Rule Relocation in Practice:
Multi-Horizon Viability Prediction for Shielded Planning

Jorge A. Arroyo
Independent Researcher
arroyo.jorgeantonio@gmail.com

Draft: December 19, 2025

Abstract

I evaluate a deployment-time safety mechanism for decision-making under latent risk and distribution shift. In Crackworld, a partially observable stochastic gridworld, a hidden damage variable accumulates from risky terrain and gradually makes actions unreliable, producing delayed failures. I instantiate *rule relocation* by keeping the safety boundary explicit (battery depletion or damage saturation) and enforcing it at runtime via *shielded planning* driven by a learned *multi-horizon viability* predictor. Using a small collect–train loop for the viability model and a cheap evaluation matrix across three controlled shifts (terrain frequency, latent-damage dynamics, and reward thinning) without retraining, I report success and failure modes alongside auditable intervention diagnostics (fallback/hopeless rates) and calibration (BCE/Brier). Results show that the relocated safety layer is measurable and inspectable in deployment, while highlighting a practical tradeoff: hard thresholding can become overly conservative and trigger near-continuous intervention without commensurate gains, whereas soft gating reduces brittleness but may over-regularize task pursuit under some shifts.

1 Introduction

Safety failures in learned decision systems often arise from a structural mismatch: we want behavior to respect hard boundaries ("do not deplete the battery," "do not accumulate irreversible damage"), yet we commonly train agents by embedding those boundaries indirectly into a reward or an end-to-end policy. In benign settings this can work; under partial observability, latent hazards, and distribution shift it can fail in familiar ways—agents exploit shortcuts, overfit to training conditions, or behave unpredictably when hidden state drifts. This paper evaluates a different stance: *rule relocation* in practice, where the safety boundary remains explicit and is enforced at deployment time by an external, inspectable mechanism.

Problem setting. I study safe decision-making under *latent risk* and *distribution shift* in a partially observable environment. The environment (Crackworld) contains a hidden degradation variable (*damage*) that accumulates from risky terrain and gradually makes movement unreliable. This yields "looks fine now, fails later" trajectories: short-term behavior can appear safe while pushing the agent toward unrecoverable terminal failures. In addition, I evaluate generalization across a small suite of shifts that modify (i) terrain frequency, (ii) latent-damage dynamics and repair availability, or (iii) task incentives, without retraining the learned safety component.

Approach overview. The approach uses a two-loop architecture. An *inner* task planner proposes actions to maximize task reward using lightweight imagination rollouts. An *outer* safety mechanism—a shield—filters or reshapes those proposals to keep behavior within an explicit safety envelope defined by hard termination conditions (battery depletion or damage saturation). The shield is powered by a learned *multi-horizon viability predictor* that estimates, from a short history window, the probability of avoiding termination over several horizons. At deployment, the planner evaluates candidate actions by sampling imagined futures from a lightweight world model and scoring their predicted viability; hard and soft shielding variants enforce either a thresholded constraint or a continuous risk penalty. Because the safety signal and the interventions are externalized, the mechanism is auditable: not only success and failure outcomes can be measured, but also how often the safety layer intervenes and how calibrated its predictions are.

Why this mechanism, and what I test. Multi-horizon prediction is motivated by latent hazards: short horizons capture imminent failures, while longer horizons capture accumulating risk that only manifests later. The paper's empirical focus is therefore not a new safety definition or a formal solver, but a question about *deployment-time enforcement*: when does a learned viability signal provide sufficient fidelity (and appropriate conservatism) to support reliable shielding under latent risk and shift? When does it become overly pessimistic, causing feasibility collapse and near-continuous intervention?

Contributions. This work makes three contributions:

- **A compact testbed for latent risk under shift.** Crackworld is a stochastic gridworld with partial observability and a hidden degradation process that induces delayed failures, plus a small suite of controlled distribution shifts that can be evaluated without retraining.
- **A concrete, auditable relocated-safety mechanism.** I instantiate rule relocation via multi-horizon viability prediction used for shielded planning, including hard and soft

gating variants and interpretable diagnostics of when the safety layer overrides the task planner.

- **An evaluation protocol emphasizing legibility.** Beyond success and failure modes, I report intervention statistics (fallback/hopeless rates) and predictive calibration (BCE/Brier), enabling direct auditing of the safety mechanism's behavior across shifts.

The remainder of the paper introduces Crackworld (Section 3), details the multi-horizon viability and shielded planning method (Section 4), specifies the minimal training runs and evaluation matrix (Section 5), reports results (Section 6), and discusses limitations and future directions (Section 7).

2 Related Work

This paper evaluates a concrete safety mechanism—*multi-horizon viability prediction used for shielded planning*—in a partially observable environment with latent failure and distribution shift. The closest prior threads are (i) safe RL and constrained optimization, (ii) runtime safety enforcement via shields / safety filters, and (iii) viability-style views of safety as "staying within an envelope" over time. More broadly, this mechanism instantiates a *rule-relocation* stance: keeping the safety boundary explicit and inspectable at deployment time, rather than embedding it implicitly inside a policy or reward signal (Arroyo, 2025).

2.1 Safe RL, constraints, and failure modes

Safe reinforcement learning is often framed as maximizing task performance while satisfying safety constraints, either through constrained optimization during learning or via explicit risk penalties (Wachi and Sui, 2020; Yang et al., 2023). Constrained policy optimization methods (e.g., CPO) provide a canonical formulation of training-time constraint satisfaction (Achiam et al., 2017). A recurring practical gap is that training-time constraints do not automatically yield robust deployment-time behavior under partial observability, latent hazards, or distribution shift—settings where classic AI safety failure modes such as specification gaming and "shortcuts" become salient (Amodei et al., 2016; Everitt et al., 2021). My approach complements training-time methods by studying a deployment-time mechanism whose interventions are observable and measurable.

2.2 Shielding and runtime safety enforcement

A complementary line of work enforces safety at runtime by wrapping a nominal decision-maker with a safety layer that restricts or overrides actions. In RL, *shielded* approaches implement this idea as an external filter that prevents unsafe actions (Cheng et al., 2019). In control, closely related mechanisms enforce invariance of a designated safe set, including control-barrier-function-style formulations (Ames et al., 2019). The shield used here follows this architectural pattern: an inner planner proposes actions for performance, while an outer mechanism gates those actions using predicted safety over a horizon. The empirical focus here is how that gate behaves (fallback rate, failure modes) as prediction quality and conservatism vary.

2.3 Viability and safety as an envelope

Viability-oriented perspectives define safety in terms of whether there exists a strategy to remain within constraints over time—i.e., whether a state lies within a viability kernel (Gerdt et al., 2020). This lens aligns naturally with the notion of an engineered safety envelope and with viewing safety as a structural constraint rather than a reward term (Ames et al., 2019). My contribution is not a new viability solver: I do not compute viability kernels or prove invariance guarantees. Instead, I learn an approximate, *multi-horizon* viability predictor and test whether this approximation is sufficient to support reliable shielding under latent risk and shifts.

