotank, matching its training pairs, collected by the frozen policy in the shifted environment with the same exploration rate. Random shift augmentation is disabled because the pixel PPO policies were not trained with augmentation, and the dynamics consistency term is omitted, as the original ablation shows it inessential. The sensitivity variant uses learning rate 10^{-5}, 10,000 updates, and four times the data.

TARL. TARL follows its Atari code: the loss $H(\pi_{\mathrm{tta}}) + 1.5\, D_{\mathrm{KL}}(\pi_0 \,\|\, \pi_{\mathrm{tta}})$ on the current state, applied only when the entropy is below $0.1 \log |\mathcal{A}|$, with Adam at learning rate 3×10^{-4} and one update per decision. The action is chosen before the update with $\epsilon = 0.001$ exploration. Parameters persist across the twenty evaluation episodes of one domain and reset for each repair seed and domain. The entropy is the policy entropy of its Equation 1, and the code's unreported mixup pseudo label term is omitted. TARL adapts the LayerNorm parameters of networks it retrains with such layers; the supplied checkpoints stay frozen here, so TARL instead adapts a zero initialized feature residual with the architecture of Appendix B.7. Its response is measured with the parameters reached at the end of the shifted deployment. The low learning rate variant uses 10^{-6}.

Native PPO continuation and UCB-DrAC. Both arms continue PPO from the supplied checkpoint in the native game for two million decisions, a fifth of the original training budget, with the original training arguments: ten environments, 128 steps per rollout, four minibatches, four epochs, clipping 0.1, entropy coefficient 0.01, value coefficient 0.5, gradient norm 0.5, GAE 0.95, discount 0.99, Adam at 2.5×10^{-4} annealed linearly to zero, reward normalization, and the training wrappers of the original script. The final weights are evaluated; the shifted domain, evaluation returns, and evaluation seeds never enter fitting or selection.

UCB-DrAC uses $\alpha_r = 0.1$, exploration coefficient 0.1, and a window of ten returns. The PPO surrogate uses unaugmented observations, and the regularizer matches the policy and value on augmented observations to their detached unaugmented outputs. Because the input is four stacked grayscale frames, the grayscale augmentation is dropped and color jitter reduces to brightness and contrast in $[0.6, 1.4]$. The remaining augmentations are random crop, implemented as a four pixel random shift, cutout with zeros or with one random gray level, horizontal flip, rotation by a multiple of ninety degrees, and a random 3×3 convolution. Each draw is shared by the four frames of a stack. The Kullback–Leibler term is computed in log space, because the library divergence returns infinity when an augmented view's probability underflows. The control arm is the same continuation without augmentation.

Five fitting seeds change the environment and optimizer seeds, in blocks disjoint from every collection and evaluation seed. After the first results showed the restart perturbing the Breakout policy, a sensitivity pair repeated both arms at one tenth of the learning rate, 2.5×10^{-5}.

Robotank collection. Robotank uses the same feature adaptor, loss, optimizer, selection rule, and code path, with the Sample Factory policy and its SB3 preprocessing wrapped behind the pixel PPO interface. Four source trajectories with ϵ-greedy actions supply the training pairs and two the validation pairs, but each is collected for 5,000 decisions, the evaluation horizon. A Robotank episode drifts through a day cycle, and a 2,048 decision collection covers only its first 40%: with it, the same run recovers 29.8 on average and two seeds select the near-identity checkpoint. With the full horizon, four of five seeds select updates 600 to 2,000 and one seed the near-identity checkpoint. Randomize-all redraws the HUD clock value (native, 0, or 128), brightness halving, the logo strip mask, and the panel flank mask every 256 decisions, each from its own admitted values. The occlusion has a single value, so no palette is randomized.

C.3.3 CARLA BASE POLICY AND COLLISION PREDICTOR

Source behavior cloning uses 40,000 training frames and 8,000 validation frames, with 25 epochs, batch size 128, and AdamW at initial learning rate 3×10^{-4} with cosine decay and weight decay 10^{-4}; the checkpoint with the best validation speed-bin accuracy (epoch 7) is kept. The base training seed is 0, and the training and validation collection seeds are 0 and 777. The crossing is painted in every crossing episode and in half of the idle episodes, and each episode inserts a scripted stop of 25 to 80 steps, mostly 3 to 9 m before the crossing, so the data contain expert labels for resuming

from rest. Expert actions are available in this original dataset; the repair labels come from source outcomes and do not query the expert.

The collision predictor h_ψ is a $256 \to 64 \to 1$ MLP with 16,513 trainable parameters on frozen policy features. Every retained frame is labeled by whether its whole episode ends in a pedestrian collision. Class weighted binary cross entropy is fitted with AdamW for twelve epochs, batch size 256, learning rate 10^{-3}, and weight decay 10^{-4}, with no source policy KL penalty. The labels are per episode rather than per action, so the predictor scores outcomes; it does not estimate action values.

Construction uses 60 source rollouts with crossing probability 0.6 and alternating cue values $(0, 0)$ and $(6, 0)$, the stopped and cruising values of the diagnosed family. Retaining every second frame gives 4,923 labels with collection seed 5. One fitted predictor serves all three evaluation seeds.

C.4 Evaluation protocols and statistical reporting

Statistical decisions use 95% confidence. Joint decisions require simultaneous coverage across their claims and any candidates considered. Source rollouts calibrate the thresholds for matching failures and detecting policy response; failure rules, task tolerances, and deployment margins follow the application. Split sizes, score scales, loss weights, optimizers, and checkpoint rules follow the reported protocol, with source validation for model selection. Apart from the Robotank collection length (Appendix C.3.2), no threshold or training parameter is selected using target evaluation performance.

