Expert-Supported Reinforcement Learning for Automated Environment Anomaly Discovery

Aurélien Chambon[1], **Nicolas Grelier**[1]
{aurelien.chambon,nicolas.grelier}@pullupent.com

[1]**PulluP Entertainment**

Abstract

The problem of detecting anomalies in interactive environments is crucial for validating complex simulators or virtual game worlds. We formalize anomalies as states where a learned policy achieves realized higher return under the real dynamics than an expert policy assumed optimal under nominal anomaly-free dynamics. To better expose such discrepancies, we propose an expert-supported reinforcement learning framework that incorporates expert information directly into the learning objective through auxiliary losses. We introduce and compare several expert-referenced loss functions, integrated progressively through a scheduled coefficient to benefit from expert structure while retaining the freedom to find exploits whenever the environment violates nominal assumptions, signaling anomaly discovery. We evaluate the approach and variations through a case study on automated bug detection in virtual worlds via goal-conditioned navigation tasks, using a navmesh agent as the expert. Experiments across a research environment and a commercial game show that our method improves both task solvability, anomaly-discovery rates and state coverage.

1 Introduction

Creating reliable virtual worlds such as simulators or video games requires detecting and resolving environment anomalies before deployment. While unit tests are essential, many failures only surface through interaction. For instance, a physics engine might pass all deterministic test cases but still allow a player-controlled character to get stuck in rare terrain configurations or trigger unintended collisions under unusual movement patterns. To uncover such situations, a purely random agent is insufficient in complex worlds and tasks, as uncovering relevant issues often demands agents that understand the environment, closer to how users actually behave (Sestini et al., 2024).

Reinforcement Learning (Sutton & Barto, 1998) has been explored for environment anomaly discovery (e.g. in automated bug finding), mainly through imitation or curiosity-driven approaches (Gordillo et al., 2021; Bergdahl et al., 2020). However, imitation methods can over-anchor policies to the expert's support (Sestini et al., 2025), and suffer from distribution mismatch (Biré et al., 2025), while curiosity-based methods exhibit strong environment-scale dependence.

In this paper, we formalize environment anomalies as states where an RL policy outperforms an expert optimal under nominal dynamics, a formalization not previously considered to our knowledge. Based on this prior, we propose an expert-supported RL framework that progressively incorporates expert information into the objective to help uncover environment anomalies. We present and compare several loss designs (action-space alignments through Mean Square Error and Negative Log-Likelihood, value supervision, and expert-referenced advantage targets) of the integration of expert support across a research-oriented open source environment and a commercial video game, and show that our method improves task solvability, anomaly-discovery rates and state coverage.

2 Related work

Prior work on anomaly detection thought RL has largely focused on identifying data anomalies, with applications such as network intrusion detection, or recognition of potential cyber-attacks (Arshad et al., 2022; Zhou et al., 2021). In contrast, we define anomalies as expert-model violations in the environment's dynamics, detectable when an RL policy outperforms an expert optimal under nominal assumptions.

One might attempt to discover such environment anomalies using a purely random agent. However, certain anomalies can only be revealed once the agent understands the dynamics of the world, especially in settings with complex interaction loops (Sestini et al., 2024). Detecting these anomalies therefore requires an intelligent and adaptive agent, motivating our use of RL (Sutton & Barto, 1998), especially for its exploration mechanism.

Auxiliary objectives are a well-established means to improve representation learning and performance in RL. He et al. (2022) demonstrated the promise of automatically discovering effective auxiliary losses, while emphasizing the need for careful deployment, as certain auxiliary objectives were shown to degrade performance. Closer to our setting, Álvaro Serra-Gomez et al. (2025) proposed aligning a learned policy with the planner's action distribution, treating the expert planner as a prior and presenting an explicit trade-off between return maximization and trajectory proximity to the planner. Our approach is conceptually related but differs in purpose: We introduce expert information through scheduled auxiliary losses to uncover environment anomalies.

In applied domains of environment anomaly discovering, automated gameplay testing methods incorporate imitation or curiosity-driven exploration (Sestini et al., 2024; Gordillo et al., 2021). However, imitation methods can over-anchor policies to the expert's support (Sestini et al., 2025), and suffer from distribution mismatch (Biré et al., 2025). Furthermore, curiosity-based methods exhibit strong environment-scale dependence, making them difficult to scale. To address these limitations, we propose progressively incorporating expert signals into the loss to uncover environment anomalies. To our knowledge, no prior work defines environmental anomalies as outperforming the expert, nor does leveraging scheduled auxiliary losses to expose these anomalies across empirical use cases.

3 Expert-Supported Reinforcement Learning for Automated Anomaly Discovery

3.1 RL Setup, Expert, and Environment Anomalies

For clarity, we consider an RL agent interacting within a deterministic environment, though we justify in **??** that our framework remains extensible to stochastic dynamics. We model a POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}_{\text{real}}, R, \mathcal{O}, \gamma, H \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, R the reward function, $\mathcal{O}$ the observation space, γ the discount factor, and H the episode horizon. The transition function $\mathcal{T}_{\text{real}}$ denotes the actual environment dynamics, which may contain anomalies. We denote by $\mathcal{T}_{\text{nom}}$ the nominal transition model of an anomaly-free version of the environment.

