WMATTACK: AUTOMATED ATTACK SEARCH FOR ADVERSARIAL EVALUATION OF WORLD-MODEL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Assessing adversarial robustness in world-model agents requires costly, noisy closed-loop rollouts. We cast this task as a search for attack configurations against a fixed victim, subject to observation perturbation limits and a finite online budget. We present **WMAttack**, a closed-loop framework with two modules: Representation-Guided Attack Retrieval (RGAR) and Self-Correcting Attack Search (SCAS). RGAR matches internal and behavioral descriptors to stored attack records, retrieving configurations that give search an informed starting point. SCAS uses feedback from the current task to guide the construction, ranking, and selection of feasible candidates. An Auditor summarizes return degradation, policy-output disagreement, runtime, and return variability into structured advice. Fixed rules use this advice to build and rank candidates, and an LLM proposer selects from the resulting shortlist. A scout–confirm schedule screens candidates, confirms promising choices, and freezes the selected configuration for independent testing. Test results never feed back into search. On 26 Atari and 20 DeepMind Control tasks with DreamerV3, full WMAttack exceeds the highest-mean baseline, Successive Halving, by 0.138 and 0.133 in mean independent-test normalized return degradation, respectively, under matched online episode budgets; only WMAttack uses historical records. Controlled ablations support both modules' value, while further evaluations demonstrate applicability to three other world-model architectures.

1 INTRODUCTION

World models learn latent dynamics for prediction, imagination-based planning, and decision-making (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2023), enabling visual control (Tassa et al., 2018; Bellemare et al., 2013; Hansen et al., 2024), interactive simulation, and driving-scenario generation (Yang et al., 2024; Hu et al., 2023). Strong task performance, however, does not imply adversarial robustness. In closed-loop control (Micheli et al., 2023; Zhang et al., 2024), small observation perturbations can alter latent states, actions, and subsequent observations, cumulatively degrading long-horizon returns (Zhang et al., 2020). Prescribed configurations may miss vulnerabilities across tasks and architectures and overestimate robustness, as in static classifiers (Croce & Hein, 2020a; Croce et al., 2021). Automated attack discovery also extends to language models (Panfilov et al., 2026). We study *attack-configuration search* for world-model agents: discovering and confirming high-impact configurations against a fixed victim under explicit observation constraints and a finite rollout budget, then assessing performance under the selected configurations.

Three challenges motivate this search. **First, rollouts are expensive.** Complete episodes require repeated encoder, dynamics, and policy inference. **Second, configurations are heterogeneous.** Attack families and hyperparameters vary with tasks, architectures, and dynamics, creating cold-start and negative-transfer risks. **Third, feedback is sparse and noisy.** Delayed, high-variance returns may remain unchanged despite altered behavior (Vafa et al., 2024; Balogh & Jelasity, 2026; Upadhyay et al., 2026). We observe policy-output disagreement at over half the evaluated decisions on some tasks with nearly unchanged returns, motivating complementary diagnostics of policy behavior.

WMAttack combines a feedback-guided search controller with optional **Representation-Guided Attack Retrieval (RGAR)** of historical configurations using composite internal and behavioral descriptors. Fixed rules construct, constrain, deduplicate, and rank configurations. In LLM-backed **Self-Correcting Attack Search (SCAS)**, an Auditor converts evaluation records into structured

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.

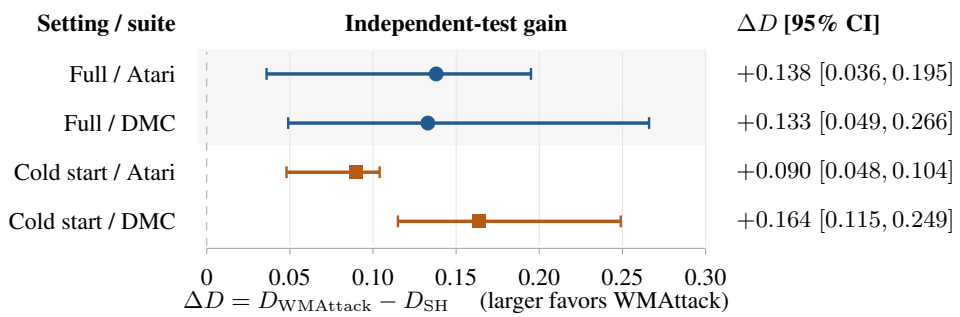


Figure 1: Controlled search gains on 26 Atari and 20 DMC tasks. Points and bars show paired mean differences and individual 95% bootstrap confidence intervals from Table 2, relative to Successive Halving (SH), the highest-mean baseline in each panel. All methods use 32 configurations per family, a 10-pixel radius, and 18 online attack episodes per task–family–seed unit, followed by independent testing. Full WMAttack alone uses history; cold start disables history for all methods. ΔD denotes the difference in normalized return degradation.

recommendations, and a proposer selects shortlist identifiers. Scout evaluations screen candidates; confirmation evaluations guide selection, after which the selected configuration is frozen for independent testing. Controlled comparisons across 46 DreamerV3 tasks support a mean independent-test advantage under matched online episode budgets, including without historical access, as shown in Figure 1. Separate experiments assess historical retrieval and LLM increments over shared programmatic rules; historical runs and additional victims—DreamerV2, TD-MPC2, and IRIS (Hafner et al., 2021; Hansen et al., 2024; Micheli et al., 2023)—characterize applicability and discovered vulnerabilities.

Our contributions are threefold:

- We formulate adversarial evaluation of fixed world-model agents as attack-configuration search under explicit observation constraints and finite online rollout budgets, separating selection from independent performance estimation.
- We introduce a closed-loop search framework in which RGAR retrieves historical configurations using internal and behavioral descriptors, while SCAS converts evaluation feedback into structured recommendations that guide rule-based candidate construction and LLM-based selection.
- Across 46 DreamerV3 tasks, we demonstrate independent-test search gains under matched online episode budgets, including cold start, and use controlled ablations to assess the additional value of representation-guided retrieval and LLM components.

2 Related Work

2.1 Automated Adversarial Evaluation

Automated adversarial evaluation combines ensembles, configuration search, and algorithm generation: AutoAttack combines complementary attacks (Croce & Hein, 2020a); adaptive attack discovery and AutoAE search for effective strategies or ensembles (Yao et al., 2021; Liu et al., 2023); Claudini generates LLM attacks (Panfilov et al., 2026). World-model evaluation additionally requires closed-loop interaction, where actions change subsequent observations and internal states. WMAttack manages the resulting rollout cost and return noise through budgeted proposal, screening, and confirmation under explicit observation constraints.

2.2 Adversarial Attacks on RL and World Models

Early RL attacks and SA-MDP study observation perturbations (Huang et al., 2017; Lin et al., 2017; Zhang et al., 2020); ATLA and PA-AD learn attacks targeting long-term returns (Zhang et al., 2021;

Sun et al., 2022). PhysCond-WMA and CtrlAttack target conditional control in generative world models (Guo et al., 2026; Xu et al., 2026), while TRAP uses deployment-time visual patches to disrupt imagined-trajectory evaluation in fixed models (Duan et al., 2026). WMAttack focuses on selecting and evaluating configurations within prescribed attack families, allocating resources across candidates against a fixed closed-loop agent under explicit threat and budget constraints.

2.3 Budgeted Search and Optimization

Random search, Bayesian optimization, and successive halving support exploration and resource allocation (Bergstra & Bengio, 2012; Srinivas et al., 2010; Jamieson & Talwalkar, 2016); multi-fidelity falsification searches for control-system failures (Shahrooei et al., 2024); OPRO generates subsequent solutions from previous solutions and their scores, while Reflexion uses verbal feedback to guide subsequent attempts (Yang et al., 2023; Shinn et al., 2023). WMAttack applies these principles to closed-loop observation attacks, coupling retrieval from internal and behavioral descriptors with structured feedback, constrained candidate construction, and candidate selection. Scout and confirmation stages use different episode counts, and the selected configuration is frozen before independent testing.

3 Preliminaries

3.1 Victim and Threat Model

We evaluate a trained world-model agent in closed-loop interaction with an environment:

$$M_\theta = (\phi_\theta, f_\theta, \pi_\theta), \tag{1}$$

where ϕ_θ encodes observations, f_θ models latent dynamics, and π_θ is a policy or planning-based controller. This abstraction covers recurrent (Hafner et al., 2021; 2023), Transformer-based (Micheli et al., 2023), and planning-oriented agents (Hansen et al., 2024). During evaluation, θ remains fixed while observations drive internal states, actions, and trajectories. Attacks modify only observations, leaving model parameters, environment dynamics, and rewards unchanged. Altered latent-state updates and actions can affect subsequent observations and long-horizon returns, as shown in Appendix A.

3.2 Adversary Capabilities

For task τ, an admissible configuration $c \in \mathcal{C}_\tau$ specifies the attack family, perturbation magnitude, optimization steps, restarts, and family-specific controls; seeds are recorded for reproducibility, not optimized. We instantiate APGD-CE, APGD-DLR (Croce & Hein, 2020a), FAB (Croce & Hein, 2020b), a model-guided Square variant (Andriushchenko et al., 2020), and a PhysCond-WMA-inspired two-stage variant (Guo et al., 2026), with configuration ranges as shown in Appendix B.

The attacks access victim representations, policy outputs, and gradients as required, including gradient-based guidance for model-guided Square. Neither environment-model gradients nor future outcomes are assumed. DreamerV3 attacks perturb preprocessed encoder image keys $k \in \mathcal{K}_{\mathrm{img}}$, including all slices of the current frame stack when enabled:

$$\tilde{o}_{t,k} = \Pi_{[-0.5,0.5]}(o_{t,k} + \delta_{t,k}), \qquad \|\delta_{t,k}\|_\infty \le \epsilon/255. \tag{2}$$

Under DreamerV3's $x/255 - 0.5$ normalization, ϵ is in pixel units; non-image inputs remain unchanged. Elementwise projection enforces the converted budget before valid-range clipping, as shown in Appendix B; coordinates and clipping are adapter-specific, as shown in Appendix A. Configurations obey the declared threat model; controlled admissible spaces and information access are specified separately from historical conditions as shown in Section 5.1.

3.3 Attack Goal

For episodic return $R = \sum_{t=0}^{H-1} r_t$ over horizon H, expected clean and attacked returns are

$$J_{\mathrm{clean}}(\tau) = \mathbb{E}[R_{\mathrm{clean}}], \qquad J_{\mathrm{adv}}(\tau, c) = \mathbb{E}[R_{\mathrm{adv}}(c)]. \tag{3}$$

Their sample means are reported as *clean return* and *attacked return*; lower attacked return indicates greater task-performance degradation.

Search favors higher $U_\tau(c)$, which combines normalized return degradation and policy-output disagreement with runtime and variability penalties, as shown in Appendix C; this criterion differs from reported return. For N candidate evaluations with n_i episodes in evaluation i and confirmed set $\mathcal{C}_B^{\text{conf}}$, the online rollout budget and selection rule are

$$\sum_{i=1}^{N} n_i \le B, \qquad \hat{c}_B \in \arg\max_{c \in \mathcal{C}_B^{\text{conf}}} \widehat{U}_{\tau,\text{conf}}(c). \tag{4}$$

Scout and confirm episodes count toward B; clean-reference, independent-test, and offline diagnostic episodes are separate. The protocol specifies allocations and horizons; runtime is tracked separately because configuration costs differ. Since confirmation selects the configuration, independent performance estimation uses the fixed, Utility-selected configuration on fresh episodes.