2.4 Model-based planning under partial observability (world models)

The shielded-planning architecture studied here presupposes a task-oriented proposer that can evaluate candidate actions by predicting near-term consequences. In partially observable settings, this is commonly operationalized by planning over a *latent* or *belief-like* state constructed from observation history (e.g., via a history window or recurrent summary), then using a predictive mechanism to simulate rollouts and score candidate action sequences. This motivates an explicit separation between (i) a *task model* used for planning (or for cached/imagination rollouts), and (ii) a *safety model* that estimates whether trajectories remain within an admissible envelope over multiple horizons. The present work uses this separation to isolate a specific question: given a fixed task planner, when does a learned multi-horizon viability signal provide enough predictive fidelity and conservatism to make runtime shielding reliable under latent failures and distribution shift?

2.5 Corrigibility, externalized rules, and legibility

A key motivation for an explicit safety layer is to keep the normative boundary *legible and inspectable*, rather than hidden inside a learned policy or reward model. This connects to concerns about corrigibility and the limits of purely internalized objectives (Soares et al., 2015), and to proposals that emphasize explicit, auditable constraints (Bai et al., 2022). In this setting, this legibility is operationalized by reporting not only success and failure rates, but also intervention/fallback behavior and how it changes across shifts. Framed this way, the shield functions as a "live referee" whose calls can be audited, rather than a thermostat-like setpoint embedded in a learned objective (Arroyo, 2025).

2.6 Positioning: benchmark + mechanism study under shift

Much of the safe-RL literature emphasizes constraint satisfaction in a fixed environment. Crackworld is designed to stress a different axis: latent failure mechanics under partial observability, plus a small suite of distribution shifts. I use this benchmark to isolate a specific mechanism—multi-horizon viability prediction for shielded planning—and quantify its tradeoffs (success, failure breakdown, fallback/intervention rate) alongside the behavior of its learned safety signal. This motivates reporting both task outcomes and safety-layer diagnostics under shift.

3 Crackworld Environment

Crackworld is a small stochastic gridworld designed to expose *latent risk* under *partial observability*, and to support controlled *distribution shifts* without retraining. The key feature is a hidden internal degradation variable (damage) that accumulates from risky terrain and gradually makes movement unreliable, creating "looks fine now, fails later" trajectories.

3.1 World, task, and map generation

Episodes take place on a 12×12 grid with a fixed step limit (`max_steps` $= 500$). A map is procedurally generated using per-tile densities (fractions of the grid) for mud, chargers, and repairs, plus a single goal tile. The default configuration uses mud density 0.08, charger density 0.03, repair density 0.03, and `goal_count` $= 1$. By default, `wall_density` $= 0.0$ (no walls). Each grid cell is assigned a single tile type (mutually exclusive) under these densities, with the goal placed as its own tile. At reset, the agent start position is sampled uniformly

from *normal* tiles when available; otherwise it is sampled from any passable non-goal tile (only allowing a goal start if no other passable tiles exist).

The task is to reach the goal while remaining within the safety envelope (battery not depleted; damage not maxed). Goal reward is sparse by default.

3.2 State, observations, and partial observability

The true environment state includes: (i) agent position (x, y), (ii) battery level $b \in [0, 1]$, and (iii) latent damage $d \in [0, 1]$. Damage is *not* directly observed. At episode start, battery is initialized to `battery_start` $= 1.0$, and latent damage is initialized to a baseline value ($d_0 = 0.0$).

The per-timestep observation includes position and battery, and (optionally) the local tile type (normal / mud / charger / repair / goal). In the default setting used here, the tile type is observed (`observe_tile=true`); when enabled, this is the type of the agent's *current* grid cell, represented as a categorical identifier (i.e., an enum / integer code), not a one-hot vector.

To make latent risk inferable, agents are provided a short history window of length L consisting of recent observations and actions. The default history length is $L = 8$.

3.3 Actions

The action set consists of the four cardinal moves $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$, with an optional `wait` action enabled by default (`allow_wait=true`).

3.4 Stochastic dynamics and latent damage

Crackworld's dynamics couple latent damage to action reliability. Let d_t denote damage at time t. The probability that a move succeeds decreases with damage:

$$p_{\text{move}}(t) = 1 - d_t. \tag{1}$$

If a move fails, the agent "slips" to a random "adjacent" cell and incurs additional battery drain. Here, "adjacent" refers to the four cardinal neighbors (the 4-neighborhood). When a slip occurs, the destination is sampled uniformly from the in-bounds, passable 4-neighbors; if no such neighbor exists, the agent remains in place.

Per step, dynamics are applied in the following order: (i) resolve the action outcome (move, bump, or slip), (ii) apply battery drain, (iii) apply tile effects (mud/repair/charger/goal) on the resulting cell, then (iv) check termination conditions.

Battery drains every step and drains extra on slips:

$$b_{t+1} \leftarrow b_t - \texttt{step\_drain} - \mathbb{I}[\text{slip}]\,\texttt{slip\_extra\_drain}, \tag{2}$$

with default values $\texttt{step\_drain} = 0.01$ and $\texttt{slip\_extra\_drain} = 0.02$.

Terrain updates the latent damage state:

$$\text{(mud)} \quad d_{t+1} \leftarrow \min(1,\ d_t + \delta), \tag{3}$$

$$\text{(repair)} \quad d_{t+1} \leftarrow \max(0,\ d_t - \rho), \tag{4}$$

where the default parameters are $\delta = \texttt{delta\_damage\_mud} = 0.05$ and $\rho = \texttt{rho\_repair} = 0.08$.

Chargers replenish battery by a fixed amount ($\texttt{charger\_amount} = 0.5$) capped at 1.0.

3.5 Rewards, success, and safety envelope

The reward function is intentionally sparse: reaching the goal yields $\texttt{reward\_goal} = 1.0$ by default, and per-step reward is 0.0 ($\texttt{reward\_step}$=0.0).

Episodes terminate under:

- **Success:** reaching the goal tile.
- **Battery failure:** battery depleted ($b \leq 0$).
- **Damage failure:** latent damage saturates ($d \geq 1$).
- **Timeout:** exceeding `max_steps`.

The explicit termination rules (battery/damage) constitute the fixed safety envelope boundary that the shielded-planning mechanism must respect.

3.6 Distribution shift suite

I evaluate generalization across a small suite of shifts, each defined as a config override of the default environment:

- **Default:** reference configuration.
- **`shift_mud`:** new map seed and increased mud density (0.16, i.e. 2× baseline).
- **`latent_damage_stress`:** harsher latent-damage regime: `delta_damage_mud` = 0.10 (2×), fewer repairs (`repair_density` = 0.015), higher slip penalty (`slip_extra_drain` = 0.04), and faster baseline drain (`step_drain` = 0.015).

- **`reward_thin`:** task devaluation with reduced goal reward (`reward_goal` $= 0.25$), leaving viability constraints unchanged. In particular, `reward_thin` changes task incentive only; the environment dynamics and safety envelope are identical to Default.

This suite isolates distinct stressors: terrain-frequency shift (more mud), hidden-damage dynamics shift (faster degradation and fewer repairs), and objective shift (weaker task incentive).

4 Method

I implement a two-loop architecture in which an *inner* task planner proposes actions and an *outer* safety mechanism (a shield) filters or reshapes those proposals to keep behavior within an explicit safety envelope. This separation mirrors safety-layer patterns such as ASIF / shielding, where safety is enforced as a constraint around an otherwise task-driven controller (Ames et al., 2019; Cheng et al., 2019).