Assumption on Deterministic Expert Optimality under nominal dynamics. We assume access to an expert deterministic policy $\pi^\star$ optimal in the nominal environment governed by $\mathcal{T}_{\text{nom}}$:

$$\forall s \in \mathcal{S},\ V^{\pi^\star}_{\text{nom}}(s) = V^{\star}_{\text{nom}}(s) \tag{1}$$

where $V^{\star}_{\text{nom}}$ denotes the optimal value function under the nominal dynamics.

Given the previous **Assumption**, we define an *environment anomaly* as any state $s \in \mathcal{S}$ for which a policy π attains realized higher return under the real dynamics than the expert, i.e.

$$G^{\pi}_{\text{real}}(s) \ > \ G^{\pi^\star}_{\text{real}}(s) \tag{2}$$

where $G^{\pi}_{\text{real}}(s)$, $G^{\pi^\star}_{\text{real}}(s)$ denote the return (accumulated reward) of a rollout following the learned or expert policy respectively.

Whenever this situation occurs, it indicates that the agent following π has uncovered an anomaly and is exploiting the discrepancy between $\mathcal{T}_{\text{nom}}$ and $\mathcal{T}_{\text{real}}$. Since all interaction occurs under the real dynamics, further notations drop the subscript real and write V^{π}, $V^{\pi^\star}$, G^{π} and $G^{\pi^\star}$.

3.2 Motivation for Integrating Expert Signals into the Loss Rather than into Demonstrations or Rewards

Recall that we want to uncover as many environment anomalies as possible, or otherwise to obtain a reasonable certificate that there are none. A first approach generally considered is a uniformly random agent, but this is ineffective in practice (Sestini et al., 2024). A second natural approach is to script a bot that traverses the entire map. However, this strategy also becomes inefficient in large-scale environments and, more importantly, is unlikely to automatically detect anomalies under our definition unless it exhaustively explores all possible trajectories through the map. Another approach consists in using the training of an RL agent, as proposed by Bergdahl et al. (2020). We present in this paper a third approach, that yields better results when one has access to an optimal expert policy $\pi^\star$. One may leverage this policy through reward shaping, imitation learning, inverse reinforcement learning or an auxiliary expert loss. We argue here why the latter corresponds best to the objective.

We do not know what the environment anomalies are until they are detected. Thus, we cannot attribute a reward for having reached those states. Furthermore, we cannot define a reward based on anomaly definition (Equation 2) as we don't have access to $G^{\pi}(s)$ until the episode ends.

Imitation learning and inverse reinforcement learning both suffer from the same limitation: In anomalous situations, the agent must often deviate significantly from the nominal expert policy to maximize value. Thus, the agent should initially act without expert influence, exploring freely, and only be progressively guided by the nominal policy if it fails to converge or reach the goal.

Gradually incorporating expert information into the training objective satisfies all our needs. The agent initially explores the environment without guidance. If the environment's complexity prevents convergence or goal attainment, expert guidance is introduced progressively, steering the agent toward the goal while preserving early exploratory freedom. Therefore, we introduce expert information progressively into the training objective via a time–dependent increasing coefficient $c(t) \in [0, 1]$. Early in learning, a small $c(t)$ preserves exploratory behavior and reduces over-anchoring to the expert; later, a larger $c(t)$ stabilizes optimization and improves solvability. Let L_t^{base} and L_t^{EXP} denote the base RL objective and the expert-referenced auxiliary loss respectively, the total objective is:

$$L_t(\theta) \;=\; L_t^{\text{base}}(\theta) \;+\; c(t)\, L_t^{\text{EXP}}(\theta), \tag{3}$$

3.3 Expert-Referenced Auxiliary Losses

Let $\pi_\theta(\cdot \mid s_t)$ and $\pi^\star(\cdot \mid s_t)$ denote respectively the learned and expert policies' action distribution, and let $\hat{G}(s_t, a_t)$ denote the return obtained by the learned policy from state s_t and taking action a_t. We define and later investigate a family of expert-referenced auxiliary losses:

$$\textbf{Action-space MSE comparison:} \quad L_t^{\text{MSE}}(\theta, a_t^*) := \frac{1}{2}\left\|\mu_\theta(s_t) - a_t^*\right\|_2^2 \tag{4}$$

$$\textbf{Action-space NLL comparison:} \quad L_t^{\text{NLL}}(\theta, a_t^*) := \frac{(a_t^* - \mu_\theta)^2}{2\sigma_\theta} + \log(\sigma_\theta) + \frac{1}{2}\log(2\pi) \tag{5}$$

$$\textbf{Value supervision:} \quad L_t^{\text{VAL}}(\theta, a_t^*) := \left\| V^{\pi_\theta}(s_t) - G^{\pi^\star}(s_t) \right\|_2^2 \tag{6}$$

$$\textbf{Advantage supervision:} \quad L_t^{\text{ADV}}(\theta, a_t^*) := \left\| \hat{A}^{\pi_\theta}(s_t, a_t^\star) \right\|_2^2 \tag{7}$$

where a_t^* is the expert action at time t, $\mu_\theta(s_t)$ is the agent action gaussian mean at time t, $\hat{A}^{\pi_\theta}(s_t, a_t^\star) = \hat{G}(s_t, a_t^\star) - V^{\pi_\theta}(s_t)$.