Repair-seed variation and episode sampling are reported separately. Main figures show means and standard errors across the five repair seeds with the evaluation episodes held fixed and shared across arms, so episode sampling does not enter that error. Per-fit paired bootstrap intervals resample evaluation episodes and describe episode uncertainty for that fitted controller, and Table 3 reports their envelope over the five fits. An episode-level standard error for a single controller treats its evaluation episodes as independent draws.

C.4.1 Joint confirmatory evaluation

In ForkRoad, target, source, and dependence evaluations each use 1,200 fresh paired episodes, and dependence uses Equation 25. The three one sided lower bounds for Equations 19 to 21 use Bonferroni allocation and 5,000 paired bootstrap resamples, with margins $\epsilon_{\mathrm{tgt}} = 0$, $\epsilon_{\mathrm{src}} = 0.02$, and $\epsilon_{\mathrm{dep}} = 0$. The diagnosis, base policy, and evaluation episodes are fixed. Five repair seeds change only the construction seed blocks and the scorer and adaptor initializations; seed 0 is the original confirmatory run.

Randomize-all uses the same 320 construction state rule, applied to histories affected by any admitted cue, and builds four views per state, each redrawing all ten admitted cues from their own values, so it receives twice as many labeled views as RENDER. Its scorer zeroes all ten cues instead of the diagnosed one, and its preservation set keeps only observations that respond to no admitted cue (about 410 per seed, against 895 for RENDER); the vote, gate, adaptor, and update budget are unchanged. Figure 4 reports the mean over seeds and its standard error, and Table 2 reports every seed. Executed actions define the response; temperature scaled Q values only break ties between equally responsive decisions.

C.4.2 Return evaluation

Freeway uses twenty matched seeds starting at $70{,}640{,}000$ and Breakout twenty starting at $10{,}000$, all disjoint from fitting, validation, and confirmation. Pixel preprocessing uses OpenCV grayscale conversion and area resizing to 84×84, four frame stacks, frameskip 4, sticky action probability 0.25, and no max pooling. Actions are greedy, apart from the TARL exploration stated above. The environment limit is 108,000 raw frames.

Freeway episodes end naturally. Breakout uses FireResetEnv only at full game reset; after its two reset actions the limit is 26,998 decisions. The frozen policy reaches the limit in eight of twenty episodes in each condition, and across the five repair seeds RENDER reaches it in two to ten native and three to six red episodes. Every episode enters the reported returns. Forty cached Breakout base-

line episodes are reused after matching the checkpoint, interface, environment, software versions, action selection, and seeds.

Robotank uses twenty matched seeds starting at $91{,}740{,}000$, disjoint from training and validation. Its 5,000 decision horizon means the returns are not full-game scores. Table 3 uses 10,000 paired bootstrap resamples over evaluation episodes for each repair seed in all three games. Final evaluation returns never select a checkpoint.

C.4.3 Success rates and uncertainty

CARLA uses evaluation seeds 0, 1, and 2, with 42 matched episodes per condition and seed, so 126 per condition. Four controllers share these episodes and one base checkpoint. Frozen uses the factual query. RENDER is the adaptor of Appendix B.8. Equal weights keeps that adaptor and its gate but sets $\beta = 0$, so the vote in Equation 12 averages the two speed distributions without the collision scores. The oracle gate keeps $\beta = 1$ and replaces only the runtime gate with the simulator's pedestrian state: it fires from the moment the pedestrian starts crossing until the vehicle is two meters past the crossing. It reads privileged state and is a reference for the gate, not a deployable controller.

RENDER is fitted with five repair seeds. Seed 0 is the retained fit; seeds 1 to 4 each collect their own 60 construction rollouts and refit the predictor. All are evaluated on the same 126 episodes per condition, with the same gate, β, and logit score scale. Figure 6 reports the mean over the five fits and its standard error. Frozen, equal weights, and the oracle gate are each evaluated once.

Table 4 reports pooled Wilson intervals, and Table 5 reports every repair fit. Both crossing and idle evaluations use unmarked scenes and the learner's own action history.

D Main experiment results and analysis

D.1 Outcome label audit

The diagnosis results of every benchmark appear with their protocols in Appendix C.2. Before ForkRoad repair construction, an independent audit compares the outcome vote of Appendix B.5.1 with the best candidate's actual source continuation return at 240 states. All 240 labels are retained and have zero regret within the candidate set. The one sided 95% upper bound on the error rate is 0.01115, below the declared tolerance of 0.05.

D.2 Return differences and success intervals

Tables 2 to 5 report per-fit results and uncertainty under the protocols of Appendix C.4: every ForkRoad repair seed with its deployment-test bounds, paired return differences for Atari, pooled CARLA success intervals, and the CARLA runtime adaptor for each repair seed.

Table 2: ForkRoad per-seed results and simultaneous one sided 95% lower bounds. Target and source columns are success rates; response is the population weighted response at the frozen policy's affected decisions. Bounds are for the paired gain over the frozen policy; the deployment test requires target ≥ 0, source ≥ -0.02, and dependence reduction ≥ 0.