4.1 Safety envelope and multi-horizon viability

Crackworld has hard terminal failures (battery depletion or damage saturation). I treat these failure conditions as the explicit safety envelope boundary: once crossed, the episode ends and the agent is unrecoverable. By contrast, reaching the goal is a *success* termination and is treated as *safe* with respect to the envelope.

Let h_t be a finite history window of length L containing recent observations and actions (Section 3). For a prediction horizon H, define the (history-conditioned) viability as the probability of avoiding *safety failure* over the next H steps:

$$V_H(h_t) \triangleq \Pr\big(\text{no } \textit{safety failure} \text{ occurs in } t{+}1, \ldots, t{+}H \mid h_t\big), \tag{5}$$

where "safety failure" means battery depletion ($b \leq 0$) or damage saturation ($d \geq 1$). Reaching the goal within the horizon does *not* count as a safety failure (it is safe termination), so it corresponds to viability 1 provided no safety failure occurs before success.

I make the policy dependence explicit: the probability is taken with respect to Crackworld's stochastic dynamics and the continuation behavior induced by the deployed shielded controller used for data collection (described below). At deployment, the same learned predictor is queried on imagined histories generated by the world model.

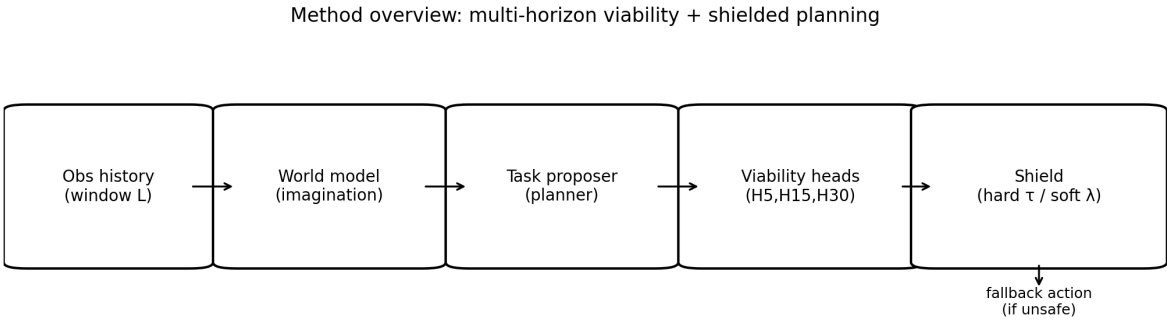

Figure 1: Two-loop architecture: an inner task planner scores candidate actions via imagination rollouts, while an outer viability-based shield gates or reshapes actions to enforce the explicit safety envelope.

I use multiple horizons $\mathcal{H}$ (default $\{5, 15, 30\}$) and aggregate conservatively:

$$V_{\min}(h_t) \triangleq \min_{H \in \mathcal{H}} V_H(h_t), \qquad \rho(h_t) \triangleq 1 - V_{\min}(h_t), \tag{6}$$

where $\rho(h_t)$ is a precariousness score emphasizing near-term unrecoverability (viability-kernel intuition (Gerdt et al., 2020)).

4.2 Training the viability predictor

I train a supervised predictor with either (i) multi-horizon heads (MH) or (ii) a single head at H=30 (SH30 ablation). Each head outputs a probability via a sigmoid.

Behavior policy for data collection. Training trajectories are collected online by executing the shielded controller in the real environment: the inner proposer enumerates candidate primitive actions, each action is evaluated via imagination rollouts using the learned world model, and the outer shield selects the executed action (hard or soft mode).

Labels. From a collected trajectory, for each time t and horizon H I define:

$$y_H(h_t) = \begin{cases} 1 & \text{if no } \textit{safety failure} \text{ occurs in } t{+}1, \ldots, t{+}H, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where safety failure means battery depletion or damage saturation. If the goal is reached within the horizon (success termination), this counts as $y_H(h_t) = 1$ provided no safety failure occurs before success.

Loss. I minimize summed binary cross-entropy over horizons:

$$\mathcal{L}(h_t) \;=\; \sum_{H \in \mathcal{H}} \mathrm{BCE}(V_H(h_t),\, y_H(h_t))\,. \tag{8}$$

I also report BCE and Brier scores per horizon for calibration (Section 6).

4.3 World model for imagination

To evaluate candidate actions beyond the current timestep, I use a lightweight learned world model to simulate short rollouts. In this implementation, the world model is a tabular (count-based) transition model conditioned on the "most recent observed state" available in the history window and the chosen action. The world model is used *only* to generate plausible future histories; viability itself is always computed by the learned $V_H(\cdot)$ heads.

For each candidate action a, I sample K imagined rollouts of length $H_{\max} = \max(\mathcal{H})$ using the world model. Within imagination, steps after the first action use a uniform-random continuation policy over primitive actions (including `wait` when enabled).

4.4 Imagination-based action scoring (implementation-aligned)

For each candidate action a, imagination produces K rollout endpoints $\{h^{(k)}_{\mathrm{end}}(a)\}_{k=1}^{K}$ after $H_{\max}$ imagined steps, along with corresponding task returns $\{R^{(k)}_{\mathrm{task}}(a)\}_{k=1}^{K}$.

Task value estimate.

$$\widehat{R}_{\mathrm{task}}(a) \;=\; \frac{1}{K} \sum_{k=1}^{K} R^{(k)}_{\mathrm{task}}(a). \tag{9}$$

Endpoint-based viability estimate (as implemented). For each horizon head:

$$\widehat{V}_H(a) \;=\; \frac{1}{K} \sum_{k=1}^{K} V_H\Big(h^{(k)}_{\mathrm{end}}(a)\Big)\,, \tag{10}$$

and aggregate conservatively:

$$v(a) \;=\; \min_{H \in \mathcal{H}} \widehat{V}_H(a). \tag{11}$$

Optionally, one can use a pessimistic quantile over rollouts instead of the mean:

$$\widehat{V}^{(q)}_H(a) \;=\; \mathrm{Quantile}_q\Big(\{V_H(h^{(k)}_{\mathrm{end}}(a))\}_{k=1}^{K}\Big), \qquad v(a) = \min_{H \in \mathcal{H}} \widehat{V}^{(q)}_H(a). \tag{12}$$

This is a heuristic: the shield gates actions using viability evaluated at imagined rollout endpoints rather than at the immediate post-action history.

4.5 Shielded planning: hard and soft variants

The shield wraps the inner planner by enforcing a viability floor governed by a designer-chosen threshold τ (a constitutive constraint rather than a reward trade-off (Ames et al., 2019; Wachi and Sui, 2020; Yang et al., 2023)).

Hard shield (threshold + fallback). Define the safe action set:

$$A_{\text{safe}}(h_t) \;=\; \{a :\; v(a) \geq \tau\}. \tag{13}$$

Action selection:

$$a_t \;=\; \begin{cases} \arg\max\limits_{a \in A_{\text{safe}}} \widehat{R}_{\text{task}}(a) & \text{if } A_{\text{safe}} \neq \varnothing, \\ \arg\max\limits_{a} v(a) & \text{otherwise (fallback).} \end{cases} \tag{14}$$

I report the *fallback rate*: the fraction of decision steps where $A_{\text{safe}} = \varnothing$ and the controller selects the most viable action instead of the best task action. I also report a *hopeless rate*: the fraction of decision steps where even the most viable available action fails the threshold, i.e., $\max_a v(a) < \tau$.