In the action-space Mean Square Error (MSE) and Negative Log-Likelihood (NLL) comparisons, the loss encourages the learned policy to match the expert's action distribution at the given state. MSE penalizes Euclidean deviations between probability vectors, whereas NLL penalizes the agent based on the logarithmic likelihood of the expert's action under the learned policy. Their formulations above describe the continuous Gaussian case; for categorical outputs, the framework adopts their discrete equivalents: the Brier Score and Categorical Cross-Entropy, respectively.

In value supervision from expert returns, the value function of the learned policy is trained to regress toward the return obtained by the expert from that state. Thus, the learner is encouraged to predict the outcome achieved by the expert, using it as a supervisory signal.

In advantage supervision, the objective minimizes the squared magnitude of the agent's estimated advantage for the expert action at each state. Specifically, the loss encourages the policy to assign higher value to expert actions relative to the agent's own value baseline, effectively guiding the agent toward behavior that reproduces expert performance.

4 Empirical Evaluation: Navmesh Loss Integration in Goal-Conditioned RL for Automated Bug Findings in Game Worlds

4.1 Goal-Conditioned RL Setting for Automated Bug Finding

A concrete instance of our anomaly definition arises in navigation within virtual worlds. If a goal-conditioned RL agent reaches a target faster than a pathfinding expert, this discrepancy can be an anomaly such as an unintended shortcut through geometry, physics abuse or faulty collision handling (Chambon & Grelier, 2025).

We consider a navigation GCRL agent trained to reach a target g represented by its spatial coordinates. For each environment, the state space $\mathcal{S}$ comprises all virtual world variables, while the agent receives only partial observations $\mathcal{O}$, and the action space $\mathcal{A}$ corresponds to navigation controls.

The reward comprises a sparse reward on reaching the goal, and a time penalty to encourage rapid progress towards the goal.

$$R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R},\ R(s, a, s') = \omega_1 \mathbb{1}_{||x-g|| \leq \varepsilon} - \omega_2 \tag{8}$$

where x is the agent position, ε defines the radius within which the goal is considered reached and $(\omega_1, \omega_2) \in \mathbb{R}^2_+$ are configurable weights.

For both environments, we employ a PPO (Schulman et al., 2017) with entropy regularization. Therefore, the objective with our auxiliary loss is:

$$L_t^{\text{base}}(\theta) = L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \Big[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \Big] \tag{9}$$

where L_t^{CLIP} is the clipped surrogate policy loss, L_t^{VF} is the value regression loss, $\mathcal{S}$ is the policy bonus, c_1, c_2 are static coefficients.

4.2 Our Expert: Navmesh Definition & Properties

Notably, obtaining an expert that satisfies our assumptions does not require significant domain expertise. We further discuss expert availability in section 5. In our setting, we rely on a navmesh (navigation mesh) agent, a standard component available in most modern 3D engines.

A navmesh is a representation of the walkable regions used in a virtual environment in video games or robotics. It consists of a set of 2D convex polygons (typically triangles) that are exploited by agents for pathfinding. Convexity ensures that any two points within a polygon can be connected by a straight-line path, while navigation is performed by constructing a graph of polygons and running a shortest-path algorithm such as A* (Snook, 2000). These surfaces can be generated automatically

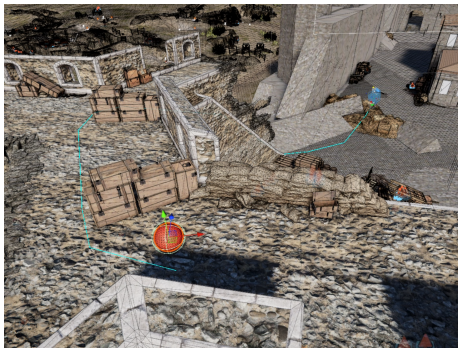


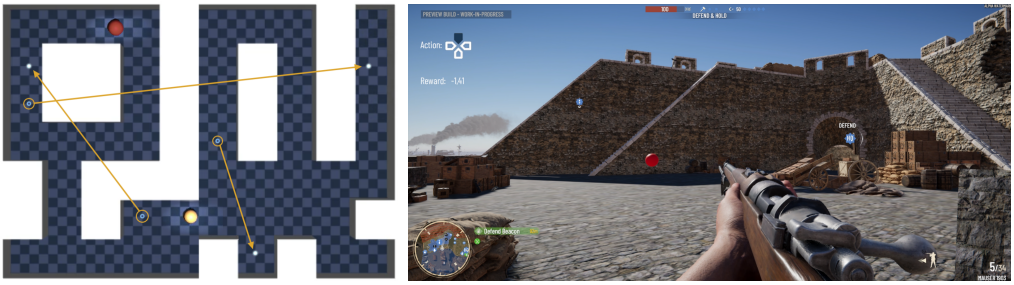
Figure 1: Navmesh path example in cyan from an agent (on the right) to a red sphere (on the left)

from level geometry or manually created level designers. Although navmesh is generally static, it may be locally regenerated at runtime for dynamic pathfinding (van Toll et al., 2012).

Outperforming navmesh guidance with RL in order to approximate player-like behavior has already been studied (Alonso et al., 2021). Moreover, as emphasized by Gordillo et al. (2021), a navmesh agent miss bugs that human players frequently trigger, which motivates its use as "ground-truth" to expose anomaly discovery (Chambon & Grelier, 2025; Bergdahl et al., 2020).