Repair	Seed	Target	Source	Response	Target LB	Source LB	Dep. LB
Randomize-all	0	0.921	1.000	0.030	+0.404	+0.000	+0.427
	1	0.843	1.000	0.090	+0.326	+0.000	+0.368
	2	0.817	1.000	0.080	+0.302	+0.000	+0.378
	3	0.813	1.000	0.057	+0.299	+0.000	+0.401
	4	0.849	1.000	0.041	+0.333	+0.000	+0.417
RENDER	0	0.998	0.995	0.003	+0.480	−0.010	+0.456
	1	1.000	1.000	0.000	+0.482	+0.000	+0.457
	2	0.971	1.000	0.000	+0.455	+0.000	+0.458
	3	0.953	0.991	0.001	+0.437	−0.016	+0.456
	4	0.965	1.000	0.000	+0.448	+0.000	+0.458

Table 3: Paired return differences over the frozen policy for the pixel policies in Figure 5. Difference is the mean over five repair seeds of the paired mean gain on 20 matched episodes; seed range is the span of the five per-seed means; the interval is the envelope of the five per-seed 95% paired bootstrap intervals.

Task	Condition	Repair	Difference	Seed range	95% envelope
Freeway	Source	Randomize-all	0.04	[0.00, 0.10]	[−0.15, 0.25]
Freeway	Source	RENDER	−0.63	[−1.35, −0.15]	[−1.60, 0.05]
Freeway	Shift	Randomize-all	3.53	[1.85, 6.35]	[1.10, 6.80]
Freeway	Shift	RENDER	9.63	[7.45, 11.55]	[7.05, 11.90]
Breakout	Source	Randomize-all	5.70	[−15.35, 35.85]	[−67.00, 77.90]
Breakout	Source	RENDER	14.68	[6.60, 29.55]	[−50.40, 80.10]
Breakout	Shift	Randomize-all	−13.39	[−46.65, 43.35]	[−80.95, 85.80]
Breakout	Shift	RENDER	70.18	[50.55, 91.00]	[7.15, 134.40]
Robotank	Source	Randomize-all	1.05	[0.05, 1.85]	[−2.65, 5.70]
Robotank	Source	RENDER	0.83	[−1.60, 2.25]	[−4.65, 5.55]
Robotank	Shift	Randomize-all	5.35	[−3.60, 11.40]	[−5.75, 15.20]
Robotank	Shift	RENDER	18.42	[2.70, 23.50]	[−0.50, 28.75]

Table 4: CARLA deployment success and Wilson 95% intervals ($n = 126$ per condition) for the four arms of Figure 6. The RENDER row is repair seed 0, the retained fit, whose predictor the oracle gate also uses; all five repair seeds are in Table 5.

Controller	Crossing ↑	Idle ↑
Frozen	0.500 [0.414, 0.586]	1.000 [0.970, 1.000]
Equal weights	0.508 [0.422, 0.594]	1.000 [0.970, 1.000]
RENDER	0.881 [0.813, 0.927]	0.984 [0.944, 0.996]
Oracle gate	0.897 [0.831, 0.939]	1.000 [0.970, 1.000]

Table 5: RENDER's CARLA runtime adaptor per repair seed, on the same 21 sites, three evaluation seeds and 126 episodes per condition as Figure 6. Seed 0 is the retained fit. AUC is the collision predictor's ranking accuracy on its own fitting data, a fitting statistic rather than a held-out estimate, and the firing columns are the fraction of decisions at which the gate intervenes. Every failure in idle episodes is a stall; none is a collision.

Repair seed	$\hat{V}$ AUC	Crossing	Idle	Fires, crossing	Fires, idle
0	0.719	0.881	0.984	0.125	0.047
1	0.825	0.889	0.937	0.174	0.101
2	0.722	0.889	0.992	0.131	0.036
3	0.754	0.905	0.984	0.121	0.044
4	0.752	0.921	0.984	0.141	0.046
Mean ± s.d.	0.754 ± 0.043	0.897 ± 0.016	0.976 ± 0.022	0.138 ± 0.021	0.055 ± 0.026

Crossing success is stable across the five repair fits; idle completion varies more. The fit whose collision predictor has the highest fitting AUC also fires its gate most often, in 17.4% of crossing and 10.1% of idle decisions against 12.1 to 14.1% and 3.6 to 4.7% for the others, and it stalls in eight of 126 idle episodes. More frequent intervention thus goes with lower idle completion in these runs; a higher fitting AUC alone does not mean better counterfactual scoring. The retained fit, repair seed 0, is the lowest of the five on crossing episodes.

D.3 Published adaptation baselines on Atari

Table 6 repeats the arms of Figure 5 with their learning-rate variants and adds a native-agreement column. Every arm starts from the same base checkpoint for its game and uses the same repair seeds and matched evaluation episodes. ILA (Yoneda et al., 2022) aligns source and target features, and TARL (Xu et al., 2025) adapts online with an entropy objective and a KL penalty. Both see target observations, which RENDER never uses for fitting. Appendix C.3.2 lists every setting and adaptation to the present checkpoints.

Response is the fraction of shared confirmation states at which the native and shifted views give different greedy actions. Native agreement is the fraction of those states at which the method's greedy action on the native view is the frozen policy's action. The two are read together, because a method can lower the response by changing the policy everywhere. ILA in Breakout is the clearest case: it lowers the response from 27.9% to 21.4%, but keeps only 59.4% of the native actions and drops the source return from 185.40 to 54.24. RENDER keeps 94.4% of the native actions, lowers the response to 18.4%, and reaches source return 200.08. In Robotank, ILA's response (27.5%) is below RENDER's (31.6%), but ILA loses 16 points of source return and does not raise the shifted return (17.78 against 19.70).