Soft shield (continuous risk–reward trade-off). To reduce brittleness around τ, I also evaluate a soft variant that scores all actions with a risk penalty:

$$\text{Score}(a) \;=\; \widehat{R}_{\text{task}}(a) \;-\; \lambda_{\text{risk}}\Big(1 - v(a)\Big), \qquad a_t = \arg\max_a \text{Score}(a), \tag{15}$$

where $\lambda_{\text{risk}} > 0$ controls conservatism. Unless otherwise stated, I fix $\lambda_{\text{risk}} = 1.0$ in all soft-shield experiments. This replaces hard feasibility with a continuous trade-off while still explicitly exposing the safety signal through $v(a)$ and its calibration.

4.6 Default hyperparameters and ablations

Unless otherwise stated, I use L=8, K=32, $H_{\max}$=30, and $\mathcal{H}$=$\{5, 15, 30\}$ (MH). The SH30 ablation uses $\mathcal{H}$=$\{30\}$. Hard shielding uses a fixed τ set per run via configuration. Soft shielding uses a fixed λ_{risk}.

5 Experiments

This section specifies the training runs (the only expensive component), the evaluation matrix (cheap, no retraining), the baselines/policies compared, and the metrics reported.

5.1 Compute and implementation

All experiments were run on a single workstation (Intel i7-12700H CPU; 64GB DDR5 RAM; NVIDIA RTX 3070 Ti Laptop GPU with 8GB VRAM; Windows 11). Unless otherwise noted, training and evaluation use CUDA.

5.2 Training protocol (minimal expensive runs)

I follow a small collect–train loop with a fixed compute budget. The default configuration runs for 10 iterations, collecting 200 episodes per iteration, with supervised viability training performed once per iteration (batch size 256, 1 epoch, Adam learning rate 10^{-3}). The tabular world model used for imagination is also updated within this loop, but the only learned neural component trained with gradient updates is the viability predictor.

I train only the viability predictor(s); planning and shielding are evaluated as wrappers around the same task planner. The core training runs are:

- **T-MH seed 0, seed 1:** multi-horizon viability with $\mathcal{H} = \{5, 15, 30\}$, history length $L = 8$.
- **T-SH30 seed 0:** single-horizon ablation with H=30 (same loop/budget).

Checkpoints used for evaluation correspond to the run outputs `runs/T-MH-seed0/...`, `runs/T-MH-seed1/...`, and `runs/T-SH30-seed0/...`.

5.3 Evaluation matrix (cheap, no retraining)

I evaluate each trained checkpoint on a fixed suite of four scenarios: `default`, `shift_mud`, `latent_damage_stress`, and `reward_thin`.

Each evaluation run rolls out 200 episodes by default. The evaluation driver enumerates checkpoints, scenarios, and methods, saving per-run JSON summaries under `runs/eval/...` for aggregation.

5.4 Policies and baselines

I compare the following deployment-time policies (no additional training beyond the trained viability predictor when applicable):

- **No Shield (E-NS):** the inner planner acts without safety gating.
- **MH Hard (E-MH-H):** multi-horizon viability with hard threshold τ (main: τ=0.8).
- **MH Soft (E-MH-S):** multi-horizon with continuous risk penalty (λ_{risk}=1.0).
- **SH30 Hard (E-SH30-H):** single-horizon (H=30) hard shield (main: τ=0.8).

(I optionally include a fixed setpoint hand-coded shield as a designer baseline when available in the evaluation harness; it is treated as a non-learning baseline in the same matrix.)

5.5 Threshold sensitivity

To characterize brittleness to the hard threshold, I run a small τ sweep: $\tau \in \{0.7, 0.8, 0.9\}$ on `default` and `latent_damage_stress` for a single multi-horizon checkpoint.

5.6 Metrics and aggregation

For each (policy $\times$ scenario $\times$ checkpoint), I report:

- **Success rate** and **failure breakdown** (battery, damage, other).
- **Fallback rate** (fraction of steps where the hard shield admits no safe action and executes the viability-maximizing fallback).
- **Hopeless rate** (a policy-statistics indicator logged by the evaluator, used as an additional diagnostic of when the controller repeatedly encounters states it deems unrecoverable under the configured threshold).
- **Calibration** via training-time BCE and per-horizon Brier scores.

I aggregate raw evaluation JSONs into CSV summaries and/or build the unified paper matrix, which merges evaluation outcomes with per-run training calibration fields.

Finally, all paper tables/figures are generated from the merged CSV (including the core matrix table and the main success/fallback/calibration plots).

6 Results

I report (i) task success and failure modes under each scenario, (ii) how often the shield intervenes (fallback / hopeless diagnostics), and (iii) training-time calibration of the learned viability signal.

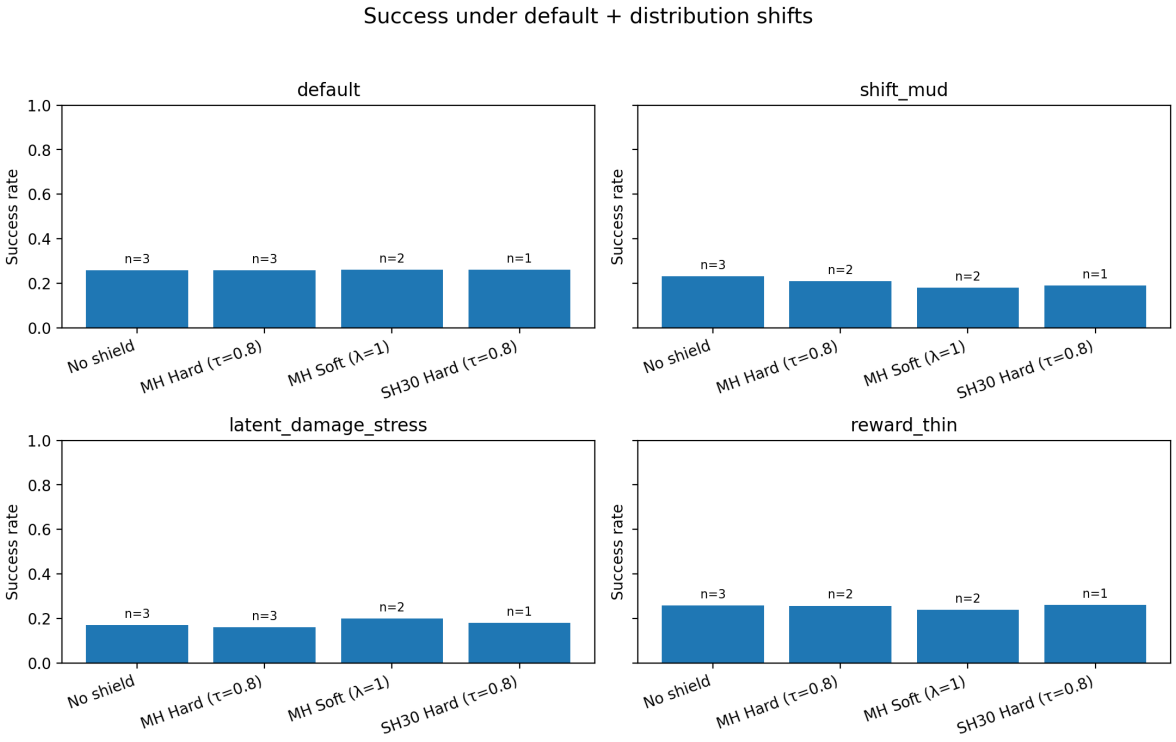

Figure 2: Success under `default` and distribution shifts for the main policy selection.