A key property of the navmesh is that its graph encodes a deterministic, low-cost optimal feasible route between any two reachable points. This makes the navmesh a reliable expert policy $\pi^\star$ that we can generate from any state $s \in S$. Therefore, once an episode ends, we retrospectively compute the return, as well as the expert return, for each traversed state. States satisfying the inequality defined in Equation 2 are then automatically flagged as anomalies.

4.3 Environments and Anomaly Definitions



(a) *PointMaze:Teleport* (b) *Gallipoli*

Figure 2: Two environments

We consider two environments, the research-oriented open source environment *PointMaze:Teleport* (Park et al., 2025) which resembles Super Monkey Ball, and the commercial video game *Gallipoli* by Blackmill Games. The latter is employed to exhibit a real use case in a highly challenging setting. For both environments, the reward function employed is given in Equation 8, with $\omega_1 = 1$ and $\omega_2 = 1/H$, so that the highest and lowest episodic return possible are respectively 1 and -1.

In *PointMaze:Teleport*, shown in Figure 2a, a point-mass must reach a randomly located red sphere using a linear horizontal force; from its own position together with that of the goal as observation in $H = 2000$ steps. There are multiple deterministic teleporters, where a black hole immediately transports the agent to an associated white hole. Bugs correspond to specific states that we manually designate as faulty. In this environment, we assume the navmesh expert is unaware of the teleporters, so the learned agent can reach the destination faster by exploiting them. This setup abstracts

any localized spatial discontinuity in virtual worlds (e.g., geometry glitches or unintended warps in games). We reduce the default teleporter size by a factor of 4, reflecting the practical rarity of such glitches, and train for $1,000,000$ steps, a value found sufficient to yield meaningful insights.

In *Gallipoli*, we employ the ML-Agents framework (Juliani et al., 2020) to integrate RL within the game, shown in Figure 2b. The agent moves at a speed of $3\,\mathrm{m/s}$ and must reach a target randomly placed $50\,\mathrm{m}$ away within $H = 2000$ steps. This horizon is chosen to ensure that the target is reachable, and we adopt the same reward function as the previous environment. Similarly, the location of the ball is always generated to be attainable. Observations comprise a semantic 3D map as introduced by Sestini et al. (2024) of resolution $64 \times 3 \times 64$, the vector to the goal, and the angle between the agent's facing direction and this vector. The action space includes moving forward, backward, left, or right, and rotating clockwise or counterclockwise. We deploy the agent in a real game map containing four anomalies, corresponding to objects with missing collision geometry that the navmesh does not account for. Figure 3 illustrates the level, the agent's spawn location, and the objects with missing collisions. While a count of four anomalies might seem low, it corresponds to a real-life scenario of video game development where environment anomalies are not too numerous in limited regions, and thus hard to find. We provide further (non-central) details on environments and implementations in Appendix A.

(a) *Gallipoli* level overview with the spawn location in red, and the target red sphere.

(b) *Gallipoli* objects with collision anomalies colorized.

Figure 3: Experimental setup in the *Gallipoli* level, showing the map layout, agent spawn location, and anomalies.

4.4 Comparison and Ablation Study Protocol

In this section, we conduct a comparison of the proposed loss designs described in subsection 3.3 and ablation study to answer the following research questions: **(Q1)** Which of the losses provides the best trade-off between solvability, state coverage and environment anomalies discovered? **(Q2)** Can progressively applying an expert-guided auxiliary loss uncover environment anomalies more effectively than random, conventional RL, or imitation learning agents? **(Q3)** Can we provide early insights on how sensitive are these results to the choice of the scheduling coefficient $c(t)$?

To address **(Q1)**, we vary the auxiliary loss type: L_t^{NLL}, L_t^{MSE}, L_t^{VAL}, L_t^{ADV}, and use three metrics evaluated at each reporting step: **Success Rate** as the proportion over 100 evaluation episodes that reach the goal, **Anomaly Coverage** is the proportion of predefined bugs flagged as anomalies at least once during training; and **State Coverage** as the proportion of the navigable area visited by the agent. To address **(Q2)**, we compare the most effective loss identified in the previous study against three baselines in an ablation setup: a random policy (which serves as a standard baseline strategy for maximizing exploratory state coverage), an RL policy without the auxiliary loss (ablation study), and an imitation learning agent trained via behavior cloning of the expert (to evaluate the risk of over-anchoring on the expert's support).

To address **(Q3)**, we investigate three common types of scheduled coefficient $c(t)$: a **Linear warm-up:** $c(t) = \min\left(1, \frac{t}{T_{\text{ramp}}}\right)$; a **Cosine warm-up:** $c(t) = \left[0.5\left(1 - \cos(\pi \min(1, t/T_{\text{ramp}}))\right)\right]^{\kappa}$; and an **Exponential warm-up:** $c(t) = \frac{1-\exp(-k \min(1, t/T_{\text{ramp}}))}{1-\exp(-k)}$.

As we are looking for early insights, we arbitrarily set the cosine warm-up curvature parameter κ to 1; and the exponential warm-up steepness parameter to 5. Owing to the substantial computational cost associated with the *Gallipoli* environment, the ablation study and the warm-up coefficient analysis are conducted exclusively in the *PointMaze:Teleport* environment. For each environment, we report means and standard deviation over 10 random seeds were all hyperparameters are kept constant except the factor under test.