TARL moves the target means little at the main learning rate: Freeway rises from 16.85 to 17.83, Breakout from 55.00 to 65.55, and Robotank falls from 19.70 to 17.11. Its entropy filter admits about 3% of the shifted Breakout decisions, which limits updates on that deployment, although the shared parameters let each update change greedy actions elsewhere; its Breakout native agreement is 84.4%. In Robotank it updates on about two fifths of native and a fifth of shifted decisions without a target gain.

Native PPO continuation separates further training from augmentation. At the original learning rate, UCB-DrAC (Raileanu et al., 2021) chooses random crops in every Freeway seed, and source and target stay within a point of the frozen policy. In Breakout it selects brightness and contrast jitter in three seeds and random convolution in two; the red shift improves by 28 points on average, with per-seed gains from -16 to $+57$, while source return drops by 54 points on average and one seed falls to 38. Restarting the optimizer first drops the training return to between 20 and 80 before it recovers to between 140 and 180 by the end of the budget, and the unaugmented control shares this effect (source 171). At one tenth of the learning rate, Freeway again stays within a point of the frozen policy. In Breakout the bandit returns to random crops in four of five seeds, the red return is 38.97 for UCB-DrAC and 14.64 for the control, and source return stays below the frozen policy (138.20 and 152.35). In Robotank, continued training at the main rate deepens the dependence on the panel decoration: the shifted return falls to 14.06 with augmentation and 9.26 without, the response rises above 80%, and the bandit chooses random crops in 1,556 of 1,562 iterations. At the low rate both arms stay within two points of the frozen policy.

Where generic adaptation or continued training keeps source performance, it recovers at most a small part of RENDER's target gain. RENDER has the highest target mean in all three games and keeps at least 94% of the native actions.

D.4 Behavioral confirmation

Confirmation uses 4,096 matched histories in Freeway and Breakout and 10,000 in Robotank. The source histories are shared across arms and fixed across repair fits, and these temporally correlated states from two seeds are not treated as independent episodes. Figure 5 reports the response means over the five repair fits. RENDER keeps 98.8%, 94.4%, and 96.2% of the frozen policy's native actions in Freeway, Breakout, and Robotank. Randomize-all keeps 98.6%, 94.3%, and 96.1% but

Table 6: Atari adaptation baselines on the same frozen pixel PPO policies, 20 matched episodes per condition. Each $\pm$ is the standard error of the tabulated mean under the randomness that row has: the five repair seeds for the fitted rows, and the 20 episodes for the frozen policy, which has no repair seeds. The episodes are fixed and shared, so they do not enter the fitted rows' errors. The frozen response and its native agreement carry none, the first because the confirmation states are fixed and temporally correlated and the second because the frozen policy is its own reference. ILA (Yoneda et al., 2022) aligns the whole encoder to unlabeled target observations (8,192 in Freeway and Breakout, 20,000 in Robotank); its variant uses learning rate 10^{-5}, 10k updates and four times the data. TARL (Xu et al., 2025) adapts online on the evaluated episodes; its low-learning-rate variant uses 10^{-6}. The fine-tuning rows unfreeze the whole network and continue PPO in the native game for two million decisions, with UCB-DrAC's generic augmentations (Raileanu et al., 2021) or none; their low-learning-rate variants use 2.5×10^{-5}. Response is the fraction of shared confirmation states (4,096 in Freeway and Breakout, 10,000 in Robotank) whose greedy action differs between the native and shifted views, so it measures how much the shift still moves the deployed decision. Native agreement is the fraction of those states whose greedy action on the *native* view is unchanged from the frozen policy, so it measures how much of the given policy each method overwrote to get there. Reading the two together separates a targeted correction (low response, high native agreement) from a rewritten policy (low response bought by disagreeing with the base policy everywhere).