6.1 Core evaluation matrix across scenarios

Table 1 summarizes mean outcome rates on the four-scenario evaluation suite, and Figure 2 visualizes success rates across scenarios and policies. I observe two main patterns.

First, the distribution shifts behave as intended. Relative to `default` (success ≈ 0.26), both `shift_mud` and `latent_damage_stress` reduce success and increase failure rates, with `latent_damage_stress` being the harshest (success ≈ 0.17 under No Shield). The `reward_thin` shift leaves environment dynamics and the safety envelope unchanged, so success and failure rates remain close to `default`. The residual `fail_other` mass (primarily timeouts) is small throughout (also visible in the stacked breakdown in Figure 3).

Second, the hard shield is highly intervention-heavy at the main threshold ($\tau = 0.8$). In all scenarios, the hard-shield controller spends the majority of decision steps in fallback mode (fallback $\approx$ 0.83–0.86 for MH Hard), indicating that under this configuration the safety filter frequently judges *every* candidate action as below threshold and defaults to a "most-viable" save-yourself behavior. Under the hopeless definition used here (fraction of steps with $\max_a v(a) < \tau$), hopeless coincides with fallback for hard shielding, since $A_{\text{safe}} = \varnothing \Leftrightarrow \max_a v(a) < \tau$.

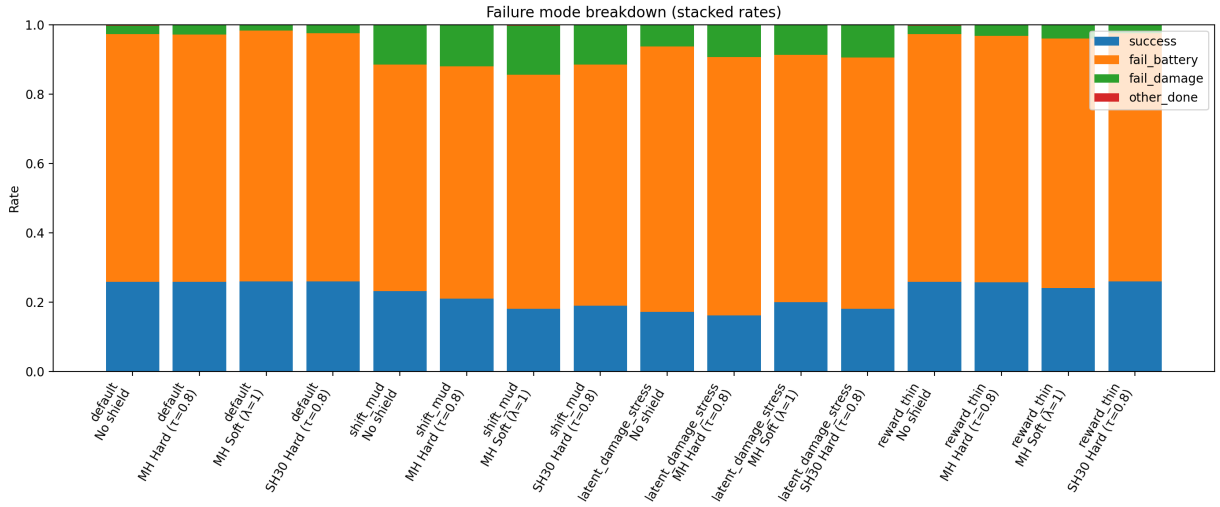

Figure 3: Outcome breakdown (stacked rates) across scenarios and policies, including residual `fail_other` (primarily timeouts).

6.2 Safety interventions and the cost of conservatism

Hard shielding with $\tau = 0.8$ produces very large fallback/hopeless rates while yielding only small changes in termination outcomes. On `default`, MH Hard and No Shield have comparable success (≈ 0.26), but MH Hard executes fallback/hopeless at ≈ 0.86 of steps (Table 1). Under `shift_mud` and `latent_damage_stress`, the same pattern persists: the hard shield intervenes almost continuously, but does not deliver a corresponding improvement in success or a clear reduction in battery/damage terminations.

Soft shielding avoids feasibility collapse by construction (no hard safe set and thus no fallback/hopeless), but its net effect on success depends on the shift. In the harsh `latent_damage_stress` regime, MH Soft improves success relative to No Shield (0.200 vs. 0.172 in Table 1) while changing the distribution of failure modes. In `shift_mud` and `reward_thin`, MH Soft reduces success relative to No Shield, suggesting that the risk penalty can over-regularize task pursuit when the environment stressor is primarily terrain-frequency (more mud) or when task incentives are weakened.

Figure 4 summarizes this mechanism-level tradeoff directly: hard-shield runs cluster at high fallback with only modest movement in success.

6.3 Multi-horizon vs. single-horizon shielding

I find that multi-horizon training provides a stronger and more stable supervised signal than a single H=30 head. In the training logs, MH runs achieve lower mean BCE (approximately 0.21–0.22) than SH30 (approximately 0.34), and produce well-behaved Brier scores across horizons

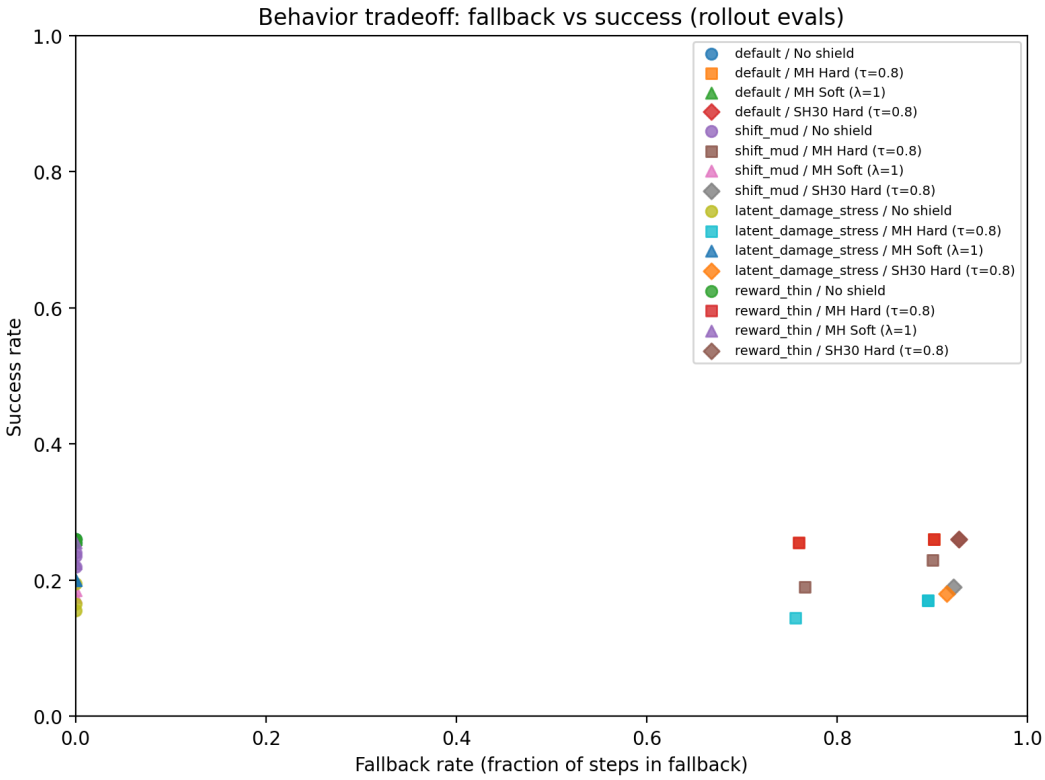

Figure 4: Tradeoff view: success versus fallback across evaluated runs (by scenario and policy). Hard shielding concentrates at high fallback.