4.5 Results and Discussion

In Figure 4, we report the metrics described above for the *PointMaze:Teleport* environment, comparing the performance of the different auxiliary losses under evaluation. Each curve is identified in the legend by the symbol of its corresponding auxiliary loss and employs a linear warm-up. Across all metrics, the MSE loss consistently outperforms the other proposed losses. Notably, it is the only loss that achieves a success rate of 1, successfully identifying all anomalies in every training run, and a full coverage of the state space. This likely results from the MSE's linear gradient providing greater exploratory opportunities, whereas the NLL's logarithmic penalty generates an aggressive gradient that induces over-anchoring to the expert's support. Consistent with the observations of He et al. (2022), we also find that some auxiliary losses can be detrimental, leading to low success rates.

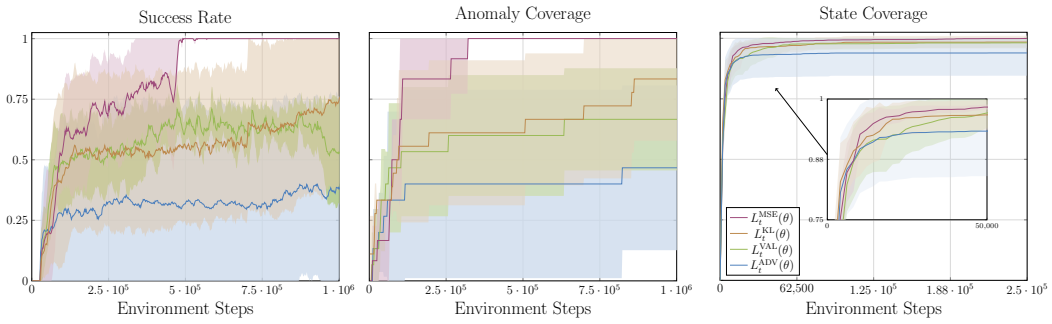


Figure 4: Comparison study on the *PointMaze:Teleport* environment

In Figure 5, we plot the same auxiliary loss-function comparison in the more challenging *Gallipoli* environment and include a random agent as a baseline to contextualize performance against purely stochastic behavior; resulting in zero anomaly discoveries. The results reinforce the trends observed previously. In particular, the MSE loss achieves the highest success rates, identifies all anomalies within the training budget, and yields the broadest state-space coverage (90%), aside from Random. The relative ranking of the loss functions remains largely consistent with the earlier experiment. The NLL loss exhibits performance noticeably closer to that of the MSE auxiliary loss in this environment than in the previous one. Based on these performance evaluations, we select the MSE loss for the ablation study presented in Figure 6.

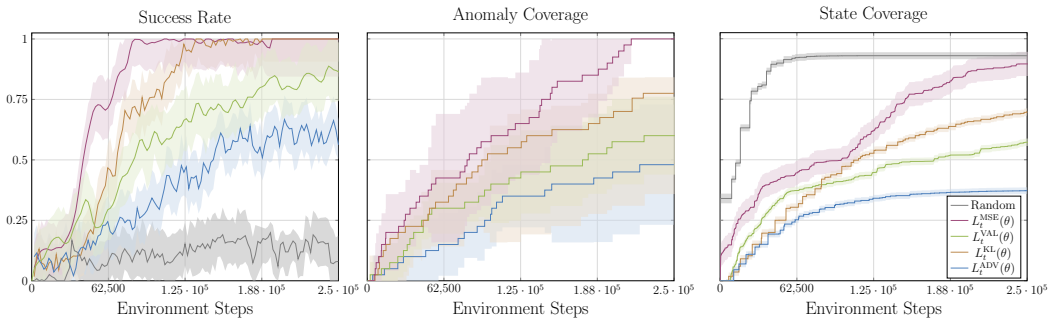


Figure 5: Loss comparison study on the *Gallipoli* environment

The ablation study shown in Figure 6 demonstrates that our proposed addition of an auxiliary loss, particularly the MSE loss, substantially improves anomaly discovery efficiency compared to the other methods. Interestingly, an unmodified RL agent can eventually achieve full anomaly coverage in this relatively simple environment, although it does so with a low success rate. In contrast, behavior cloning attains a reasonable success rate but, as noted by Sestini et al. (2024), struggles to deviate from the expert policy and therefore fails to effectively explore states that may reveal environment anomalies, achieving only 75% anomaly coverage. Finally, the random policy achieves the highest state coverage, which is expected given its exploratory nature.

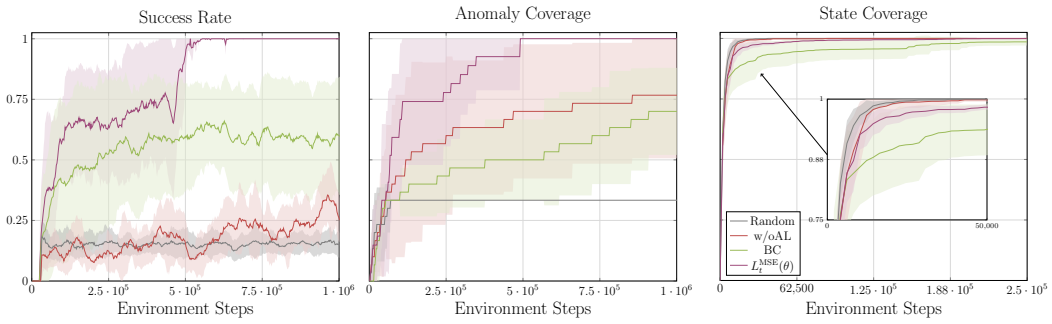


Figure 6: Ablation study on the *PointMaze:Teleport* environment

Ultimately, we plot in Figure 7 each metric for the investigated warm-up schedules in the *PointMaze:Teleport* environment using the MSE loss, as it was identified as the highest-performing loss in our previous investigations. Early insights indicate that the linear warm-up achieves the best performance, followed by the exponential and cosine schedules. This suggests that rapidly introducing the auxiliary loss improves both success rate and anomaly coverage by encouraging a stronger behavioral shift instead of simultaneously optimizing multiple objectives. However, these findings are limited by the evaluation of only selected cosine and exponential hyperparameter configurations; broader experiments and more extensive hyperparameter sweeps should be studied in future work.