4 METHOD

4.1 BUDGETED ATTACK EVALUATION

For each task and attack family, WMAttack implements the objective in Section 3.3 through a proposal–screening–confirmation loop, as shown in Figure 2. Each evaluation record measures one configuration at a specified scout or confirmation stage over complete episodes with a fixed victim.

Initialize and propose. Clean rollouts provide the return reference and available behavioral summary. SCAS constructs and ranks candidates from available memory and current-task feedback, starting from optional RGAR history or the generic candidate pool.

Screen and confirm. Scout evaluation screens each batch before the highest-Utility candidates receive confirmation with more episodes. A predefined schedule sets batch sizes, promotion rules, stage budgets, and a confirmation limit; search stops when this limit is reached or no new candidates are proposed. Both stages count toward the online budget; clean-reference, test, and diagnostic costs and runtime are reported separately, as shown in Appendix B.

Update and report. The search history records outcomes, stages, and episode counts to guide subsequent proposals. We select the configuration with the highest recorded confirmation Utility and report its attacked return. Independent post-search tests evaluate this fixed configuration with disjoint seeds and separately accounted costs.

4.2 REPRESENTATION-GUIDED ATTACK RETRIEVAL

RGAR combines latent-state information used for dynamics prediction with policy responses and rollout statistics, using descriptor similarity and historical Utility to retrieve configurations that initialize and guide search. For task τ, it forms a query descriptor from a clean baseline rollout or cached evaluation; retrieval refreshes during search apply the same aggregation to the current attacked-rollout probe:

$$\psi_\tau = g\left(\{v_t\}_{t\in\mathcal{I}}, s_\tau\right), \tag{5}$$

where $\mathcal{I}$ indexes sampled decision points, v_t contains available encoder, recurrent, latent, and actor/policy features, and s_τ contains rollout statistics. The aggregation g combines feature means, standard deviations, mean absolute values, and root-mean-square magnitudes with return and policy-response statistics.

For each attack family, RGAR zero-pads current and historical descriptors to equal length and, when PCA is enabled, jointly centers and projects them. This maps current and historical descriptors into a common numerical space. Writing the projected descriptors as $\bar{\psi}$, the historical-record score and retrieval rule are

$$s_i = 6\big(\cos(\bar{\psi}_\tau, \bar{\psi}_i) + 1\big) + 0.5\tanh(U_i), \qquad \mathcal{N}_K(\tau) = \text{TopK}_i\ s_i, \tag{6}$$

where U_i is the historical record's Utility. Cosine similarity measures descriptor proximity, while the bounded Utility term favors configurations with higher historical Utility. RGAR retrieves the

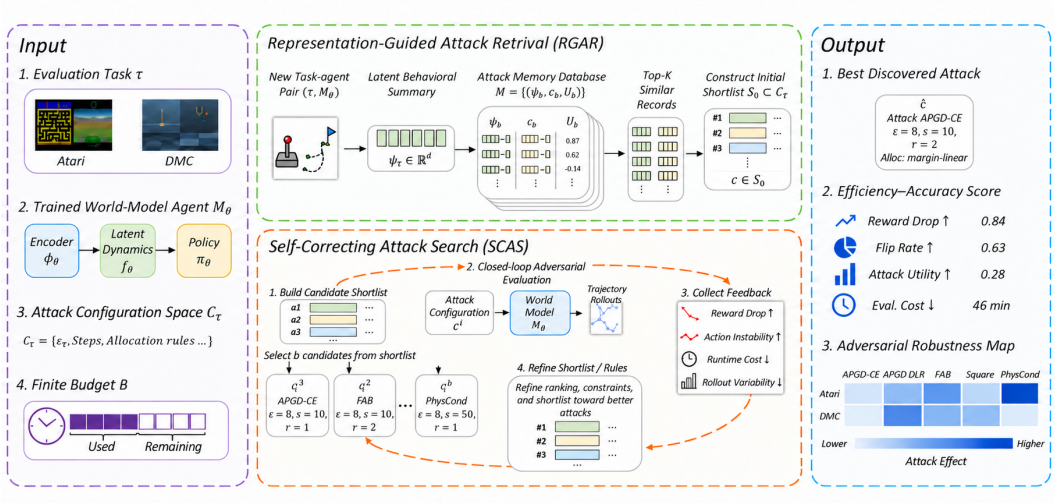


Figure 2: Overview of WMAttack. RGAR provides optional retrieval-guided warm starts, SCAS converts rollout feedback into constrained candidate proposals, and scout–confirm evaluation allocates episodes for screening and confirmation.

highest-ranked positive-score records to anchor initial candidates and local variants and influence candidate ranking. Retrieval and LLM-context limits are controlled separately. If usable latent descriptors are absent, structured task descriptors provide a fallback; without compatible history, search proceeds without retrieval.

In the controlled retrieval comparison, the added internal feature groups and separately fitted projection jointly determine changes in retrieval. The comparison evaluates these features and their projection as a composite retrieval representation. Appendix G.2 specifies history eligibility and the evaluation protocol.

4.3 Self-Correcting Attack Search

SCAS adapts configurations to the current task using performance and behavioral feedback that complements noisy scalar Utility estimates, guiding search with or without retrieved experience.

Each scout–confirm round records Utility and four measurements: return degradation for task-performance loss, policy-output disagreement (Flip) as a complementary behavioral diagnostic, runtime for evaluation cost, and episode-return variability for dispersion, as shown in Appendix C. The evaluation history is updated as

$$\mathcal{H}_t = \mathcal{H}_{t-1} \cup \left\{ \left(c_t^{(j)}, \widehat{U}_t^{(j)}, \mathbf{m}_t^{(j)}, n_t^{(j)}, \ell_t^{(j)} \right) \right\}_{j=1}^{b_t}, \tag{7}$$

where b_t counts new evaluation records, $c_t^{(j)}$ denotes the configuration, $\widehat{U}_t^{(j)}$ its estimated Utility, and $\mathbf{m}_t^{(j)}$ the measurements above. Episode count $n_t^{(j)}$ and scout/confirm stage $\ell_t^{(j)}$ distinguish evaluation fidelities; evaluations of the same configuration at different stages remain separate records.

SCAS separates feedback interpretation, candidate construction, and candidate selection. Fixed rules generate variants from the current best configuration, retrieved configurations, and recent recommendations; enforce parameter constraints; remove duplicate or previously proposed configurations; and rank the remainder with the heuristic score in Appendix B.1. Shared-menu experiments also discard candidates outside the prescribed menu. This score governs shortlisting, whereas evaluation Utility governs promotion and final selection.

With an LLM backend, the Auditor converts evaluation records into adjustment directions, target values, and allocation preferences. These recommendations guide rule-based candidate construction and ranking, after which the proposer selects shortlisted configuration identifiers for evaluation.

Scout–confirm outcomes then update the history for subsequent rounds. Appendix B.1 specifies the general interface's fallback rules and the strict controlled interface's failure handling.

4.4 Analysis of Budgeted Attack Search

Finite-budget evaluation faces both incomplete configuration coverage and noisy selection. Let $S_B \subseteq \mathcal{C}_\tau$ be the nonempty finite set of confirmed configurations, and let $U_\tau^\star = \sup_{c\in\mathcal{C}_\tau} U_\tau(c) < \infty$ under fixed confirmation fidelity and evaluation conditions. Conditional on $\max_{c\in S_B} |\widehat{U}(c) - U_\tau(c)| \le e$, any returned configuration $\hat{c}_B \in S_B$ with $\widehat{U}(\hat{c}_B) \ge \max_{c\in S_B} \widehat{U}(c) - \eta_{\text{sel}}$ satisfies

$$U_\tau^\star - U_\tau(\hat{c}_B) \le \underbrace{U_\tau^\star - \max_{c\in S_B} U_\tau(c)}_{\text{configuration coverage gap}} + \eta_{\text{sel}} + 2e, \qquad e, \eta_{\text{sel}} \ge 0. \tag{8}$$

The bound separates coverage, selection, and estimation errors; exact maximization of recorded confirmation Utility gives $\eta_{\text{sel}} = 0$. It applies to composite Utility under the stated uniform-error condition. Appendix D.4 provides the proof, while Appendix D analyzes idealized proposal distributions.

5 Experiments

5.1 Experimental Setup

Datasets and victim models. We evaluate WMAttack on Atari (Bellemare et al., 2013) and the DeepMind Control Suite (DMC) (Tassa et al., 2018). Our DreamerV3 (Hafner et al., 2023) evaluation covers 26 Atari tasks and 20 DMC tasks. To examine cross-architecture applicability, we further evaluate DreamerV2 (Hafner et al., 2021), TD-MPC2 (Hansen et al., 2024), and IRIS (Micheli et al., 2023) on task subsets. All agents are trained before evaluation, and their model parameters remain fixed during adversarial testing.

Attack search space. WMAttack searches configurations for APGD-CE, APGD-DLR (Croce & Hein, 2020a), FAB (Croce & Hein, 2020b), and a PhysCond-WMA-inspired two-stage attack (Guo et al., 2026); historical DMC experiments also include model-guided Square Attack (Andriushchenko et al., 2020). Controlled comparisons use a shared 32-configuration menu within each family, combining eight optimization-step settings with four allocation settings, to compare search policies under a common candidate space and online episode budget. Attack definitions, search ranges, and gradient paths appear in Appendix B; Appendix J.3 maps experiments to their search-space and budget protocols.

Baselines and ablations. WMAttack uses Qwen2.5-3B-Instruct for all enabled proposer/Auditor modules, except in the backbone ablation in Appendix K. Historical comparisons use heuristic Random and Claudini-style baselines. Table 2 compares Coarse Grid, Random Search (Bergstra & Bengio, 2012), GP-UCB-style (Srinivas et al., 2010), Successive Halving (Jamieson & Talwalkar, 2016), and WMAttack under matched online episode budgets in full-system and cold-start settings. Module, retrieval, and LLM ablations assess component contributions. Appendix J specifies baseline implementations and execution conditions.

Metrics. We report mean clean and attacked episodic returns, normalized as $\widetilde{R}_x = R_x/(|R_{\text{clean}}|+1)$. Higher degradation $D = \widetilde{R}_{\text{clean}} - \widetilde{R}_{\text{attack}}$ indicates stronger attacks. Configurations are selected by confirmation-stage Utility: controlled comparisons test frozen configurations on independent episodes, whereas historical evaluations report confirmation returns. Flip denotes policy-output disagreement, and Time denotes evaluation time. Metric definitions and protocol mappings appear in Appendices C and J.3.

We first examine attack effectiveness and applicability, then search performance under fixed budgets and historical search progress, and finally component contributions.