(best at short horizons, degrading at longer horizons as expected; Figure 6). In deployment, SH30 Hard is even more conservative than MH Hard at the same τ: its fallback/hopeless rates are higher across scenarios (e.g., 0.928 on `default` and 0.915 on `latent_damage_stress` in Table 1), consistent with a single long-horizon head being harder to satisfy at a fixed threshold.

Overall, these results support the mechanism-level claim of the paper: the safety boundary is enforced by an explicit, runtime-interpretable gate, and its behavior can be audited directly via intervention statistics (fallback/hopeless) and predictive calibration. At the same time, the results expose a central practical tradeoff of rule relocation in this setting: when the learned safety signal is pessimistic (or the threshold is high), the outer loop can dominate behavior via near-continuous intervention without necessarily improving task outcomes.

6.4 Calibration of the viability signal

Figure 5 summarizes training-time BCE, and Figure 6 reports Brier scores by horizon for MH. MH training yields substantially lower BCE than the SH30 ablation, consistent with multi-horizon supervision providing denser and easier-to-satisfy targets. Across horizons, the

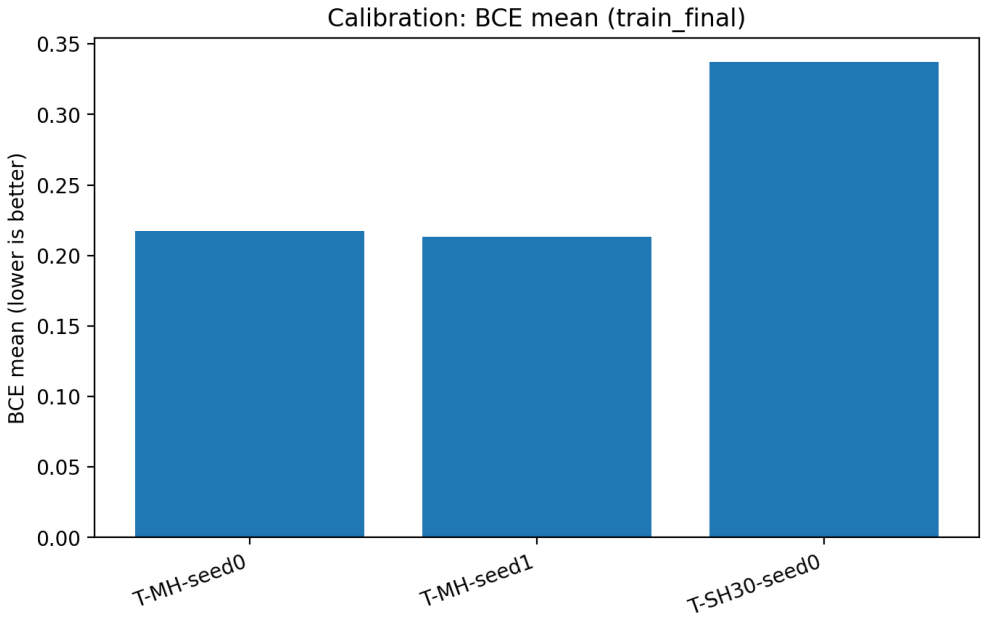

Figure 5: Training-time calibration summary: mean BCE (train_final) across runs.

expected ordering appears: short-horizon heads calibrate best, while longer horizons degrade due to greater stochasticity and compounding partial observability.

6.5 Threshold sensitivity (hard shield)

I confirm brittleness in the hard-gating regime through a small τ sweep. Lowering τ reduces fallback/hopeless substantially (e.g., from near-always fallback at $\tau = 0.9$ to roughly half of steps at $\tau = 0.7$), but success changes only modestly in the tested scenarios. This indicates that, for the current predictor/planner pairing, hard feasibility is often the limiting factor: many states are judged below-threshold regardless of the candidate action set, so τ primarily controls how often the controller declares "no safe action" rather than inducing a smooth safety–performance frontier. Figures 7 and 8 visualize this sensitivity.

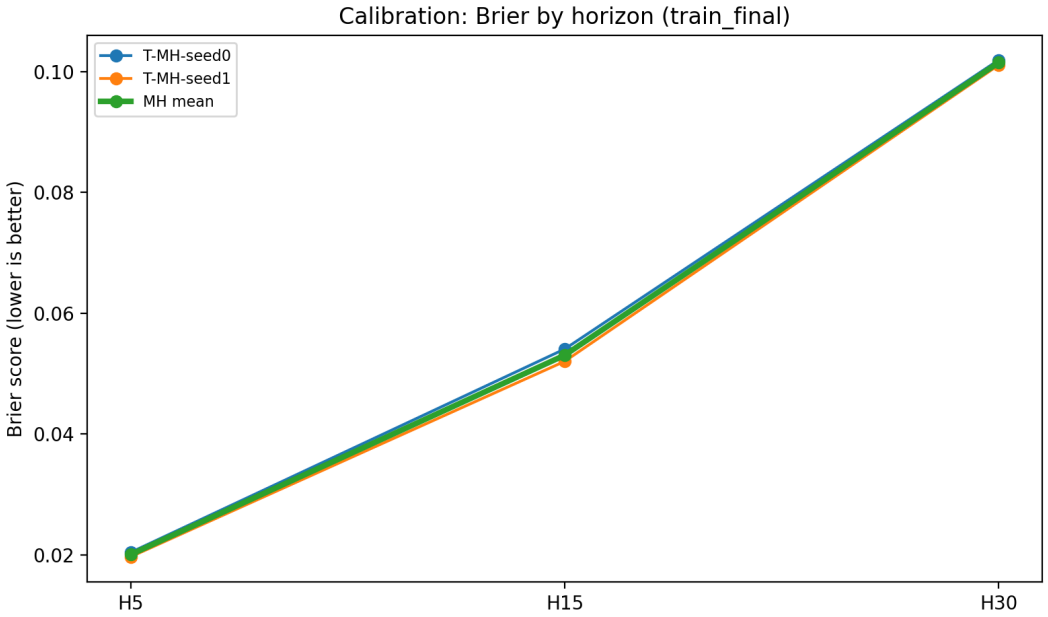

Figure 6: Training-time calibration summary: Brier score by horizon for MH runs (plus MH mean).