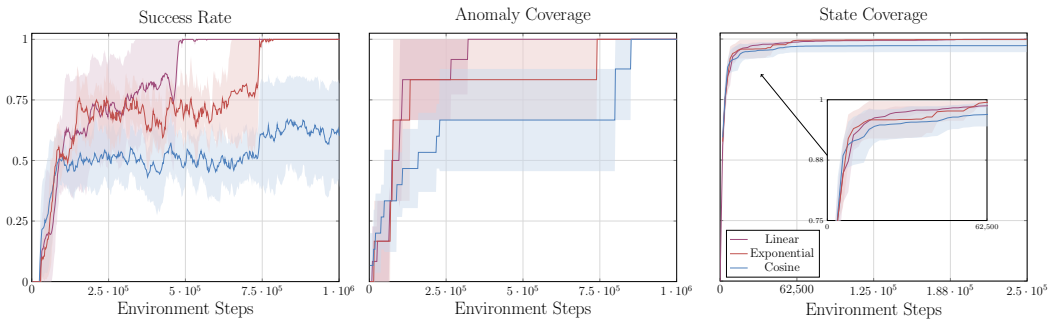


Figure 7: Coefficient comparison study on the *PointMaze:Teleport* environment

5 Limitations and Future Work

Stochastic Environments and Suboptimal Experts. Our formulation of an environment anomaly in Equation 2 assumes the nominal expert π^* is optimal under anomaly-free dynamics in a deterministic environment. The condition $G^{\pi}_{real} > G^{\pi^*}_{real}$ could trigger false positives in non-buggy regions simply because either if the reference policy is inherently flawed therefore the RL agent discovers a more efficient path than a weak baseline, or in highly stochastic domains where an agent might outperform the nominal expert due to favorable random transitions rather than genuine structural flaws.

Regarding the latter, in virtual world debugging and automated video game QA, established standard validation practices explicitly dictate locking pseudo-random number generator seeds to ensure exact reproducibility. Therefore assuming a deterministic environment aligns with this paradigm. Notably, the expert being perfect by assumption and in practice; and all presented environments being deterministic, no false positive are to be reported in our experiments.

If a stochastic domain is necessary or if the expert is necessarily suboptimal, we do believe the trained agent remains an efficient anomaly discovery tool by addressing this limitation through two main strategies:

First, one may introduce a safety margin threshold ϵ to Equation 2, modifying the activation condition to $G^{\pi}_{real} > G^{\pi^*}_{real} + \epsilon$ to absorb environmental variance or baseline inefficiencies.

Second, flagged anomalies can subsequently be verified via a secondary validation layer to filter out false positives, such as localized sensitivity analysis, statistical hypothesis testing over multiple rollouts, or targeted manual replays.

Boundaries of the Anomaly Definition. The formalization of environment anomalies introduced in this paper strictly targets functional, mechanic-driven discrepancies that alter the environment's defined reward and transition dynamics. We do not claim this definition encompasses all software bugs. Visual artifacts, broken user interfaces, or missing audio triggers, do not alter the physical state-action space and cannot be uncovered via this formulation. Instead, this methodology targets interaction-based anomalies requiring complex behavioral sequences to exploit.

This category encompasses both the bugs evaluated in this work (missing geometry collisions, unintended physics engine glitches) and other representative examples, such as structural geometry abuse (e.g. velocity accumulation), missing environmental penalties (e.g. disabled fall damage hazards), and generally behavioral exploits (reward or score hacking).

Future Directions. Future work will focus on expanding both the theoretical and empirical scope of this framework. Theoretically, we aim to develop a formal analysis detailing how the scheduling of the auxiliary loss influences exploration bounds and the exact probability of encountering anomalies during training.

Empirically, we plan to evaluate the inclusion of curiosity based modules such as Random Network Distillation in CCPT (Sestini et al., 2024) and compare our method to curiosity-driven approaches, and expand our benchmark through extensive hyperparameter fine-tuning for cosine and exponential schedules, while evaluating the framework against different environment anomalies. Specifically, we intend to validate the framework against *Craftium:Speleo* (Malagón et al., 2025) missing fall damage. In this setup, the agent would be rewarded for descent velocity while minimizing damage taken, and the nominal expert will be a navigation agent unaware of the missing fall damage mechanic.