5.2 Attack Effectiveness and Applicability

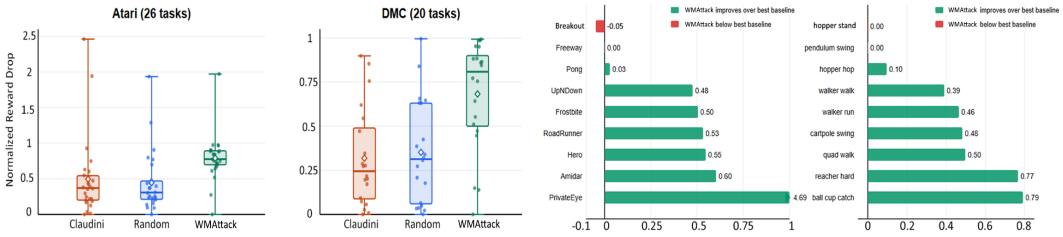
Framework-level attack effectiveness. We first examine the attack configurations found by WMAttack. Table 1 reports confirmation-stage outcomes from the historical DreamerV3 evaluation. WMAt-

tack attains lower mean normalized attacked return than both heuristic baselines: -0.077 versus $0.460/0.511$ on Atari and 0.266 versus $0.629/0.596$ on DMC (Claudini-style/Random). It also records the highest Utility on both suites and the highest DMC Flip. Section 5.3 complements these confirmation-stage results with independent tests under matched online budgets. Atari uses four attack families and DMC five, with historical running conditions specified in Appendix J.

Table 1: Historical DreamerV3 evaluation with returns normalized by each method's clean reference.

Victim	Suite	Tasks	Attacks	Method	Clean Return	Attacked Return ↓	Flip ↑	Utility ↑	Time (min)
DreamerV3	Atari	26	4	Claudini-style	0.957	0.460	0.400	-0.132	77.2
DreamerV3	Atari	26	4	Random	0.957	0.511	0.377	-0.163	73.4
DreamerV3	Atari	26	4	WMAttack	0.956	**-0.077**	**0.660**	**0.430**	106.2
DreamerV3	DMC	20	5	Claudini-style	0.948	0.629	0.705	-0.137	55.4
DreamerV3	DMC	20	5	Random	0.948	0.596	0.721	-0.123	59.0
DreamerV3	DMC	20	5	WMAttack	0.948	**0.266**	**0.800**	**0.174**	120.2

Task-level variation and attack-family selection. Figure 3 shows the historical reward-drop distribution and improvement over the stronger baseline. WMAttack achieves the highest reward drop on 24 of 26 Atari tasks and 18 of 20 DMC tasks. The strongest attack family also varies: APGD-CE is selected most often on Atari, whereas FAB dominates on DMC. This heterogeneity motivates searching multiple attack families; per-task results and selection statistics are provided in Appendix E. WMAttack's median task-level relative return reduction is 84.21%, and its mean remains 78.05% after excluding PrivateEye, both higher than those of the two baselines. Robustness summaries and metric definitions are provided in Appendix I.



(a) Reward-drop distribution (b) Per-task improvement

Figure 3: Historical task-level reward-drop distribution and per-task improvement on DreamerV3 Atari and DMC. WMAttack shifts the reward-drop distribution upward and improves over the stronger baseline on most displayed tasks.

Cross-architecture applicability. We further evaluate WMAttack on DreamerV2, TD-MPC2, and IRIS, achieving return degradation of 0.539 and 0.397 on DreamerV2 Atari and DMC, respectively, and 0.621 and 0.550 on TD-MPC2 DMC and IRIS Atari. These results reveal vulnerabilities in these models and demonstrate that WMAttack can be applied across world-model architectures, while highlighting architecture-dependent trade-offs between attack strength and evaluation cost. The corresponding metrics and architecture-dependent evaluation costs are as shown in Appendix H.

5.3 Controlled Search under Fixed Online Budgets

Independent-test gains under matched online episode budgets. To assess search under controlled resource constraints, Table 2 compares WMAttack with Coarse Grid, Random Search (Bergstra & Bengio, 2012), GP-UCB-style (Srinivas et al., 2010), and Successive Halving (Jamieson & Talwalkar, 2016) on 26 Atari and 20 DMC tasks. Each task–attack-family–seed unit searches a common 32-configuration menu at a radius of 10 pixel units using nine scout and nine confirmation episodes. The configuration with the highest confirmation-stage Utility is frozen before independent testing on ten attack and ten paired clean episodes. The four families are searched separately, using 72 online attack episodes per task–seed in total; clean-baseline and final-test episodes are additional. Reported test outcomes average the four separately selected family-level configurations.

Table 2: Independent-test degradation $D \uparrow$ under matched online episode budgets. Full WMAttack alone uses history; cold start excludes history. Bottom: gains over Successive Halving (SH) with 95% paired bootstrap CIs.

(a) Full method

Searcher	Atari	DMC
Coarse grid	0.755	0.503
Random Search	1.163	0.624
GP-UCB-style	1.289	0.767
Successive Halving	1.301	0.819
WMAttack	**1.439**	**0.952**

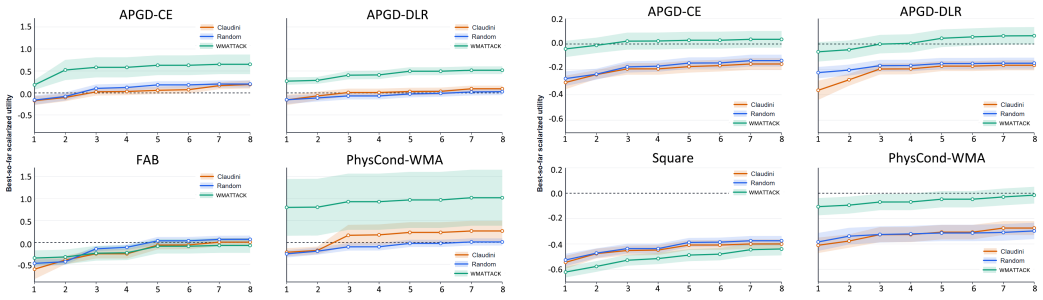
(b) Cold start

Searcher	Atari	DMC
Coarse grid	0.608	0.446
Random Search	1.095	0.597
GP-UCB-style	1.119	0.605
Successive Halving	1.121	0.623
WMAttack w/o RGAR	1.211	0.787

Paired comparison	Atari: ΔD [95% CI]	DMC: ΔD [95% CI]
Full − SH	$+0.138$ $[0.036, 0.195]$	$+0.133$ $[0.049, 0.266]$
Cold start − SH	$+0.090$ $[0.048, 0.104]$	$+0.164$ $[0.115, 0.249]$

Full WMAttack achieves mean Drop of $1.439/0.952$ on Atari/DMC, exceeding the highest-mean baseline, Successive Halving, by $0.138/0.133$. With history disabled for every method, the corresponding means are $1.211/0.787$ and gains are $0.090/0.164$. All four paired intervals in Table 2 lie above zero, supporting the search pipeline's advantage under this fixed-menu protocol, including cold start. Full WMAttack alone accesses history; online budgets are matched, with total-cost accounting specified in Appendix B.1. Baseline implementations and all paired comparisons appear in Appendix J.

Search progress across attack families. As shown in Figure 4, historical traces track running-best Utility over scout and confirmation records. WMAttack has higher running-best scores for APGD-CE, APGD-DLR, and the two-stage family; baselines are more competitive for FAB and Square. These stage-mixed traces complement the independent-test comparison in Table 2. Appendix F gives trajectories and threshold diagnostics; Section 5.4 examines component contributions.



(a) Atari (b) DMC

Figure 4: Historical search progress on DreamerV3 Atari and DMC. Curves show task-mean running-best Utility over scout and confirmation records, with standard-error bands across tasks. Each stage counts separately, including evaluations of the same configuration. Table 2 reports independent-test results.

5.4 Component Contributions

Overall module contributions. We first examine RGAR and SCAS jointly, then their internal designs. In the historical module comparison, adding SCAS increases final Drop from 0.645 to 0.665 with RGAR, but decreases it from 0.615 to 0.349 without RGAR, as shown in Table 3. Thus, SCAS's effect depends on retrieval conditions. The cold-start comparison instead evaluates the remaining pipeline against alternative searchers. First and Final refer to separate initial and final evaluation stages.

Table 3: Historical module ablation on DreamerV3 Atari. Cells report Utility / Drop / Flip.

Variant	First Utility / Drop / Flip ↑	Final Utility / Drop / Flip ↑
SCAS w/o RGAR	0.048 / 0.445 / 0.394	-0.325 / 0.349 / 0.396
RGAR + SCAS	**0.397 / 0.770 / 0.573**	**0.020 / 0.665 / 0.568**
RGAR only, w/o SCAS	0.245 / 0.744 / 0.580	-0.046 / 0.645 / 0.572
w/o RGAR and SCAS	0.214 / 0.687 / 0.604	-0.048 / 0.615 / 0.602

RGAR retrieval strategies. Table 4(a) reports mean D across four history-use strategies with SCAS, candidate spaces, and online budgets fixed. RGAR achieves the highest mean Drop. Its gains over behavior-based retrieval are 0.034 [95% CI $0.011, 0.059$] on Atari and 0.047 [$0.018, 0.074$] on DMC. Uniform sampling yields little or inconsistent improvement, whereas similarity-based retrieval improves both suite means. These results support composite internal representations, including dynamics-related features, as a complement to behavioral statistics; RGAR combines these descriptors with a fitted projection. Appendix G.2 details the history protocol for this 46-task comparison and the separate four-task experiment.

Table 4: RGAR retrieval and SCAS component ablations.

(a) RGAR retrieval strategies

Strategy	Atari D ↑	DMC D ↑
No history	1.227	0.739
Uniform	1.226	0.750
Behavior	1.249	0.784
RGAR	**1.283**	**0.831**

(b) SCAS component increments

Contrast	Mean ΔD ↑	95% paired CI
A–B	$+0.018$	$[+0.007, +0.028]$
A–C	$+0.011$	$[+0.003, +0.020]$
A–D	$+0.023$	$[+0.011, +0.035]$
B–D	$+0.005$	$[-0.007, +0.014]$

SCAS proposer and Auditor. We compare four variants across 46 tasks with RGAR disabled, retaining numerical history and matching candidate menus and online episode budgets: A = proposer + Auditor; B = programmatic selection + Auditor; C = proposer only; D = program-only. Table 4(b) reports paired increments under this separate SCAS protocol. The primary A–B contrast favors LLM selection over programmatic selection with the Auditor retained ($+0.018$). The exploratory contrasts favor retaining the Auditor with the proposer (A–C: $+0.011$) and full SCAS over program-only search (A–D: $+0.023$); all three paired 95% intervals exclude zero. The Auditor increment under programmatic selection remains unresolved (B–D). These results support additional gains from LLM-guided search within the shared construction and ranking framework. Appendix G.1 provides the protocol and component comparisons.

Flip's contribution to configuration selection. Finally, controlled ablations on four representative tasks show that retaining the Flip term in Utility increases mean independent-test normalized return degradation by 0.029–0.082, supporting its role as a complementary selection signal in these settings. The comparison changes only the Utility weight of Flip; its use in numerical feedback and candidate ranking is retained. Diagnostics and ablation results are reported in Appendix C.1.