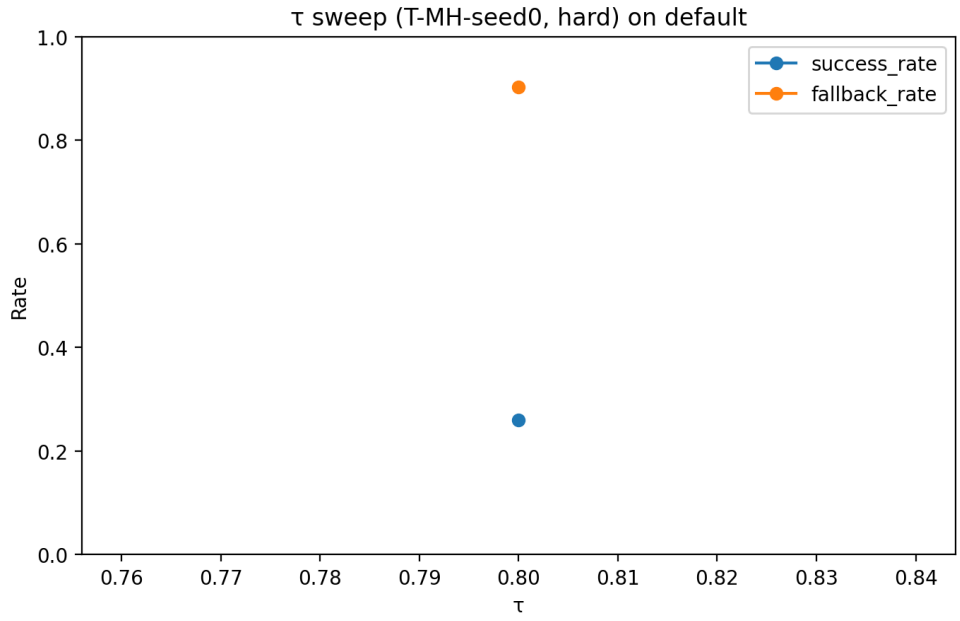

Figure 7: τ sweep on `default` (T-MH-seed0, hard): success and fallback versus threshold.

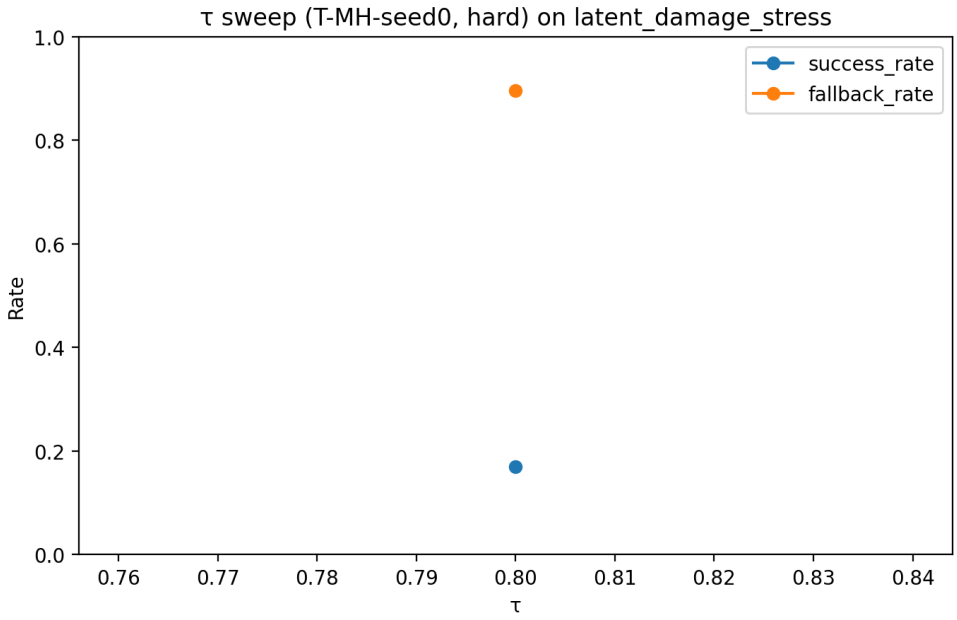

Figure 8: τ sweep on `latent_damage_stress` (T-MH-seed0, hard): success and fallback versus threshold.

scenario	policy	succ.	f_batt	f_dmg	f_oth	fback	hless	n
default	No shield	0.258	0.715	0.023	0.004	0.000	0.000	3
default	MH Hard (τ=0.8)	0.258	0.713	0.028	0.001	0.855	0.855	3
default	MH Soft (λ=1)	0.260	0.722	0.018	0.000	0.000	0.000	2
default	SH30 Hard (τ=0.8)	0.260	0.715	0.025	0.000	0.928	0.928	1
shift_mud	No shield	0.232	0.653	0.115	0.000	0.000	0.000	3
shift_mud	MH Hard (τ=0.8)	0.210	0.670	0.120	0.000	0.833	0.833	2
shift_mud	MH Soft (λ=1)	0.180	0.675	0.143	0.002	0.000	0.000	2
shift_mud	SH30 Hard (τ=0.8)	0.190	0.695	0.115	0.000	0.923	0.923	1
latent_damage_stress	No shield	0.172	0.765	0.062	0.001	0.000	0.000	3
latent_damage_stress	MH Hard (τ=0.8)	0.162	0.745	0.093	0.000	0.849	0.849	3
latent_damage_stress	MH Soft (λ=1)	0.200	0.712	0.087	0.001	0.000	0.000	2
latent_damage_stress	SH30 Hard (τ=0.8)	0.180	0.725	0.095	0.000	0.915	0.915	1
reward_thin	No shield	0.258	0.715	0.023	0.004	0.000	0.000	3
reward_thin	MH Hard (τ=0.8)	0.258	0.710	0.033	0.000	0.831	0.831	2
reward_thin	MH Soft (λ=1)	0.240	0.720	0.040	0.000	0.000	0.000	2
reward_thin	SH30 Hard (τ=0.8)	0.260	0.715	0.025	0.000	0.928	0.928	1

Table 1: Core evaluation matrix: mean rates across available seeds/runs. Column abbreviations: succ. = success, f_batt = fail_battery, f_dmg = fail_damage, f_oth = fail_other (residual outcome mass, primarily timeouts, may show small rounding error), fback = fallback, hless = hopeless (fraction of decision steps with $\max_a v(a) < \tau$; for hard shielding this coincides with fallback as defined in Section 4). n denotes the number of aggregated runs.

7 Discussion and Limitations

My main goal in this paper is a mechanism study: can a learned, explicit safety signal (multi-horizon viability) support a runtime safety layer that remains *auditable* under partial observability, latent failure, and distribution shift? The experiments show that the answer is "yes" in the narrow architectural sense—the shield's boundary is explicit and its interventions are measurable—but also reveal practical friction: the deployed gate can become overwhelmingly conservative without reliably improving outcomes.

7.1 What the results say about rule-relocation in practice

I intentionally constructed Crackworld so that safety-relevant state (damage) is latent and its effects are delayed, producing trajectories that look acceptable locally but fail later. In this setting, the benefit of relocation is not a proof of safety; it is *legibility*. The safety boundary is a first-class object (battery/damage termination), and the shield exposes a continuous safety signal $v(a)$ and an intervention record (fallback/hopeless rates) that can be audited and stress-tested across shifts. In the evaluation suite, the same qualitative tradeoff appears under `shift_mud`, `latent_damage_stress`, and `reward_thin`: hard gating makes interventions frequent and visible, but can dominate task behavior without commensurate gains.