6 Conclusion

We presented a method leveraging an expert policy to enhance anomaly detection via scheduled auxiliary losses, improving discovery while preserving exploration. Evaluating four variants across research and commercial environments, the mean squared error loss consistently achieved superior success rates and discovery efficiency. Crucially, our results decouple discovery from mere coverage or solvability: In *Gallipoli*, while random policies maximized coverage without finding bugs, behavior cloning's over-anchoring restricted discovery despite task success. By bridging these objectives, our framework aims for agents to solve the task while retaining the flexibility to exploit and flag environmental anomalies, outperforming variants and baseline methods on considered metrics. While this methodology is currently tailored to deterministic, interaction-based anomalies, it remains extensible to stochastic domains and suboptimal references via adaptive safety thresholds or secondary validation layers. Future work will expand our empirical validation to a broader range of anomalies, parameters, and environments.

A Environment and implementation details

In this appendix, we provide additional details on PPO implementations, environments and training parameters. All experiments were performed on an AMD Ryzen Threadripper PRO 7965WX 24-Core CPU and a NVIDIA RTX 6000 Ada Generation GPU.

A.1 *PointMaze:Teleport* experiment

Figure 8: Additional *PointMaze:Teleport* environment screeenshots

In our setup, the agent operates with a decision frequency of 3e-1, repeating actions in between decisions. We employ a PPO agent using stable-baselines3 (Raffin et al., 2021) with the default `MlpPolicy`, since this environment returns a dictionary as the observation space. This architecture flattens the observation data before using MLP extractors and finally separates actor and value heads for policy and state-value predictions, as shown in Figure 9.

In the original layout, teleporters are stochastic whereas in our setting they are made deterministic. Each black hole is associated with and leads to a specific white hole. Furthermore, in the original layout, one of the three white "teleporters" terminates in a dead end, so taking any teleporter carries inherent risk. As illustrated in the screenshots, we modified the maze geometry to remove the dead-end section of the level and added a black hole to create a one-to-one black/white holes relation.

We also modify the reward signal of 1 when reaching the goal to the reward specified in Equation 8 instead , thereby adding a time penalty. Specifically in the implementation, we add the goal location to the observation space, force termination on reaching the goal, and set the action scaling coefficient to 0.5. Training is performed using the Adam optimizer. Eventually, we provide additional hyperparameters in Table 1. Most values have been kept as default to focus on the auxiliary loss integration.

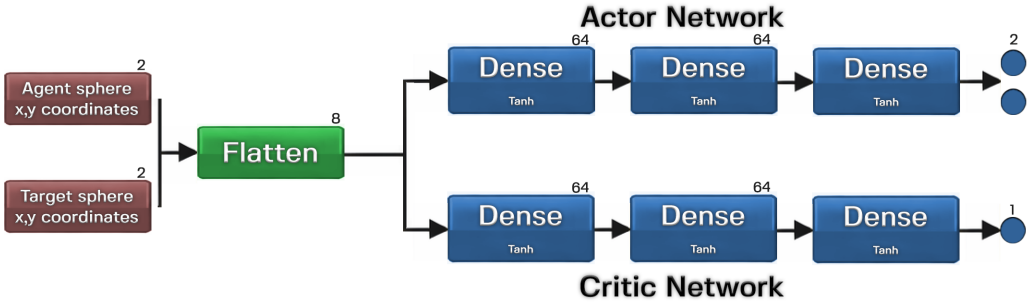


Figure 9: Diagram of the MlpPolicy used in the *PointMaze:Teleport* environment

A.2 *Gallipoli* environment

Gallipoli is a first-person shooter video game set on the Middle Eastern front of World War I, developed by BlackMill Games. The implementation of RL within this commercial game as been done

using ML-Agents (Juliani et al., 2020). The agent used in this environment is trained with PPO (Schulman et al., 2017), with a decision frequency of 3e-2, a value empirically determined to be appropriate given the environment's dynamics. The PPO actor network outputs two linear heads corresponding to the agent's two discrete action branches, as shown in Figure 10. The first head is a categorical distribution over movement actions and the second head is a categorical distribution over rotational actions. These two pieces of information are simultaneously transmitted and applied in *Gallipoli*, allowing for a behavior close to that of a real player.

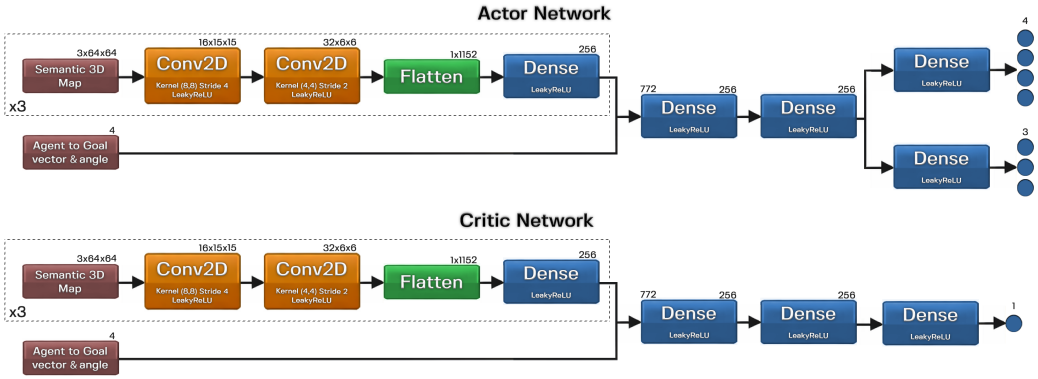


Figure 10: Diagram of the actor and critic architectures used in the *Gallipoli* environment

Training is performed using backpropagation with the Adam optimizer. The 3D semantic map uses voxels of size $0.1\text{m} \times 0.5\text{m} \times 0.1\text{m}$, chosen to match the spatial scale of the game and to ensure that the discretization captures navigation-relevant structure in the scene. Eventually, we provide additional hyperparameters in Table 1. Similarly, most values have been kept as default to focus on auxiliary loss integration. Note that in this context, as mentioned in subsection 3.3, the MSE loss becomes the Brier Score computed using the probabilities output of the softmax layer, while the NLL loss becomes $L_t^{\text{NLL}}(\theta, a_t^*) \quad := -\log \pi_\theta(a_t^* | s_t)$, which is mathematically equivalent to a cross-entropy objective given a deterministic expert.