Method	Source ↑	Target ↑	Response (%) ↓	Native agr. (%)
Freeway, white cars				
Frozen	32.85 ± 0.08	16.85 ± 0.13	34.4	–
ILA	27.48 ± 0.72	17.77 ± 0.38	23.9 ± 1.1	88.1 ± 0.5
ILA (small steps, more data)	29.32 ± 0.67	12.59 ± 2.06	41.3 ± 6.0	87.3 ± 0.6
TARL	32.61 ± 0.08	17.83 ± 0.08	31.4 ± 0.7	96.2 ± 0.4
TARL (low lr)	32.60 ± 0.04	18.07 ± 0.11	29.3 ± 0.1	96.4 ± 0.0
Native fine-tune (control)	32.57 ± 0.11	16.46 ± 0.83	32.2 ± 1.1	87.7 ± 0.6
UCB-DrAC fine-tune	32.42 ± 0.16	16.14 ± 0.44	31.0 ± 0.9	84.7 ± 0.4
Native fine-tune (low lr)	32.77 ± 0.18	17.54 ± 0.34	31.0 ± 0.9	92.3 ± 0.5
UCB-DrAC fine-tune (low lr)	32.90 ± 0.15	17.74 ± 0.35	29.7 ± 1.0	90.7 ± 0.3
Randomize-all	32.89 ± 0.02	20.38 ± 0.88	21.1 ± 0.9	98.6 ± 0.1
RENDER	32.22 ± 0.22	26.48 ± 0.66	8.3 ± 0.1	98.8 ± 0.0
Breakout, red ball and paddle				
Frozen	185.40 ± 19.10	55.00 ± 16.43	27.9	–
ILA	54.24 ± 6.05	37.21 ± 3.60	21.4 ± 0.3	59.4 ± 0.9
ILA (small steps, more data)	49.46 ± 3.64	29.73 ± 1.89	23.2 ± 0.5	60.6 ± 1.0
TARL	209.29 ± 9.95	65.55 ± 6.61	31.2 ± 0.8	84.4 ± 1.8
TARL (low lr)	217.58 ± 6.59	50.14 ± 1.39	27.9 ± 0.1	99.9 ± 0.0
Native fine-tune (control)	171.30 ± 11.50	42.99 ± 12.15	27.5 ± 1.7	52.0 ± 1.1
UCB-DrAC fine-tune	131.13 ± 24.42	83.44 ± 12.87	19.7 ± 1.5	47.6 ± 1.1
Native fine-tune (low lr)	152.35 ± 12.92	14.64 ± 3.59	28.8 ± 0.6	70.9 ± 1.7
UCB-DrAC fine-tune (low lr)	138.20 ± 19.78	38.97 ± 13.01	25.6 ± 1.4	65.0 ± 2.2
Randomize-all	191.10 ± 9.42	41.61 ± 16.91	26.4 ± 0.9	94.3 ± 1.1
RENDER	200.08 ± 4.02	125.18 ± 6.67	18.4 ± 0.4	94.4 ± 0.1
Robotank, occluded instrument panel				
Frozen	66.80 ± 1.11	19.70 ± 1.19	59.9	–
ILA	50.32 ± 2.94	17.78 ± 0.64	27.5 ± 1.0	75.2 ± 0.7
ILA (small steps, more data)	53.35 ± 0.96	19.37 ± 0.68	28.2 ± 0.7	76.8 ± 1.2
TARL	67.35 ± 0.58	17.11 ± 0.68	50.7 ± 1.1	78.3 ± 1.1
TARL (low lr)	67.34 ± 0.50	18.14 ± 0.61	53.7 ± 0.3	92.7 ± 0.2
Native fine-tune (control)	63.78 ± 0.97	9.26 ± 1.05	81.7 ± 4.1	44.2 ± 2.3
UCB-DrAC fine-tune	65.72 ± 0.42	14.06 ± 3.10	80.1 ± 2.6	38.5 ± 2.7
Native fine-tune (low lr)	66.29 ± 0.71	18.45 ± 1.45	60.1 ± 0.5	86.3 ± 0.2
UCB-DrAC fine-tune (low lr)	68.45 ± 0.37	20.87 ± 0.77	61.0 ± 0.6	84.6 ± 0.6
Randomize-all	67.85 ± 0.30	25.05 ± 2.48	42.8 ± 1.3	96.1 ± 0.6
RENDER	67.63 ± 0.70	38.12 ± 3.96	31.6 ± 3.9	96.2 ± 0.3

leaves responses of 21.1%, 26.4%, and 42.8%, against 34.4%, 27.9%, and 59.9% for the frozen policy.

Palette and added-cue controls. The reported Atari repair fits the cue value observed at deployment, the value the diagnosis identifies. As a control, the same adaptor redraws only the diagnosed cue over all of its admitted values, the palette that Randomize-all draws from, with no other cue changed. Over five repair seeds it gives Freeway source 32.73 and white 23.29 with response about 16%, between the single-value repair's 26.48 and Randomize-all's 20.38. In Breakout the six-color paddle and ball palette fails: red return 44.18 against the frozen policy's 55.00, with per-seed means from 26 to 71 and a validation response that falls by about one point, against about ten for the single-value repair. A second control keeps the paddle and ball at the single red value and redraws the other four cues of Randomize-all under its rule. It also fails: red return 50.77, with per-seed means from 32 to 75, response 25.1%, and source return 184.72. Either difference from Randomize-all alone therefore reproduces its Breakout failure: the adaptor fits one value of the diagnosed cue, but neither a palette of that cue nor the same value mixed with other cues. The single-value repair is specific to the fitted value: on unseen Freeway car colors it gains at most two points, while the palette variant and Randomize-all gain 5.6 on the unseen color 130 and 7.6 and 6.8 on color 0, which their palettes contain.

Additional experiment: blanked HUD clock. Robotank's library contains a second harmful mechanism, the HUD clock digits. Blanking them lowers the frozen return from 66.80 to 49.95 on the twenty evaluation seeds, and the diagnosis selects the HUD family and passes Steps 1 to 3 (Appendix D.5). The same single-value adaptor, fitted over five repair seeds on 3,072 states subsampled from twelve full-horizon trajectories, raises the target return by 5.55 (per-seed gains from 4.80 to 6.55) with a source change of +1.32. The clock costs the frozen policy 16.85, about a third of the panel occlusion's 47.10, and changes fewer decisions (35% against 64% of validation states). The repair recovers about a third of this loss. Under the same fitting protocol the panel occlusion repair gains 19.37, about two fifths of its loss, and reaches 39.07, consistent with the 38.12 of the common code path in Figure 5.

D.5 Diagnosis over five seed blocks

Each benchmark repeats Steps 1 to 3 on five independent seed blocks. Block 0 is the one reported in Appendix C.2. A block redraws every source and target episode (calibration, target failures, decision library, and causal audit) from its own seed range and keeps the budgets of Appendix C.2. Table 7 counts the outcomes.

Table 7: Steps 1 to 3 over five independent seed blocks. Accepted counts the blocks in which Steps 1 and 2 accept the selected mechanism, and Passed the blocks in which Step 3 then confirms real-world bias. Every accepted block selects the true family. Negatives counts the held out and split null trials that the complete rule rejects, over all blocks. The last column gives the range of the lower bound on mean excess task loss over the passing blocks. The Breakout action delay target has no mechanism in the library, so abstention is its correct output.