6 Conclusion

We introduced **WMAttack**, a framework for attack-configuration search against fixed world-model agents under explicit observation constraints and limited online rollout budgets. Controlled comparisons across 46 DreamerV3 tasks support a mean independent-test advantage over the evaluated search baselines, including under cold start. Controlled ablations further show gains from composite retrieval over behavior-based history and from LLM-based feedback interpretation and candidate selection within a shared programmatic search framework. Evaluations on additional architectures further illustrate the framework's applicability. Together, these findings demonstrate the effectiveness of WMAttack's search procedure within the evaluated candidate menus and online rollout budgets.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and modifying supporting code and checking consistency between experimental records and the manuscript. We have not used generative AI tools for theoretical derivations or proofs, experimental design, interpretation of research results, or manuscript translation, and qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to improve manuscript readability and organize supplementary materials. We have manually reviewed all AI-assisted work, including the relevant code and the results retained in the supplementary materials against the original experimental records. The relevant code also underwent syntax checks and core protocol tests. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, 2020.

András Balogh and Márk Jelasity. Verification of the implicit world model in a generative model via adversarial sequences. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=BLOIB8CwBI.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. URL https://www.jmlr.org/papers/v13/bergstra12a.html.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020a.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2020b.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: A standardized adversarial robustness benchmark. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Siyuan Duan, Ke Zhang, and Xizhao Luo. Trap: Tail-aware ranking attack for world-model planning. *arXiv preprint arXiv:2605.01950*, 2026.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Zhixiang Guo, Siyuan Liang, Andras Balogh, Noah Lunberry, Rong-Cheng Tu, Mark Jelasity, and Dacheng Tao. When world models dream wrong: Physical-conditioned adversarial attacks against world models, 2026. URL https://arxiv.org/abs/2602.18739.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *International Conference on Learning Representations Workshop*, 2017.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 240–248, 2016. URL https://proceedings.mlr.press/v51/jamieson16.html.

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *International Joint Conference on Artificial Intelligence*, 2017.

Shengcai Liu, Fu Peng, and Ke Tang. Reliable robustness evaluation via automatically constructed attack ensembles. In *AAAI Conference on Artificial Intelligence*, 2023.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

Alexander Panfilov, Peter Romov, Igor Shilov, Yves-Alexandre de Montjoye, Jonas Geiping, and Maksym Andriushchenko. Claudini: Autoresearch discovers state-of-the-art adversarial attack algorithms for llms. *arXiv preprint arXiv:2603.24511*, 2026.

Zahra Shahrooei, Mykel J. Kochenderfer, and Ali Baheri. Optimizing falsification for learning-based control systems: A multi-fidelity bayesian approach. *arXiv preprint arXiv:2409.08097*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010. URL https://arxiv.org/abs/0912.3995.

Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep RL. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JM2kFbJvvI.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Rishi Upadhyay, Howard Zhang, Jim Solomon, Ayush Agrawal, Pranay Boreddy, Shruti Satya Narayana, Yunhao Ba, Alex Wong, Celso M. de Melo, and Achuta Kadambi. WorldBench: Disambiguating physics for diagnostic evaluation of world models. *arXiv preprint arXiv:2601.21282*, 2026. URL https://arxiv.org/abs/2601.21282v1.

Keyon Vafa, Justin Y. Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. Evaluating the world model implicit in a generative model. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Shuhan Xu, Siyuan Liang, Hongling Zheng, Yong Luo, Han Hu, Lefei Zhang, and Dacheng Tao. Ctrlattack: A unified attack on world-model control in diffusion models. *arXiv preprint arXiv:2603.13435*, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations*, 2024.

Chengyuan Yao, Pavol Bielik, Petar Tsankov, and Martin Vechev. Automated discovery of adaptive attacks on adversarial defenses. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/e1c13a13fc6b87616b787b986f98a111-Abstract.html.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In *Advances in Neural Information Processing Systems*, 2020.

Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2101.08452.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. STORM: Efficient stochastic transformer based world models for reinforcement learning. *arXiv preprint arXiv:2310.09615*, 2024.

A Victim and Threat Model Details

We describe clean and attacked closed-loop evaluation for a fixed world-model agent

$$M_\theta = (\phi_\theta, f_\theta, \pi_\theta), \tag{9}$$

where ϕ_θ is the encoder or representation module, f_θ is the latent dynamics model, and π_θ is the policy.

Let x_t denote the environment state, which need not be observable, and let z_t summarize the victim's observation–action history. Under clean evaluation,

$$z_t = \phi_\theta(o_{\le t}, u_{<t}), \qquad u_t \sim \pi_\theta(\cdot \mid z_t), \qquad x_{t+1}, o_{t+1}, r_t \sim \mathcal{E}(\cdot \mid x_t, u_t). \tag{10}$$

Here $\mathcal{E}$ combines state transition, observation generation, and reward. Under configuration c, the agent instead receives perturbed observations from its own attacked trajectory:

$$\tilde{z}_t = \phi_\theta(\tilde{o}_{\le t}, \tilde{u}_{<t}), \qquad \tilde{u}_t \sim \pi_\theta(\cdot \mid \tilde{z}_t), \qquad x^c_{t+1}, o^c_{t+1}, r^c_t \sim \mathcal{E}(\cdot \mid x^c_t, \tilde{u}_t). \tag{11}$$

Perturbing the current observation o^c_t produces $\tilde{o}_t$ without changing $\mathcal{E}$ or the victim parameters, although altered actions can change subsequent states. Observations need not be Markov states. Below, o_t denotes the clean observation at the current attacked decision point.

In the DreamerV3 implementation, attacks modify the preprocessed CNN observation tensor, including historical slices when frame stacking is enabled. For each image key $k \in \mathcal{K}_{\text{img}}$,

$$\tilde{o}_{t,k} = \Pi_{[-0.5,0.5]}\left(o_{t,k} + \delta_{t,k}\right), \qquad \|\delta_{t,k}\|_\infty \le \epsilon/255. \tag{12}$$

Inputs use $x/255 - 0.5$ coordinates and pixel-unit radii ϵ. Projection bounds perturbations of valid clean tensors by the configured radius; the realized norm is defined separately in Appendix B. Adapter-specific clipping uses $[0, 1]$ for IRIS and no additional input-range clip for TD-MPC2; both retain the attack-radius constraint. Non-image keys are left unchanged:

$$\tilde{o}_{t,k} = o_{t,k}, \qquad k \notin \mathcal{K}_{\text{img}}. \tag{13}$$

The attacker may choose the attack configuration $c \in \mathcal{C}_\tau$, including the attack family and hyperparameters, but cannot modify the environment dynamics, reward function, policy parameters, world-model parameters, checkpoint, or training data.

Stochasticity and attribution. The original DreamerV3 path clones internal states without restoring random-number generator states between policy probes. Additional victim forward passes can therefore advance randomness before action execution, even with matched episode seeds. The separate diagnostic in Appendix C.1 controls initial randomness and restores generator states after probing. Its validation scope is specified in Appendix B.

B Attack Configuration Space

Configurations specify attack family, radius ϵ, steps, restarts, ρ, and allocation rule; seeds are reproducibility metadata. Historical search starts from a canonical grid and permits bounded numerical expansion, whereas the controlled comparisons use the fixed menus in Appendix J.2.

Perturbation units. DreamerV3 maps $x \in [0, 255]$ to $z = x/255 - 0.5 \in [-0.5, 0.5]$, so $\epsilon_{\text{in}} = \epsilon_{\text{pix}}/255$. The controlled radius of 10 pixel units is therefore $10/255 \approx 0.039216$ in input space. Pixel units refer to preprocessed-image intensities; other architectures retain their own input transformations.

Constraint implementation. For the current clean input z_t and a candidate perturbation δ_t, the projection implementation first bounds the perturbation elementwise and then clips the result to the valid input range:

$$z_t^{\text{adv}} = \text{clip}_{[-0.5,\,0.5]}\left(z_t + \text{clip}_{[-\epsilon_{\text{in},t},\,\epsilon_{\text{in},t}]}(\delta_t)\right).$$

For finite valid clean inputs, this projection constrains the complete image tensor, including all stacked pixels and channels. The realized pixel-unit perturbation at each decision point is

$$d_t = 255\left\|z_t^{\text{adv}} - z_t\right\|_\infty.$$

A budget violation is defined by $d_t > \epsilon_{\text{pix},t} + \tau$, where τ is a numerical tolerance in pixel units; input-range violations use the corresponding normalized tolerance $\tau/255$.

Validation scope. The projection equations define the input constraints. The paired-probe diagnostic in Appendix C.1 evaluates randomness control for policy-output measurements; full-pipeline validation of realized perturbation bounds and zero-perturbation trajectory equivalence is outside this diagnostic's scope.

WMAttack evaluates five attack families:

- **APGD-CE**: an adaptive projected-gradient attack using cross-entropy for discrete actions and the negative-log-likelihood objective defined below for continuous actions.
- **APGD-DLR**: an adaptive projected-gradient attack using a difference-of-logits-ratio objective for discrete actions (with a margin-style fallback for fewer than three actions), and the DLR-style action-shift objective defined below for continuous actions.
- **FAB**: a FAB-inspired variant adapted to policy outputs. For continuous actions, it targets a mean absolute action deviation exceeding 0.05, using sign-gradient updates, contraction toward the clean observation, and projection onto the perturbation and input constraints.
- **Square Attack**: a model-guided adaptation that uses gradient-based saliency to select square perturbation regions.
- **PhysCond-WMA**: a PhysCond-WMA-inspired two-stage adaptation targeting the evaluated world-model agent.

Continuous-action objectives. For continuous-action policies, the CE adaptation maximizes the negative log-likelihood of the clean reference action under the perturbed policy distribution, $L_{\text{CE}} = -\operatorname{mean} \log \pi_{\tilde{o}}(\bar{a}_{\text{clean}})$, where $\bar{a}_{\text{clean}} = \operatorname{clip}(\operatorname{stopgrad}(a_{\text{clean}}), -0.999, 0.999)$. The continuous-action DLR-style adaptation instead maximizes the scale-normalized squared action shift, $L_{\text{shift}} = \operatorname{mean}[((a_{\text{adv}} - \operatorname{stopgrad}(a_{\text{clean}}))/(\sigma_{\text{adv}} + 10^{-6}))^2]$, providing an action-space objective for continuous policies. Here, a denotes the reference action output returned by the attack interface, and σ is its distribution scale parameter. For tanh-normal policies, σ denotes the scale of the normal distribution before the tanh transformation.

Two-stage observation-space adaptation. The PhysCond-WMA-inspired variant perturbs image observations using a policy attack loss and a feature-consistency penalty. Let L_{atk} denote cross-entropy against the clean greedy action for discrete policies, summed over action heads, or the continuous-action negative-log-likelihood objective above. With feature groups $\mathcal{G}$ comprising encoder, recurrent, stochastic, actor-hidden, and policy features, define

$$L_{\text{feat}} = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \text{MSE}\Big(\widehat{f}_g(\tilde{o}), \widehat{f}_g(o)\Big) .$$

Each feature vector is flattened and L_2-normalized with denominator floor 10^{-12}; clean features are detached. Historical agent state is fixed during optimization, as specified below.