Table 1: Parameters used for evaluation, difference between *PointMaze:Teleport* and *Gallipoli* are seperated by a vertical bar

PARAMETER	**VALUE**
Batch	128
Buffer size	2048
Learning rate	$3\text{e}{-4}$
Entropy loss coefficient	$1\text{e}{-2}$
Critic loss coefficient	0.5
Epsilon clip	0.2
GAE Lambda	0.95
Discount factor	0.99
Epochs number	3
Horizon H	1000 | 2000 steps
Gallipoli 3D semantic map resolution	$6.4\text{m} \times 1.5\text{m} \times 6.4\text{m}$
Gallipoli agent speed	3 m/s
Total timesteps for training	1e6 | 2.5e5 steps
Success trigger distance ε	1 unit | 1 m

References

Eloi Alonso, Maxim Peter, David Goumard, and Joshua Romoff. Deep reinforcement learning for navigation in aaa video games. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2133–2139. International Joint Conferences on Artificial Intelligence Organization, 8 2021. DOI: 10.24963/ijcai.2021/294. URL https://doi.org/10.24963/ijcai.2021/294. Main Track.

Kinza Arshad, Rao Faizan Ali, Amgad Muneer, Izzatdin Abdul Aziz, Sheraz Naseer, Nabeel Sabir Khan, and Shakirah Mohd Taib. Deep reinforcement learning for anomaly detection: A systematic review. *IEEE Access*, 10:124017–124035, 2022. DOI: 10.1109/ACCESS.2022.3224023.

Joakim Bergdahl, Camilo Gordillo, Konrad Tollmar, and Linus Gisslén. Augmenting automated game testing with deep reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, pp. 600–603, 2020. DOI: 10.1109/CoG47356.2020.9231552.

Emilien Biré, Anthony Kobanda, Ludovic Denoyer, and Rémy Portelas. Efficient active imitation learning with random network distillation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=GFgn2LprFR.

Aurélien Chambon and Nicolas Grelier. Comparing high entropy reinforcement learning to navmesh for automated collision bug testing. In *Reinforcement Learning and Video Games Workshop @ RLC 2025*, 2025. URL https://openreview.net/forum?id=PIWiufVV5a.

Camilo Gordillo, Joakim Bergdahl, Konrad Tollmar, and Linus Gisslén. Improving playtesting coverage via curiosity driven reinforcement learning agents. In *2021 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE Press, 2021. DOI: 10.1109/CoG52621.2021.9619048. URL https://doi.org/10.1109/CoG52621.2021.9619048.

Tairan He, Yuge Zhang, Kan Ren, Minghuan Liu, Che Wang, Weinan Zhang, Yuqing Yang, and Dongsheng Li. Reinforcement learning with automated auxiliary loss search. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 1820–1834. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/0be44cc1d459731928501cae5699f57a-Paper-Conference.pdf.

Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2020. URL https://arxiv.org/pdf/1809.02627.pdf.

Mikel Malagón, Josu Ceberio, and Jose A. Lozano. Craftium: Bridging flexibility and efficiency for rich 3d single- and multi-agent environments. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=htP5YRXcS9.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Alessandro Sestini, Linus Gisslén, Joakim Bergdahl, Konrad Tollmar, and Andrew David Bagdanov. Automated gameplay testing and validation with curiosity-conditioned proximal trajectories. *IEEE Transactions on Games*, 16(1):113–126, 2024. DOI: 10.1109/TG.2022.3226910.

Alessandro Sestini, Joakim Bergdahl, Konrad Tollmar, Andrew D. Bagdanov, and Linus Gisslén. TROFI: Trajectory-ranked offline inverse reinforcement learning. In *Reinforcement Learning and Video Games Workshop @ RLC 2025*, 2025. URL https://openreview.net/forum?id=5Lvr5xFkfX.

Greg Snook. Simplified 3d movement and pathfinding using navigation meshes. In Mark DeLoura (ed.), *Game Programming Gems*, pp. 288–304. Charles River Media, 2000.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

Wouter van Toll, Atlas Iv, and Roland Geraerts. A navigation mesh for dynamic environments. *Computer Animation and Virtual Worlds*, 23:535–546, 11 2012. DOI: 10.1002/cav.1468.

Kun Zhou, Wenyong Wang, Teng Hu, and Kai Deng. Application of improved asynchronous advantage actor critic reinforcement learning model on anomaly detection. *Entropy*, 23(3), 2021. ISSN 1099-4300. DOI: 10.3390/e23030274. URL https://www.mdpi.com/1099-4300/23/3/274.

Álvaro Serra-Gomez, Daniel Jarne Ornia, Dhruva Tirumala, and Thomas Moerland. A kl-regularization framework for learning to plan with adaptive priors, 2025. URL https://arxiv.org/abs/2510.04280.