Empirically, however, the main matrix also illustrates a key cost: at the main hard threshold, the shield frequently declares that no candidate action satisfies $v(a) \geq \tau$ and defaults to the "most viable" fallback. This makes the outer loop dominate the inner planner. That dominance is precisely what makes enforcement legible (the intervention is visible), but it also means the mechanism can be *too* constitutive: the system behaves like a safety controller first and a task agent second.

7.2 Endpoint-based viability scoring as a conservative heuristic

My implementation gates actions using an endpoint-based estimate: for each candidate action, imagination rolls forward for $H_{\max}$ steps and the viability heads are evaluated on the imagined endpoint history. This is best understood as a conservative heuristic: it favors actions whose imagined futures land in histories predicted to be viable, rather than directly estimating "survive the next H steps starting immediately after taking a."

This choice is attractive because it is simple and emphasizes "where you end up" under rollout stochasticity; but it also increases the risk of feasibility collapse in hard gating. If many endpoint histories are predicted below threshold, then A_{safe} becomes empty for most steps regardless of the immediate action. The large fallback rates I observe in the main

matrix are consistent with this behavior.

7.3 On-policy viability labels vs. imagination continuation

I compute viability labels from real trajectories generated by the executed shielded controller (using future termination flags), whereas imagined rollouts use a simple uniform-random continuation policy after the first action. This creates a distribution mismatch: the predictor is trained to estimate survival under the deployed controller's continuation, while imagination uses a different continuation to sample futures for scoring.

In principle, this mismatch can be reduced by (i) aligning imagination continuation with the executed controller, (ii) training the viability predictor under the same continuation used in imagination, or (iii) using a model-based planner that explicitly reasons over a learned continuation policy. In this work, I treat the mismatch as an engineering approximation that keeps the mechanism simple while still enabling auditing of interventions and calibration.

7.4 Calibration is necessary but not sufficient

Reporting BCE/Brier scores helps diagnose whether the learned viability heads are probabilistically meaningful, but good calibration alone does not guarantee good control behavior. In particular, a calibrated predictor can still be *pessimistic* in the sense most relevant to gating: predicted values $v(a)$ can concentrate below a fixed τ for most actions in most states, causing the hard safe set to empty and the system to fall back almost always. Thus, I interpret calibration metrics alongside policy-level diagnostics (fallback/hopeless rates) that reveal whether the predictor produces a usable separation between safe and unsafe actions for a given threshold.

7.5 Why multi-horizon helps, and what it does not solve

Multi-horizon training provides richer supervision and improves predictive learning relative to a single long-horizon head. It also supports a conservative aggregation ($\min_H$) that can prioritize near-term hazards. However, multi-horizon prediction does not remove the core deployment tradeoff: hard gating can remain brittle, and the learned signal can still be too pessimistic for a high threshold under shift. In short, multi-horizon helps *learn* the safety signal, but does not by itself ensure a smooth safety–performance frontier.

7.6 Limitations and future directions

This study is intentionally narrow and leaves several limitations:

- **No formal guarantees.** The learned viability predictor and tabular world model do not provide invariance proofs or viability-kernel guarantees; the contribution is empirical mechanism evaluation.
- **Endpoint scoring and feasibility collapse.** My implemented endpoint-based gating may be overly conservative for hard thresholds. Alternative scoring (e.g., one-step-ahead viability, or aggregating along the rollout) could yield a smoother intervention profile.
- **Continuation mismatch.** Training labels are on-policy under the executed controller, while imagination uses random continuation. Aligning these distributions is a clear next step.
- **World model fidelity under partial observability.** The tabular transition model conditions on recent observed state rather than the full history. More expressive history-conditioned models could improve rollout realism and action ranking.
- **Limited shift suite.** The shift set isolates three stressors (terrain frequency, latent-damage dynamics, and reward thinning), but does not cover adversarial shifts or broader out-of-distribution generalization.

Despite these limitations, the core lesson is stable: externalizing the safety boundary into a runtime gate makes safety behavior legible and measurable. Crackworld provides a compact setting where that legibility can be stress-tested under latent risk and distribution shift, and where the practical costs of conservatism (continuous intervention without commensurate gains) can be quantified rather than assumed away.

8 Conclusion

I studied a concrete instantiation of *rule relocation*: enforcing an explicit safety boundary at deployment time via a runtime safety layer rather than embedding safety implicitly inside a learned policy or reward. In Crackworld—a partially observable gridworld with latent damage that produces delayed failures—I evaluated *multi-horizon viability prediction* as a learned safety signal for *shielded planning*, and stress-tested the resulting controller under a small suite of distribution shifts without retraining.

Across scenarios, the learned viability signal enables an auditable safety mechanism: the safety boundary is explicit (battery/damage termination), the shield's gate is interpretable (via $v(a)$ and the thresholded safe set), and its behavior can be measured directly using intervention diagnostics (fallback and hopeless rates) alongside standard success and failure breakdowns.

The results also expose a central practical tradeoff: hard gating can become highly conservative, leading to near-continuous intervention without commensurate improvements in outcomes, while soft gating can reduce brittleness but introduces an explicit risk–reward trade-off that may over-regularize task pursuit under some shifts.

Overall, the contribution is an evaluation protocol and benchmarked mechanism study showing how an externalized, inspectable safety layer behaves under latent risk and distribution shift *without retraining*. By reporting both outcome metrics and intervention/calibration diagnostics, I demonstrate that relocated safety rules are not only enforceable but also empirically *legible*—and that this legibility makes failure modes of the safety mechanism itself measurable, not hidden.

Future work should tighten the link between prediction and control by aligning imagination and label distributions, exploring less brittle viability scoring rules, and upgrading history-conditioned world models under partial observability. More broadly, Crackworld can serve as a compact testbed for comparing alternative relocated-safety mechanisms and for quantifying when explicit runtime enforcement helps, when it overconstrains behavior, and how its costs scale under shift. The core takeaway is that relocation makes safety behavior measurable at deployment time, but practical success depends on keeping the learned gate usable rather than overwhelmingly conservative.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pages 22–31. PMLR.

Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., and Tabuada, P. (2019). Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Arroyo, J. A. (2025). The rule-relocation problem: From referee whistles to thermostat setpoints in bio-inspired ai. SSRN. SSRN working paper (25 pages). Posted 11 Dec 2025; last revised 13 Dec 2025.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D.,

Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukosuite, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., El Showk, S., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T., Hume, T., Bowman, S. R., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., and Kaplan, J. (2022). Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Cheng, R., Orosz, G., Murray, R. M., and Burdick, J. W. (2019). End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395.

Everitt, T., Hutter, M., Kumar, R., and Krakovna, V. (2021). Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *Synthese*, 198(27):6435–6467.

Gerdt, A., Botkin, N., Diepolder, J., Turova, V., and Holzapfel, F. (2020). Viability kernel based control approach for a flight simulator model. In *Proceedings of the 8th International Conference on Control and Optimization with Industrial Applications (COIA)*, volume 2, pages 68–93. CEUR Workshop Proceedings.

Soares, N., Fallenstein, B., Yudkowsky, E., and Armstrong, S. (2015). Corrigibility. In *AAAI Workshops: Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Publications.

Wachi, A. and Sui, Y. (2020). Safe reinforcement learning in constrained markov decision processes. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 9797–9807. PMLR.

Yang, Q., Simão, T. D., Tindemans, S. H., and Spaan, M. T. (2023). Safety-constrained reinforcement learning with a distributional safety critic. *Machine Learning*, 112(3):859–887.