Benchmark	Target	Accepted	Passed	Negatives	Harm lower bound
ForkRoad	neighboring turn	4/5	4/5	30/30	23.00
Freeway	car color	5/5	5/5	20/20	15.8 to 16.0
Breakout	red paddle	5/5	5/5	20/20	106.6 to 137.5
Breakout	white bricks	5/5	5/5	20/20	40.5 to 108.4
Breakout	action delay	0/5	–	20/20	–
Robotank	decoration	5/5	5/5	5/5	44.8 to 47.3
CARLA	previous action	4/5	3/5	30/30	0.05 to 0.10

No block accepts a wrong mechanism and no negative trial is accepted. The three blocks with a library mechanism that do not pass all fail on the side of abstention. ForkRoad block 3 misses the distance check at 0.105 against 0.101, a check that a correct match fails with probability 5% by construction. CARLA block 1 misses both the distance check (0.0035 against 0.0018) and the recurrence check (0.143 against 0.168); its decision library produced 20 failures in 100 episodes of

the learner's own command, against 28 to 38 in the other batches of that block. CARLA block 4 accepts the previous action family but its harm lower bound is exactly zero (mean 0.11), so Step 3 abstains. All three abstentions come from sampling noise in the finite target and source samples. The distance, recurrence, and prevalence thresholds each reject a correct match with probability 5%, so together they allow up to about 15% abstentions per diagnosis, or about 4.5 of the 30 in-library runs, before counting the limited power of the harm test in small audits such as CARLA's. Three occur.

Clock digits as the target. Robotank has a second harmful mechanism in its library. With the HUD clock digits zeroed as the target shift and the seed block of Appendix C.2.2, selection chooses the HUD family at energy distance 0.132, against 0.175 for the other clock value and 3.33 for the panel flanks. Confirmation gives distance 0.143 against the clock candidate's threshold of 0.301, recurrence lower bound 0.649 against 0.544, stability 1.0, and separation ratio 0.038. All 30 audited source histories are affected (prevalence lower bound 0.886), and the mean excess task loss is 24.4 with lower bound 20.7, about half that of the decoration mechanism. Each Robotank target thus selects its own family, and both dependencies carry real-world bias.

D.6 Target failures select the deployed mechanism

This subsection gives the protocol and the checks behind Section 4.4. The red paddle and white brick repairs do not transfer (Table 8). The red paddle adaptor leaves white bricks at the frozen level, and a white brick adaptor fitted with the same recipe at its own single value lowers return under the red paddle. T3 lowers return to 12.9. Harm-only applies the Step 3 audit to all eleven task-preserving candidates on the same source histories as RENDER and abstains if none passes. A block uses 120 source episodes per candidate and split, 60 target failures per split, and 30 audited source histories.

On T1 and T2, RENDER confirms the true mechanism against that candidate's own threshold: the red paddle's confirmation distances run from 0.019 to 0.097 against $d_{\max}$ from 0.223 to 0.272, and white bricks' from 0.047 to 0.068 against $d_{\max}$ from 0.124 to 0.150. On T3, the closest candidate in four blocks is white bricks, which fails only the distance check, at distances from 0.80 to 1.13 against its $d_{\max}$ of at most 0.150; in the fifth block the closest candidate, green bricks, fails all four checks. For Harm-only, five or six candidates pass Step 3 in every block, and the source audit always ranks the gray background and the blue paddle first (lower bounds from 151.8 to 183.8), dimming third, the red paddle fourth, and white bricks fifth. Harm-only therefore deploys the gray background repair in four blocks and the blue paddle repair in the fifth, where the two tie at 177.4. The gray background repair lowers return on both harmful targets.

Return-only is computed from the same cached diagnosis, as defined in Appendix D.7; rebuilding RENDER's thresholds and decisions from these caches reproduces all fifteen cases. Halved brightness, a library candidate, makes the policy fail with nearly the same returns as the red paddle, so on T1 the return alone cannot separate the two families: in four blocks the nearest other family is about as close as the selected one (separation ratios from 1.21 to 1.61), against at most 0.14 for RENDER, and Return-only abstains. On T2 it abstains in one block, where halved brightness is nearest and fails three of the four checks, and accepts white bricks in the other four. It rejects T3 in every block and never accepts a wrong mechanism.

Table 8: Every repair that a selection rule deploys in Table 1, evaluated on every target. Each fitted row uses five single-value adaptors fitted with the Atari recipe of Appendix B.7 at the named cue value. The frozen row gives mean returns; the fitted rows give the paired gain over the frozen policy as mean ± s.d. over the five repair seeds, on the 20 matched evaluation episodes of the main experiment.

Repair	T1 red paddle	T2 white bricks	T3 action delay	Source
Frozen	55.0	70.8	12.9	185.4
Red paddle and ball	$+70.2 \pm 14.9$	-2.3 ± 21.9	$+2.8 \pm 1.8$	$+14.7 \pm 9.0$
White bricks	-20.6 ± 9.1	$+61.0 \pm 16.7$	$+1.1 \pm 1.2$	$+3.9 \pm 21.9$
Gray background	-30.1 ± 5.4	-10.6 ± 26.4	-0.2 ± 1.0	$+2.5 \pm 13.0$
Blue paddle and ball	$+20.0 \pm 32.2$	$+2.7 \pm 16.4$	$+0.4 \pm 1.6$	-19.6 ± 18.6

D.7 EQUALLY HARMFUL CANDIDATES IN ROBOTANK

This subsection repeats the ablation of Section 4.4 on the Robotank policy, with candidates made equally harmful. The library keeps the six Robotank candidates of Appendix C except that dimming scales every pixel by 0.75 instead of 0.5. The factor was set before any diagnosis, on source seeds disjoint from all diagnosis blocks: from 0.55 to 0.8 every factor fails all source episodes with a mean failure return between 19.6 and 22.1. We took 0.75, whose failure return matched the occluded panel's on a second disjoint batch (21.0 against 21.2; Table 9). The logo strip and the clock digits needed no change. A block uses 60 source episodes per candidate and split, 30 target failures per split, and 30 audited source histories, as in the Robotank main experiment. The two clock values give identical policy inputs, so selecting either counts as the true mechanism; RENDER selects the other value on the clock target in two blocks, and both use the same repair.