For the allocated step budget T, stage one uses at most $T_1 = \max(1, \operatorname{round}(0.4T))$ projected sign-gradient updates to maximize $L_{\text{atk}} - 0.1 L_{\text{feat}}$. It stops when the current feature penalty reaches 0.15, or rejects a proposed update whose penalty exceeds 0.15. Stage two starts from the resulting iterate and uses at most $T_2 = \max(1, T - T_1)$ updates to maximize $L_2 = L_{\text{atk}} - 0.6 L_{\text{feat}}$; unused stage-one steps are not reassigned. For each perturbed image tensor,

$$m_{k+1} = 0.9 m_k + 0.1 \frac{\nabla L_2(\tilde{o}_k)}{\max(\operatorname{mean} |\nabla L_2(\tilde{o}_k)|, 10^{-12})},$$

$$\tilde{o}_{k+1} = \Pi(\tilde{o}_k + \eta \ \operatorname{sign}(m_{k+1})) , \qquad \eta = \max\left(\frac{\epsilon}{255T}, \frac{1}{255} \right) .$$

Here $m_0 = 0$, ϵ is the configuration's radius in pixel units, and Π enforces the perturbation budget and the adapter's valid-input range. Initialization uses uniform noise within the perturbation budget; both stages project each update. Stage two checks the current iterate for a changed greedy action, or a mean absolute continuous-action shift above 0.05, and stops on success. Stored stage-two iterates are selected by success status, then by L_2. The feature threshold controls stage-one switching; stage two uses the feature penalty in its objective.

Architecture-specific gradient paths. DreamerV2/V3 gradients pass through the current-observation encoder, latent-state computation, and actor with historical state fixed, without back-propagating through environment trajectories. TD-MPC2 uses an encoder–policy-prior surrogate for attack gradients and the actual agent for closed-loop evaluation, without differentiating through the full planner.

Table 5 gives initialization ranges. Historical runs under all three method labels could halve lower bounds and double upper bounds, yielding ϵ ranges 1–40 for APGD, FAB, and PhysCond-WMA and 1–32 for Square, with step ranges 2–48, 3–64, and 10–320, respectively. Appendix J.1 specifies each method's initialization, history access, and episode allocation.

Table 5: Canonical initialization ranges; the bounded proposal expansion is specified in the accompanying text.

Attack	ϵ candidates	Step candidates	Allocation modes
APGD-CE	2–20	4–24	fixed, margin-linear
APGD-DLR	2–20	4–24	fixed, margin-linear
FAB	2–20	6–32	fixed, margin-linear
PhysCond-WMA	2–20	6–32	fixed, margin-linear
Square	2–16	20–160	fixed, margin-linear

Margin-linear allocation. Fixed allocation uses the configuration's step count and radius at every decision. For margin-linear allocation, let κ_t denote clean-policy confidence: the top-two logit gap, averaged over action heads, for discrete policies, or the reciprocal of the mean absolute policy standard deviation for continuous policies, with denominator floor 10^{-6}. The allocator computes

$$h_t = \mathrm{clip}\left(\frac{\kappa_t - \kappa_{\mathrm{low}}}{\kappa_{\mathrm{high}} - \kappa_{\mathrm{low}}}, 0, 1\right),$$
$$T_t = \max(1, \mathrm{round}[T_{\min} + h_t(T - T_{\min})]).$$

The default thresholds are $\kappa_{\mathrm{low}} = 0.1$ and $\kappa_{\mathrm{high}} = 1.5$. The candidate constructor sets both radius-scaling endpoints to one, so the radius remains fixed within each configuration while the allocated step count varies with confidence. Historical search can select different radii across configurations; the controlled menu fixes the radius at 10 pixel units. Evaluation records retain restarts, ρ, seeds, and allocation settings. Proposal backends, history access, and selection follow Section 5.1.

B.1 SCAS Candidate Construction and Selection

Structured feedback and candidate construction. The Auditor converts evaluation records, task information, and available history into adjustment directions, optional target values, and allocation preferences. Fixed rules construct candidates, and the proposer selects shortlist identifiers. Sources are the base grid and local variants anchored to the current best, up to four retrieved configurations, and configurations from the four most recent reflections. Initialization is task-conditioned; current-batch feedback affects later rounds.

Edits and constraints. Local edits change radius and steps by one grid increment; recommended increases/decreases also use factors 1.5/0.75 and any Auditor target values. Values are clipped to allowed bounds, with two-decimal radii and integer steps. Task, attack family, parameters, seed, and allocation settings form the key used to deduplicate candidates and exclude previous proposals. Shared-menu experiments additionally reject off-menu or mismatched configurations before execution.

Heuristic ranking. Candidates are ranked in descending order of a fixed priority score. Given a current best configuration c^*, the base score is

$$S_{\mathrm{base}}(c) = -|\epsilon - \epsilon^*| - \frac{|T - T^*|}{6} + 0.3\mathbf{1}[m = m^*] - \frac{\mathbf{1}[m \neq \mathrm{fixed}]}{6}|\tilde{\ell} - \tilde{\ell}^*|,$$

where ϵ, T, m, and ℓ denote perturbation magnitude, step count, allocation mode, and minimum step count. Missing minimum-step values are replaced by T for the candidate and $\max(1, \lfloor T^*/2 \rfloor)$ for the best configuration. Before a current-task result is available, the base score is

$$\begin{aligned} S_{\text{base}}(c) = & -|\epsilon - \epsilon_{\text{mid}}| - \frac{|T - T_{\text{mid}}|}{10} - 0.75|\epsilon - \epsilon_0| - \frac{|T - T_0|}{5} \\ & + 1.25\mathbf{1}[m = \text{margin-linear}] \\ & + \sum_{r \in \mathcal{R}} w_r \left(0.4 - 0.25|\epsilon - \epsilon_r| - 0.05|T - T_r| + 0.6\mathbf{1}[m = m_r]\right). \end{aligned}$$

The subscript mid denotes the upper median of the corresponding grid, (ϵ_0, T_0) are task-conditioned defaults, and $\mathcal{R}$ contains at most four retrieved configurations with retrieval scores w_r; the sum is zero without retrieved history. Both cases add the same feedback and time-budget terms:

$$\begin{aligned} S(c) = S_{\text{base}}(c) + \sum_j & \left(0.5 b_{\epsilon,j} \epsilon + 0.1 b_{T,j} T + 1.5\mathbf{1}[m = p_j] - 1.5\mathbf{1}[m = q_j]\right) \\ & + 0.5\mathbf{1}[B_{\text{time}} < 300,\ m = \text{margin-linear}]. \end{aligned}$$

The sum uses at most three recent audits, with biases $b_{\epsilon,j}, b_{T,j} \in [-2, 2]$ and preferred/avoided modes p_j, q_j; absent preferences contribute zero. B_{time} is in seconds. These fixed heuristic coefficients determine shortlisting; confirmation and final selection use evaluation Utility.

SCAS: one-round candidate selection and feedback update

In	Base space, proposed keys, evaluation/audit records, available experience, shortlist size K, batch size b.
1	**Construct:** generate local and recommendation-based variants from best, retrieved, and recent-reflection anchors; initialize from task information when feedback/history is unavailable.
2	**Constrain:** clip and type parameters; merge adaptive candidates before the base grid; retain the first occurrence per key and exclude proposed keys. Apply shared-menu filtering when required.
3	**Rank:** stably sort by descending $S(c)$; score ties retain generation/merge order.
4	**Shortlist:** retain the top K and up to four tail candidates with distinct (ϵ, T, m, ℓ) signatures; retain the selected tail candidates' relative order.
5	**Select:** assign identifiers and ask the proposer for b selections; apply the interface-specific invalid-output policy below.
6	**Evaluate:** scout, promote by Utility, and confirm. The strict controlled interface breaks Utility ties by lexicographic configuration key.
7	**Update:** audit evaluation results and retain recommendations/reflections for later rounds.
Out	Updated search state; at termination, select by confirmation Utility. Strict controlled held-out results are not used for selection.

Invalid-output handling. Structured outputs. The proposer selects configuration identifiers from the supplied shortlist. Its JSON response contains `candidate_ids`, `summary`, and `per_candidate_notes`. The Auditor returns `failure_tags`, `summary`, `root_cause`, `recommendations`, and `strategy`; these fields guide subsequent programmatic candidate construction and ranking. The general interface ignores invalid or duplicate identifiers, fills missing slots in shortlist order, and falls back to leading entries on parsing failure; Auditor exceptions invoke heuristic auditing. The strict controlled interface instead requires the requested valid, distinct identifiers and terminates on backend or response-validation failure, without automatic filling or heuristic substitution.

Evaluation-cost accounting. For each task–family search unit, $B_{\text{search}} = B_{\text{scout}} + B_{\text{confirm}} \le B$. A complete episode ledger additionally separates clean-reference, post-search test, and offline-diagnostic evaluations:

$$B_{\text{total}} = B_{\text{search}} + B_{\text{clean}} + B_{\text{test}} + B_{\text{diagnostic}}. \tag{14}$$

Each physical episode is counted once, including shared clean evaluations reused across arms. Historical-memory costs are reported separately. The matched budget covers online scout and confirmation episodes. Total cost additionally depends on episode horizons, attack steps and gradient evaluations, model loading, LLM calls, and the separately accounted evaluation stages. Independent tests freeze the Utility-selected configuration, use fresh seeds with paired clean/attack seeds where specified, and provide no search feedback. Historical confirmation results in Section 5.1 remain selection-stage measurements.

C Utility Components

For task τ and configuration c, let $J_{\text{clean}}(\tau)$ and $J_{\text{adv}}(\tau, c)$ denote expected clean and attacked episodic returns.

The normalized reward degradation is defined as

$$D_\tau(c) = \frac{J_{\text{clean}}(\tau) - J_{\text{adv}}(\tau, c)}{|J_{\text{clean}}(\tau)| + 1}. \tag{15}$$

Larger $D_\tau(c)$ means stronger degradation; normalization accommodates task reward scales and zero clean returns. Empirical values use mean clean and attacked returns within each evaluation unit and are not clipped; negative attacked returns can produce $D_\tau(c) > 1$.

Flip $F_\tau(c)$ compares policy outputs for clean and perturbed observations from the same cloned pre-decision state on an attacked trajectory. For discrete outputs a_t^{ref} and a_t^{pert},

$$f_t = \mathbb{I}[a_t^{\text{ref}} \neq a_t^{\text{pert}}], \tag{16}$$

where the action identifiers are obtained from the policy-output argmax. For a continuous action vector of dimension d_a, the probes use the actor's reported mean/mode actions, and the implemented rule is

$$f_t = \mathbb{I}\left[\frac{1}{d_a}\|a_t^{\text{ref}} - a_t^{\text{pert}}\|_1 > 0.05\right]. \tag{17}$$

Flip pools disagreement indicators across evaluated decisions and episodes. The threshold 0.05 is diagnostic, and probe outputs differ from the sampled action executed in the environment. DreamerV3 latent inference remains stochastic even for argmax or mean/mode outputs: internal states are cloned, but generator states are not restored between probes. Flip can therefore include both perturbation and sampling effects. Clean runs record zero by convention, without a clean–clean disagreement measurement. Its interpretation as an auxiliary diagnostic and its contribution to search are examined separately below.

Runtime $T_\tau(c)$ is measured in seconds and records the total evaluation-loop time for the corresponding scout or confirmation stage, including resets, policy probes, attack optimization, and environment interaction. It excludes prior environment/model construction and LLM calls. The penalty $\log(1 + T_\tau(c))$ uses this value directly; comparisons must therefore specify episode allocation, horizon, hardware, and timing boundaries.