Table 9: The two equally harmful pairs. Source failure rate and failure return (mean $\pm$ s.d.) come from the 60 episode screen that set the dimming factor; Step 3 harm is the range over the five blocks of the lower bound on first-affected excess task loss. Frozen and target returns are on the 20 evaluation episodes of the main experiment, where the source return is 66.8.

Pair	Candidate (family)	Failure rate	Failure return	Step 3 harm	Frozen return
1	Occluded panel (decoration)	1.00	21.7 ± 6.7	44.4 to 49.5	19.7
	Dimming to 75% (brightness)	1.00	21.2 ± 3.4	45.3 to 47.2	21.2
2	Occluded logo strip (decoration)	0.72	41.5 ± 4.4	25.1 to 28.7	42.9
	Blanked clock digits (HUD)	0.78	42.3 ± 8.0	18.9 to 21.5	50.0

Return-only is RENDER's Steps 1 and 2 with the signature cut to the episode return: the same splits, library batches, recurrence thresholds and four checks, with distance thresholds recomputed from the same null on the return alone. In Breakout this gives a distance threshold to every candidate, including those whose matched calibration trial RENDER could not use; in Robotank RENDER already has one for all six. Harm-only applies the Step 3 audit to all six candidates on the same source histories as RENDER. All six pass in every block. The occluded panel and dimming rank first and second in every block, 0.5 to 1.0 apart in four blocks and 3.7 apart in the fifth; Harm-only selects the panel in three blocks and dimming in two.

Table 10 gives the outcomes. RENDER accepts every in-library target with a confirmation stability of 1.00 and a separation ratio (selected distance over the nearest other family) of at most 0.12, and rejects the camera shift at distances from 8.1 to 9.5. Return-only abstains on the occluded panel in one block, where the dimming batch is nearest, and on the logo strip in two, each time on the distance check; the matches it accepts reach separation ratios of 0.48 and stabilities as low as 0.67. Repairs are specific to their mechanism (Table 11). The clock repair lowers return under the occluded logo strip and the panel repair lowers it under blanked clock digits. The logo strip repair also helps the occluded panel, whose mask contains the logo strip. No adaptor restores return under dimming, including its own, so a correct selection there gains nothing.

Table 10: Robotank outcomes by target, as true mechanism / other mechanism / abstention counts over five diagnosis seed blocks, with the target gain over the frozen policy averaged over blocks and repair seeds. Oracle equals RENDER on every target.

	Harm-only		Return-only		RENDER	
Target	Counts	Gain	Counts	Gain	Counts	Gain
Occluded panel	3/2/0	$+10.2$	4/0/1	$+15.5$	5/0/0	$+19.4$
Dimming to 75%	2/3/0	-1.6	5/0/0	-1.9	5/0/0	-1.9
Occluded logo strip	0/5/0	-3.1	3/0/2	$+11.1$	5/0/0	$+18.5$
Blanked clock digits	0/5/0	-4.1	5/0/0	$+5.6$	5/0/0	$+5.6$
Camera shift	–/5/0	-0.6	–/0/5	0.0	–/0/5	0.0
All 25 cases	5/20/0	$+0.2$	17/0/8	$+6.1$	20/0/5	$+8.3$

Table 11: Every Robotank repair on every target. Each row uses five single-value adaptors fitted with the Atari recipe of Appendix B.7 under the Robotank fitting protocol of Appendix D.4; cells give the paired gain over the frozen policy as mean ± s.d. over the five repair seeds on the 20 evaluation episodes.

Repair	Panel	Dimming	Logo strip	Clock	Camera	Source
Frozen return	19.7	21.2	42.9	50.0	9.3	66.8
Occluded panel	$+19.4 \pm 1.7$	-1.3 ± 1.0	$+1.6 \pm 1.5$	-7.9 ± 1.6	-0.3 ± 1.2	$+2.2 \pm 1.6$
Dimming to 75%	-3.5 ± 1.7	-1.9 ± 1.2	-10.1 ± 2.3	$+1.7 \pm 2.1$	-1.0 ± 0.9	$+0.8 \pm 0.6$
Occluded logo strip	$+13.0 \pm 0.9$	-1.1 ± 0.9	$+18.5 \pm 2.4$	-2.6 ± 2.7	-1.7 ± 1.1	$+0.2 \pm 0.9$
Blanked clock digits	-3.9 ± 1.0	-0.8 ± 0.5	-7.9 ± 0.8	$+5.6 \pm 0.7$	-1.3 ± 1.4	$+1.3 \pm 2.0$

E Related work

RENDER addresses a problem that arises after deployment: given a trained policy, a source simulator, and a few unpaired target failures, identify a supported harmful dependence and repair it. Shortcut behavior, causal confusion, intervention-based auditing, and policy adaptation all have substantial precedents, and we do not claim the phenomenon. Our contribution is a task-loss definition that makes these instances comparable, an identifiability result about what source records can decide, and a procedure that connects failure reenactment to a verified dependence and its repair without presupposing which cue is responsible. Its scope is the admitted intervention library and the audited history distributions.