The rollout-variability term is defined as

$$V_\tau(c) = \frac{\text{Std}[R_{\text{adv}}(c)]}{|J_{\text{clean}}(\tau)| + 1}, \tag{18}$$

where $R_{\text{adv}}(c)$ denotes episodic returns from repeated attacked rollouts under configuration c. The implementation uses empirical clean/attack means and population-style return standard deviation with denominator n. Single-episode scout dispersion is zero by convention. Episode count changes the runtime and dispersion scales of scout and confirm Utilities. These are stage-specific search scores; confidence intervals follow each experiment's statistical protocol. Final selection compares confirmation scores at a common fidelity.

Combining these components, WMAttack uses the scalarized utility

$$U_\tau(c) = D_\tau(c) + w_f F_\tau(c) - w_r \log(1 + T_\tau(c)) - w_v V_\tau(c). \tag{19}$$

In our implementation, we use $w_f = 0.25$, $w_r = 0.15$, and $w_v = 0.05$. Utility trades off return degradation, policy-output disagreement, evaluation time, and return dispersion.

C.1 Flip Randomness Diagnostics and Utility-Weight Ablation

Metric definition and randomness control. Flip measures policy-output disagreement between clean and perturbed observations from the same initial internal state. For discrete actions,

$$F = \frac{1}{N}\sum_{t=1}^{N} \mathbb{I}[a_t^{\text{clean}} \neq a_t^{\text{adv}}].$$

For continuous actions, a disagreement is recorded when the mean absolute difference across action dimensions exceeds 0.05. Flip complements episodic-return measurements with a local policy-output diagnostic.

To separate sampling from perturbation effects, the supplementary diagnostic pairs clean–clean and clean–perturbed probes under shared or different initial generator states. Both calls start from the same internal state, and generator states are restored afterward. This isolates probing randomness from the executed trajectory, unlike the original path in Appendix A. Probe-on/off trajectory-equivalence checks passed on the tested Atari and DMC runs.

Paired-probe setup. We evaluate Alien, Pong, acrobot swingup, and cheetah run with APGD-CE, APGD-DLR, FAB, and PhysCond-WMA. Three episodes per task–family and ten probed decisions per episode yield 48 episodes and 480 decisions. Each family uses a prespecified 10-pixel menu configuration, without selection by attack effectiveness.

Diagnostic statistics. Diagnostic intervals use 5,000 percentile-bootstrap resamples (seed 20260922), aggregating decision indicators within each episode and resampling whole episodes within fixed task–attack-family strata, with shared indices across comparisons and equal stratum and episode weights. Decisions within one episode are not treated as independent repetitions.

Table 6: Paired-probe disagreement rates. Each task includes 12 episodes and 120 probed decisions; parentheses show disagreement counts. Shared and different RNG refer to the initial random-generator states of the paired calls.

Task	Clean–clean Shared RNG	Clean–clean Different RNG	Clean–perturbed Shared RNG	Clean–perturbed Different RNG
Alien	0.00% (0/120)	15.00% (18/120)	43.33% (52/120)	46.67% (56/120)
Pong	0.00% (0/120)	30.83% (37/120)	44.17% (53/120)	51.67% (62/120)
acrobot swingup	0.00% (0/120)	19.17% (23/120)	20.00% (24/120)	29.17% (35/120)
cheetah run	0.00% (0/120)	72.50% (87/120)	78.33% (94/120)	94.17% (113/120)

With different random-number generator states, repeated clean-input probes also disagree, showing that random sampling contributes to unsynchronized Flip. With shared initial random-number generator states, clean–clean disagreement is zero on all four tasks, whereas clean–perturbed disagreement remains nonzero. Under controlled probing, the remaining clean–perturbed disagreement captures policy sensitivity to input perturbations and complements return measurements.

The paired diagnostic characterizes input sensitivity under shared randomness, whereas the original unsynchronized Flip retains the sampling contribution described above. Across all four tasks, clean–clean disagreement with different random-number generator states is 34.38% (95% CI [31.04%, 37.51%]), while clean–perturbed disagreement is 46.46% [43.13%, 50.00%] with shared states and 55.42% [51.67%, 58.96%] with different states.

Controlled ablation of the Flip weight in Utility. To test whether using Flip for configuration selection improves final search effectiveness, we compare the original Flip weight $\lambda_F = 0.25$ with $\lambda_F = 0$, keeping the other weights unchanged:

$$U_{\lambda_F} = D + \lambda_F F - 0.15 \log(1 + T) - 0.05V.$$

Both arms disable RGAR and share construction, constraints, menus, feedback, and online budgets. Flip remains in numerical feedback and ranking, isolating its Utility weight while allowing selection-induced trajectory differences.

Setup. Four tasks, four families, three repetitions, and two arms yield 48 complete pairs (96 units). The shared 32-configuration, 10-pixel menu, online budget, and independent-test allocation follow Appendix J.2. Arms share episode seeds and run on the same GPU in balanced order. All units passed independent validation; all 48 pairs had identical clean-test returns. The 4,128 formal episodes exclude 172 pre-validation episodes and the separately counted 48 diagnostic episodes.

Evaluation and statistics. Final effectiveness is measured by normalized return degradation on independent tests,

$$D = \frac{\bar{R}_{\text{clean}} - \bar{R}_{\text{attack}}}{|\bar{R}_{\text{clean}}| + 1}, \qquad \Delta D = D_{\lambda_F=0.25} - D_{\lambda_F=0},$$

where positive ΔD favors the original weight. We use 5,000 hierarchical paired-bootstrap resamples (seed 20260922) over repetitions and paired test episodes within fixed task–family strata. Arms share indices; strata and repetitions receive equal weight; each resample recomputes D and ΔD. Percentiles give 95% CIs. The overall contrast is primary; task-wise contrasts are exploratory and unadjusted for multiplicity. Inference concerns these four fixed tasks, separately from the 46-task comparisons.

Table 7: Completed Flip-weight ablation on independent tests. All 48 pairs are included. Larger D indicates stronger attacks; positive ΔD favors the original weight. Intervals are 95% hierarchical paired-bootstrap CIs conditional on the fixed task–attack-family strata.

Task	Pairs	Original D	Zero-weight D	Mean ΔD	95% paired CI
Alien	12	0.580	0.529	$+0.051$	$[+0.035, +0.077]$
Pong	12	1.988	1.959	$+0.029$	$[+0.008, +0.044]$
acrobot swingup	12	0.969	0.902	$+0.067$	$[+0.034, +0.102]$
cheetah run	12	0.768	0.686	$+0.082$	$[+0.047, +0.117]$

Results. In Table 7, the original weight improves mean independent-test normalized return degradation over the zero-weight variant by 0.051, 0.029, 0.067, and 0.082 on Alien, Pong, acrobot swingup, and cheetah run, respectively. All reported task-wise 95% paired confidence intervals lie above zero. These results support retaining Flip as a complementary Utility-based selection signal under the evaluated tasks, configuration menus, and budgets.

D Theoretical Details

The first three subsections analyze idealized proposal distributions at fixed task, victim, fidelity, and evaluation conditions. They assume a nonempty finite admissible set $\mathcal{C}_\tau$ containing the full proposal support and finite population utilities $U_\tau(c)$. These distributions describe an idealized proposal process; the implementation constructs and ranks candidates directly. Appendix D.4 separately proves the coverage–estimation decomposition used in Section 4.4.

D.1 Utility-Biased Reference and Conditional Discovery

For $\eta \geq 0$, define

$$U_\tau^\star = \max_{c \in \mathcal{C}_\tau} U_\tau(c), \qquad \mathcal{G}_\eta = \{c \in \mathcal{C}_\tau : U_\tau(c) \geq U_\tau^\star - \eta\}. \tag{20}$$

A utility-biased reference is the Gibbs distribution

$$q^\star(c) = \frac{\exp(\beta U_\tau(c))}{\sum_{c' \in \mathcal{C}_\tau} \exp(\beta U_\tau(c'))}, \qquad \beta \geq 0. \tag{21}$$

It is uniform at $\beta = 0$ and increasingly concentrates on utility maximizers as β grows. This analytical reference is defined by population utilities, while WMAttack uses empirical evaluations. Let $p = q(\mathcal{G}_\eta)$ and $r = q^\star(\mathcal{G}_\eta)$. For $\gamma \geq 0$, the ideal correction satisfies

$$C_\gamma(q) = \frac{q + \gamma q^\star}{1 + \gamma}, \qquad C_\gamma(q)(\mathcal{G}_\eta) - p = \frac{\gamma}{1 + \gamma}(r - p). \tag{22}$$

For $\gamma > 0$, it improves effective-set mass precisely when $r > p$. Gibbs weighting alone does not ensure this inequality for an arbitrary current q. In the ideal case $r = 1$, outside mass contracts as $1 - C_\gamma(q)(\mathcal{G}_\eta) = (1 - p)/(1 + \gamma)$.

Conditional discovery time. Let $T_{\text{hit}} \in \{1, 2, \ldots\}$ be the first proposal round containing a member of $\mathcal{G}_\eta$. Suppose each round draws $b \geq 1$ candidates independently from q_t, conditional on the preceding history. Its conditional hit probability is

$$h_t = 1 - (1 - p_t)^b, \qquad p_t = q_t(\mathcal{G}_\eta). \tag{23}$$

If $p_t \geq p_{\min} > 0$ after every history with no previous hit, then $h_t \geq h_{\min} = 1-(1-p_{\min})^b$. Iterated conditioning gives $\Pr(T_{\text{hit}} > t) \leq (1-h_{\min})^t$ for every integer $t \geq 0$, and hence

$$\mathbb{E}[T_{\text{hit}}] = \sum_{t=0}^{\infty} \Pr(T_{\text{hit}} > t) \leq \frac{1}{h_{\min}}. \tag{24}$$

The bound permits dependence across rounds and requires independent draws within each round together with a uniform conditional lower bound before the first hit. It measures discovery time in proposal rounds. Applying this sampling model to ranked, deduplicated proposals additionally requires establishing the stated within-round independence.

D.2 Noisy Correction

Suppose a realized update satisfies the one-sided effective-set mass condition

$$q_{\text{new}}(\mathcal{G}_\eta) \geq C_\gamma(q)(\mathcal{G}_\eta) - \xi, \qquad \xi \geq 0. \tag{25}$$

Subtracting p yields

$$q_{\text{new}}(\mathcal{G}_\eta) - p \geq \frac{\gamma}{1+\gamma}(r-p) - \xi, \qquad \gamma \geq 0. \tag{26}$$

Therefore,

$$\frac{\gamma}{1+\gamma}(r-p) > \xi \quad \Longrightarrow \quad q_{\text{new}}(\mathcal{G}_\eta) > p. \tag{27}$$

Here ξ bounds approximation error in effective-set probability mass. The sufficient improvement condition compares the ideal mass gain with this update error. Connecting retrieval, feedback, or finite-evaluation errors to ξ requires a model relating their implemented candidate-ranking updates to the ideal correction C_γ.

D.3 Conditional Comparison of Proposal Updates

Consider two idealized updates, indexed by A and B, with a common initial mass p and reference mass r, but strengths $\gamma_A, \gamma_B \geq 0$. Their ideal mass difference is

$$\Delta_{\text{ideal}} = C_{\gamma_A}(q)(\mathcal{G}_\eta) - C_{\gamma_B}(q)(\mathcal{G}_\eta) = \frac{(\gamma_A - \gamma_B)(r-p)}{(1+\gamma_A)(1+\gamma_B)}. \tag{28}$$

To compare realized updates, assume the explicit two-sided error bounds

$$\left| q_j^{\text{real}}(\mathcal{G}_\eta) - C_{\gamma_j}(q)(\mathcal{G}_\eta) \right| \leq \xi_j, \qquad j \in \{A, B\}, \quad \xi_j \geq 0. \tag{29}$$

Using the lower bound for A and upper bound for B gives

$$q_A^{\text{real}}(\mathcal{G}_\eta) - q_B^{\text{real}}(\mathcal{G}_\eta) \geq \Delta_{\text{ideal}} - \xi_A - \xi_B. \tag{30}$$

Thus, A has greater effective-set mass if $\Delta_{\text{ideal}} > \xi_A + \xi_B$; in particular, a positive ideal gap requires $(\gamma_A - \gamma_B)(r-p) > 0$. The two-sided bounds above support this ordering at a shared starting mass, extending the one-sided analysis in Appendix D.2. The comparison is conditional on the supplied correction strengths and approximation bounds; ranking concrete search methods requires establishing these quantities for those methods.

D.4 Search Coverage and Finite-Evaluation Error

Here only the nonempty confirmed set $S \subseteq \mathcal{C}_\tau$ must be finite; the admissible space may be larger or continuous, provided $U_\tau^\star = \sup_{c \in \mathcal{C}_\tau} U_\tau(c) < \infty$. The following deterministic statement separates limited coverage from imperfect evaluation. Suppose, for some $e, \eta_{\text{sel}} \geq 0$,

$$\max_{c \in S} |\widehat{U}(c) - U_\tau(c)| \leq e, \qquad \widehat{U}(\widehat{c}) \geq \max_{c \in S} \widehat{U}(c) - \eta_{\text{sel}}, \quad \widehat{c} \in S. \tag{31}$$

Then

$$U_\tau^\star - U_\tau(\widehat{c}) \leq \underbrace{U_\tau^\star - \max_{c \in S} U_\tau(c)}_{\text{coverage gap}} + \eta_{\text{sel}} + 2e. \tag{32}$$

Proof. Let $c_S^\star \in \arg\max_{c \in S} U_\tau(c)$. The two assumptions imply

$$U_\tau(\widehat{c}) \geq \widehat{U}(\widehat{c}) - e \geq \widehat{U}(c_S^\star) - \eta_{\text{sel}} - e \geq U_\tau(c_S^\star) - \eta_{\text{sel}} - 2e. \tag{33}$$

Adding the gap between $U_\tau^\star$ and $U_\tau(c_S^\star)$ proves the claim. Exact maximization of recorded confirm Utility corresponds to $\eta_{\text{sel}} = 0$.

The inequality applies to adaptively generated S under the stated uniform-error condition, with e treated as an assumed error level. A probabilistic instantiation of e would require control of the estimated clean reference and normalization, return-variability and runtime terms, dependence within trajectories, and adaptive candidate selection. Additional confirmation episodes can reduce sampling uncertainty under suitable assumptions; candidate exploration determines coverage, and independent performance estimation uses fresh post-selection data.

E Historical Attack Results and Task-Level Details

Scope. This appendix expands the historical results in Table 1 and Figure 3, as discussed in Section 5.2. Atari and DMC WMAttack use Qwen2.5-3B-Instruct; Appendix K separately evaluates alternative backbones. Backends, initialization, and history access are specified in Appendix J. Atari excludes Square Attack because of its higher closed-loop cost; DMC includes all five families. A dash marks a family absent from the corresponding export.

Table 8: Attack-family selection counts for the strongest WMAttack configuration on each task. For Atari, Square Attack is listed for completeness but is excluded from the primary Atari runs due to its closed-loop evaluation cost.

Suite	APGD-CE	APGD-DLR	FAB	Square	PhysCond-WMA
Atari	17	3	2	0	4
DMC	3	1	14	0	2

Table 9: Task-level comparison on DreamerV3 Atari.

Task	Claudini D	Random D	WMAttack D	Claudini U	Random U	WMAttack U
Alien	0.380	0.447	0.671	-0.142	-0.115	0.195
Amidar	0.296	0.229	0.899	-0.382	-0.408	0.366
Assault	0.036	0.090	0.519	-0.587	-0.560	-0.098
Asterix	0.603	0.397	0.775	0.088	-0.105	0.370
BankHeist	0.748	0.797	0.974	0.074	0.121	0.459
BattleZone	0.473	0.371	0.710	-0.146	-0.257	0.106
Boxing	0.630	0.698	0.973	0.021	0.056	0.332
Breakout	0.927	0.906	0.879	0.565	0.538	0.370
ChopperCommand	0.421	0.207	0.842	-0.170	-0.392	0.330
CrazyClimber	0.126	0.302	0.755	-0.690	-0.443	0.032
DemonAttack	0.362	0.256	0.740	-0.188	-0.338	0.002
Freeway	0.000	0.000	0.000	-0.628	-0.614	-0.621
Frostbite	0.246	0.202	0.751	-0.285	-0.305	0.068
Gopher	0.531	0.235	0.846	-0.199	-0.495	0.245
Hero	0.014	0.096	0.643	-0.554	-0.484	0.229
Jamesbond	0.185	0.370	0.840	-0.390	-0.189	0.076
Kangaroo	0.549	0.477	0.892	-0.166	-0.199	0.317
Krull	0.169	0.156	0.611	-0.560	-0.554	-0.090
KungFuMaster	0.440	0.311	0.903	-0.269	-0.355	0.072
MsPacman	0.388	0.439	0.697	-0.211	-0.055	0.193
Pong	1.942	1.935	1.971	1.401	1.403	1.453
PrivateEye	2.460	1.287	7.146	1.571	0.459	5.984
Qbert	0.197	0.771	0.966	-0.526	0.188	0.505
RoadRunner	0.354	0.255	0.887	-0.264	-0.399	0.396
Seaquest	0.220	0.137	0.275	-0.404	-0.418	-0.295
UpNDown	0.217	0.233	0.709	-0.394	-0.326	0.196

F Historical Search Curves and Diagnostics

Scope. We report numerical trajectories and threshold diagnostics for the historical curves in Figure 4, as discussed in Section 5.3. These within-search Utility measurements complement the independently tested, episode-matched comparisons.

Table 10: Task-level comparison on DreamerV3 DMC.

Task	Claudini D	Random D	WMAttack D	Claudini U	Random U	WMAttack U
acrobot swingup	0.472	0.657	0.986	-0.175	-0.003	0.235
ball in cup catch	0.074	0.049	0.865	-0.366	-0.368	0.264
cartpole balance	0.273	0.319	0.510	-0.158	-0.153	0.071
cartpole balance sparse	0.277	0.647	0.953	-0.265	-0.074	0.356
cartpole swingup	0.202	0.273	0.755	-0.282	-0.266	0.180
cartpole swingup sparse	0.756	0.632	0.988	0.220	0.018	0.455
cheetah run	0.620	0.631	0.845	0.225	0.225	0.445
finger spin	0.545	0.425	0.882	-0.128	-0.232	0.096
finger turn easy	0.197	0.177	0.553	-0.166	-0.209	0.110
finger turn hard	0.171	0.209	0.472	-0.359	-0.265	0.021
hopper hop	0.854	0.839	0.950	0.457	0.437	0.590
hopper stand	0.898	0.996	0.993	0.334	0.514	0.468
pendulum swingup	0.000	0.000	0.000	-0.517	-0.547	-0.629
quadruped run	0.346	0.339	0.643	0.012	-0.009	0.307
quadruped walk	0.217	0.386	0.884	-0.141	-0.016	0.184
reacher easy	0.009	0.037	0.139	-0.357	-0.351	-0.285
reacher hard	0.093	0.064	0.860	-0.286	-0.342	0.424
walker run	0.287	0.307	0.772	-0.116	-0.097	0.384
walker stand	0.027	0.022	0.149	-0.338	-0.383	-0.272
walker walk	0.057	0.034	0.447	-0.332	-0.343	0.069

Table 11: Best discovered WMAttack configuration for each DreamerV3 Atari task.

Task	Attack	ϵ	Steps	Alloc.	Drop ↑	Flip ↑	Utility ↑
Alien	APGD-CE	20	24	margin-linear	0.671	0.684	0.195
Amidar	APGD-CE	20	24	fixed	0.899	0.686	0.366
Assault	APGD-CE	8	10	margin-linear	0.519	0.521	-0.098
Asterix	APGD-DLR	8	6	margin-linear	0.775	0.623	0.370
BankHeist	APGD-CE	20	20	margin-linear	0.974	0.743	0.459
BattleZone	APGD-CE	8	10	margin-linear	0.710	0.704	0.106
Boxing	APGD-CE	20	20	fixed	0.973	0.782	0.332
Breakout	APGD-DLR	10	12	margin-linear	0.879	0.476	0.370
ChopperCommand	APGD-CE	20	4	fixed	0.842	0.570	0.330
CrazyClimber	APGD-CE	8	6	margin-linear	0.755	0.706	0.032
DemonAttack	PhysCond-WMA	22	18	margin-linear	0.740	0.815	0.002
Freeway	FAB	10	18	margin-linear	0.000	0.524	-0.621
Frostbite	PhysCond-WMA	22	24	margin-linear	0.751	0.838	0.068
Gopher	APGD-CE	8	6	margin-linear	0.846	0.592	0.245
Hero	APGD-CE	20	24	margin-linear	0.643	0.761	0.229
Jamesbond	PhysCond-WMA	22	18	margin-linear	0.840	0.781	0.076
Kangaroo	APGD-CE	22	22	margin-linear	0.892	0.570	0.317
Krull	APGD-CE	20	24	margin-linear	0.611	0.664	-0.090
KungFuMaster	PhysCond-WMA	26	14	margin-linear	0.903	0.837	0.072
MsPacman	APGD-CE	20	10	margin-linear	0.697	0.634	0.193
Pong	FAB	10	12	margin-linear	1.971	0.322	1.453
PrivateEye	APGD-CE	10	12	margin-linear	7.146	0.702	5.984
Qbert	APGD-DLR	8	6	margin-linear	0.966	0.567	0.505
RoadRunner	APGD-CE	22	22	margin-linear	0.887	0.703	0.396
Seaquest	APGD-CE	10	16	margin-linear	0.275	0.646	-0.295
UpNDown	APGD-CE	8	6	margin-linear	0.709	0.720	0.196

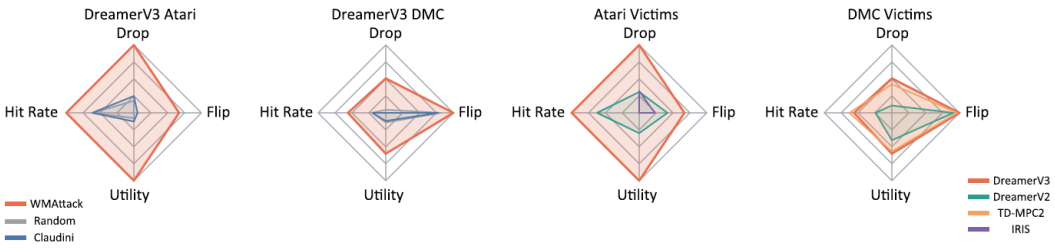


Figure 5: Historical search diagnostics. Axes are min–max normalized separately using values from Tables 25 and 16. Each axis reports a separate metric, and polygon geometry serves visualization only. Hit Rate denotes attainment of a run-relative Utility threshold.