Behavioral cloning can confuse correlates of expert actions with the variables that determine appropriate behavior (de Haan et al., 2019). The copycat problem is a temporal instance: preceding expert commands predict the next command well enough that a policy neglects the current scene (Wen et al., 2020), which is the dependence our CARLA study examines (Section 4.3). Reinforcement learning agents can converge onto narrow trajectory distributions and pick up correlations that fail once they leave them (Suau et al., 2024). Goal misgeneralization separates retained capability from pursuit of a proxy correlated with reward during training (di Langosco et al., 2022; Shah et al., 2022), and shortcut learning gives a broader account across supervised settings (Geirhos et al., 2020). Sequential instances include an Atari agent that reads the opponent's paddle as a proxy for the ball (Delfosse et al., 2024) and robot policies that bind a task to a viewpoint or background (Xing et al., 2025). In each of these works the cue is given by the study design. Our setting starts one step earlier, when the responsible cue is unknown and the only target evidence is a few failure trajectories. Definition 2 states the effect for a fixed policy, task-preserving intervention, and history distribution, so that an instance can be confirmed or rejected, and Section 3 uses the failures to guide the search for the cue.

Several lines of work define independence from a nuisance by intervening on it. Counterfactual fairness constrains predictions under interventions on protected attributes (Kusner et al., 2017). Counterfactual invariance studies changes to spurious input components and builds stress tests on them (Veitch et al., 2021). The interventional robustness score measures how much a representation moves when other generative factors change (Suter et al., 2019). Other approaches use several environments or group labels to seek invariant prediction or lower worst-group risk (Peters et al., 2016; Arjovsky et al., 2019; Sagawa et al., 2019). Our definition is stated on future task loss rather than on response. Action response is evidence of dependence, but a response that costs no task loss is not a harmful dependence in our sense. The sequential continuation captures delayed consequences, such as a cruising decision followed several steps later by a collision, and so separates whether a cue changes a decision from whether that change harms the task. Reenactment and repair use one source environment and need neither several training environments nor group labels.

Meaning-preserving perturbations are a standard robustness test. Corruption benchmarks measure errors under prescribed input changes (Hendrycks & Dietterich, 2019), behavioral testing applies label-preserving transformations (Ribeiro et al., 2020), and domain randomization varies environmental factors during training (Tobin et al., 2017). Concurrent work studies semantics-preserving interventions and the success-rate drops they induce in vision-language-action models (Liu et al., 2026). In a sequential setting, an intervention on the observation can remove information the policy needs, and the resulting drop then reflects a harder task rather than a nuisance dependence. RENDER

therefore makes preservation an explicit admission requirement: the task state, reference decision, reward, and transition must be preserved, and the information needed for the task must remain available. The last condition is checked separately because hiding a needed object can leave dynamics and rewards unchanged. Appendix B.2 describes the audit and its scope. Robotank's selected mask, for example, changes decorative panel regions while leaving the battlefield, radar, and task indicators pixel-identical. Our contribution here is the use of these checks inside a failure-guided diagnosis.

Another line models nuisance as structure of the environment. Exogenous state variables evolve independently of the agent's actions and do not enter the reward (Dietterich et al., 2018), and exogenous block MDPs separate an endogenous task state from exogenous variation, with rewards and endogenous transitions depending only on the former (Efroni et al., 2021). Task preservation is related to this separation but is stated locally, for one candidate intervention on the states where it is applied, and it is checked at diagnosis time rather than assumed as a global factorization before learning. This is what a procedure that must find the responsible cue needs: candidates are proposed, audited, and rejected. In imitation, the reference-decision equality also makes explicit which expert behavior is preserved. Under the model of Section 2, the reference depends on the task state, so changing an auxiliary input to the learner does not redefine it.

Causal identification asks whether the available distribution determines a causal query uniquely (Pearl, 2009). In imitation learning, graphical criteria decide when expert behavior can be imitated from demonstrations, including when the expert and learner observe different variables (Zhang et al., 2020; Kumor et al., 2021). Our query concerns a policy that is already trained: $\mathcal{B}_I(\pi;\mu)$ under a task-preserving intervention. The construction of Proposition 1 fixes the policy on all inputs and lets the source records include the task state, the nuisance value, and the reference action. The ambiguity remains in the observation mechanism on state and nuisance combinations absent from the source, so more records from the same source do not remove it. Proposition 2 gives identification when factual and intervened continuations can be sampled. A source simulator is one sufficient route to this access, and it is the reason our method reenacts candidate causes in the source rather than mining target failures harder.

Many remedies exist once a bias is known. Domain randomization and augmentation modify training inputs (Tobin et al., 2017; Raileanu et al., 2021). Counterfactual augmentation and data curation modify demonstrations (Ameperosa et al., 2025; Agia et al., 2025; Peng et al., 2023). Active sampling and causal auditing investigate confusing features (Gupta et al., 2023; Gajcin & Dusparic, 2022), and representation methods separate task content from distractors during learning (Fu et al., 2021). Deployed policies can also adapt through latent alignment or online test-time updates without a known shortcut (Yoneda et al., 2022; Xu et al., 2025); these are the external baselines of Section 4.2. RENDER conditions the repair on a candidate supported by failure matching and a response and harm audit. It keeps the base network frozen, fits a residual correction or an outcome-guided runtime adaptor from source data, and needs no target interventions, so a deployed policy can be corrected in place. When the diagnosis fails, the procedure abstains. The Randomize-all experiments compare this focused repair with broad randomization under the same update budget.