Table 12: Best discovered WMAttack configuration for each DreamerV3 DMC task.

Task	Attack	ϵ	Steps	Alloc.	Drop ↑	Flip ↑	Utility ↑
acrobot swingup	APGD-DLR	10	14	margin-linear	0.986	0.328	0.235
ball in cup catch	FAB	10	18	fixed	0.865	0.657	0.264
cartpole balance	FAB	10	20	fixed	0.510	0.825	0.071
cartpole balance sparse	APGD-CE	10	10	margin-linear	0.953	0.548	0.356
cartpole swingup	FAB	10	20	fixed	0.755	0.718	0.180
cartpole swingup sparse	APGD-CE	10	12	margin-linear	0.988	0.704	0.455
cheetah run	FAB	10	16	fixed	0.845	0.960	0.445
finger spin	PhysCond-WMA	10	10	margin-linear	0.882	0.511	0.096
finger turn easy	FAB	10	18	fixed	0.553	0.937	0.110
finger turn hard	FAB	10	20	fixed	0.472	0.932	0.021
hopper hop	FAB	10	18	fixed	0.950	0.950	0.590
hopper stand	FAB	10	18	margin-linear	0.993	0.830	0.468
pendulum swingup	APGD-CE	10	10	margin-linear	0.000	0.457	-0.629
quadruped run	FAB	10	14	fixed	0.643	0.994	0.307
quadruped walk	PhysCond-WMA	10	18	margin-linear	0.884	1.000	0.184
reacher easy	FAB	10	20	fixed	0.139	0.943	-0.285
reacher hard	FAB	10	16	fixed	0.860	0.904	0.424
walker run	FAB	10	16	fixed	0.772	0.942	0.384
walker stand	FAB	10	20	fixed	0.149	0.919	-0.272
walker walk	FAB	10	14	fixed	0.447	0.949	0.069

Table 13: DreamerV2 Atari task-level results.

Task	Attack	ϵ	Steps	Alloc.	Drop ↑	Flip ↑	Utility ↑
Alien	APGD-CE	10	14	margin-linear	0.312	0.650	-0.185
Amidar	APGD-CE	10	14	margin-linear	0.241	0.766	-0.222
Assault	APGD-CE	10	10	margin-linear	0.253	0.386	-0.386
Asterix	APGD-CE	10	14	margin-linear	0.452	0.675	0.118
BankHeist	APGD-CE	10	8	margin-linear	0.950	0.580	0.365

Table 14: Per-task, per-attack-family results on DreamerV3 Atari. Each cell reports Drop / Utility for the best confirmed configuration within an attack family.

Task	APGD-CE	APGD-DLR	FAB	PhysCond-WMA
Alien	0.671 / 0.195	0.611 / 0.048	0.170 / -0.344	0.722 / -0.101
Amidar	0.899 / 0.366	0.722 / 0.094	0.231 / -0.366	0.962 / 0.177
Assault	0.519 / -0.098	0.525 / -0.150	0.213 / -0.388	0.501 / -0.267
Asterix	0.744 / 0.354	0.775 / 0.370	0.214 / -0.257	0.775 / 0.164
BankHeist	0.974 / 0.459	0.973 / 0.423	0.088 / -0.600	0.923 / 0.250
BattleZone	0.710 / 0.106	0.579 / -0.075	0.526 / -0.097	0.803 / -0.052
Boxing	0.973 / 0.332	0.929 / 0.249	0.111 / -0.384	1.002 / 0.044
Breakout	0.739 / 0.083	0.879 / 0.370	0.364 / -0.167	0.933 / 0.300
ChopperCommand	0.842 / 0.330	0.513 / -0.150	0.302 / -0.260	0.736 / -0.004
CrazyClimber	0.755 / 0.032	0.807 / 0.017	0.093 / -0.719	0.773 / -0.078
DemonAttack	0.633 / -0.041	0.391 / -0.255	0.453 / -0.147	0.740 / 0.002
Freeway	0.000 / -0.648	0.000 / -0.695	0.000 / -0.621	0.000 / -0.899
Frostbite	0.190 / -0.410	0.413 / -0.123	0.128 / -0.390	0.751 / 0.068
Gopher	0.846 / 0.245	0.789 / 0.093	0.410 / -0.270	0.912 / 0.138
Hero	0.643 / 0.229	0.248 / -0.330	0.011 / -0.522	0.772 / 0.157
Jamesbond	0.341 / -0.444	0.656 / -0.021	0.184 / -0.392	0.840 / 0.076
Kangaroo	0.892 / 0.317	0.883 / 0.262	0.468 / -0.148	0.865 / 0.136
Krull	0.611 / -0.090	0.308 / -0.434	0.165 / -0.490	0.449 / -0.372
KungFuMaster	0.596 / -0.071	0.493 / -0.196	-0.113 / -0.806	0.903 / 0.072
MsPacman	0.697 / 0.193	0.730 / 0.190	0.276 / -0.295	0.837 / 0.170
Pong	1.854 / 1.128	1.961 / 1.245	1.971 / 1.453	1.942 / 1.126
PrivateEye	7.146 / 5.984	1.423 / 0.613	-3.286 / -3.997	2.676 / 1.710
Qbert	0.854 / 0.176	0.966 / 0.505	-0.026 / -0.637	0.971 / 0.402
RoadRunner	0.887 / 0.396	0.550 / -0.106	0.275 / -0.384	0.607 / -0.104
Seaquest	0.275 / -0.295	0.241 / -0.370	0.023 / -0.550	0.332 / -0.515
UpNDown	0.709 / 0.196	0.676 / 0.129	0.233 / -0.285	0.890 / 0.084

Table 15: Per-task, per-attack-family results on DreamerV3 DMC. Each cell reports Drop / Utility for the best confirmed configuration within an attack family.

Task	APGD-CE	APGD-DLR	FAB	Square	PhysCond-WMA
acrobot swingup	0.997 / 0.180	0.986 / 0.235	0.991 / 0.178	0.201 / -0.655	0.872 / -0.134
ball in cup catch	0.235 / -0.280	0.411 / -0.132	0.865 / 0.264	0.115 / -0.447	0.817 / 0.010
cartpole balance	0.385 / -0.124	0.392 / -0.071	0.510 / 0.071	0.001 / -0.915	0.411 / -0.305
cartpole balance sparse	0.953 / 0.356	0.839 / 0.277	0.958 / 0.342	0.002 / -0.755	0.922 / 0.093
cartpole swingup	0.640 / 0.089	0.628 / 0.100	0.755 / 0.180	0.068 / -0.649	0.659 / -0.109
cartpole swingup sparse	0.988 / 0.455	0.974 / 0.389	0.994 / 0.362	0.028 / -0.643	0.960 / 0.159
cheetah run	0.535 / 0.083	0.575 / 0.141	0.845 / 0.445	0.386 / -0.152	0.843 / 0.147
finger spin	0.420 / -0.218	0.505 / -0.159	0.614 / -0.122	0.730 / -0.056	0.882 / 0.096
finger turn easy	0.233 / -0.176	0.276 / -0.124	0.553 / 0.110	0.178 / -0.279	0.339 / -0.372
finger turn hard	0.111 / -0.439	0.294 / -0.180	0.472 / 0.021	-0.124 / -0.771	0.343 / -0.385
hopper hop	0.799 / 0.342	0.831 / 0.373	0.950 / 0.590	0.224 / -0.491	0.971 / 0.289
hopper stand	0.948 / 0.150	0.921 / 0.407	0.993 / 0.468	-0.157 / -0.668	0.993 / 0.095
pendulum swingup	0.000 / -0.629	0.000 / -0.654	0.000 / -0.657	0.000 / -0.815	0.000 / -0.839
quadruped run	-0.022 / -0.380	-0.058 / -0.397	0.643 / 0.307	-0.172 / -0.550	0.925 / 0.228
quadruped walk	0.223 / -0.152	0.177 / -0.195	0.494 / 0.160	0.170 / -0.214	0.884 / 0.184
reacher easy	0.034 / -0.391	0.111 / -0.308	0.139 / -0.285	0.036 / -0.407	0.236 / -0.518
reacher hard	0.325 / -0.113	0.317 / -0.093	0.860 / 0.424	0.093 / -0.350	0.616 / -0.098
walker run	0.556 / 0.136	0.549 / 0.139	0.772 / 0.384	0.018 / -0.623	0.762 / 0.044
walker stand	0.068 / -0.394	-0.032 / -0.409	0.149 / -0.272	-0.043 / -0.506	0.192 / -0.532
walker walk	0.162 / -0.241	0.269 / -0.145	0.447 / 0.069	-0.013 / -0.475	0.321 / -0.385

Table 16: Threshold-based search efficiency. Hit Rate reports the fraction of task–attack pairs that reach 90% of their final best utility.

Victim	Suite	Method	Pairs	Hit Rate ↑	Trials ↓	Time (s)
DreamerV3	Atari	Claudini	104	0.538	3.36	349
DreamerV3	Atari	Random	104	0.529	3.67	394
DreamerV3	Atari	WMAttack	104	0.875	3.45	491
DreamerV3	DMC	Claudini	100	0.170	3.53	254
DreamerV3	DMC	Random	100	0.180	3.44	325
DreamerV3	DMC	WMAttack	100	0.490	3.71	823
DreamerV2	Atari	WMAttack	40	0.550	2.47	111
DreamerV2	DMC	WMAttack	50	0.220	2.69	232
TD-MPC2	DMC	WMAttack	50	0.500	2.27	339
IRIS	Atari	WMAttack	40	0.000	–	–

Threshold-based diagnostic. Table 16 reports the threshold-based statistics. Each run uses a fraction of its own final Utility as the threshold. In particular, with negative final Utility, $0.9U_{\text{final}}$ is greater than U_{final}, so a zero hit rate can arise from this threshold definition. These statistics describe progress toward run-relative Utility thresholds. The independent-test, episode-budget analysis is reported in Section 5.3.

Curve summaries. For each suite, family, and method, we report task-mean running-best Utility after 1, 2, 4, and 8 evaluation records in logged order. Each scout or confirmation contributes one record, and both stages enter the running maximum. Figure 4 additionally shows standard errors across tasks. Stage-specific episode counts, runtime, and dispersion terms follow Appendix C; these summaries describe the recorded search trajectory.

Table 17: Historical Atari search trajectories. Entries are task-mean running-best Utility over scout and confirmation records.

Attack	Method	Record 1	Record 2	Record 4	Record 8
APGD-CE	Claudini	-0.175	-0.102	0.032	0.193
APGD-CE	Random	-0.161	-0.080	0.124	0.202
APGD-CE	WMAttack	**0.183**	**0.522**	**0.582**	**0.651**
APGD-DLR	Claudini	-0.158	-0.067	0.010	0.095
APGD-DLR	Random	-0.155	-0.114	-0.068	0.029
APGD-DLR	WMAttack	**0.269**	**0.286**	**0.409**	**0.516**
FAB	Claudini	-0.609	-0.401	-0.258	0.006
FAB	Random	-0.476	-0.438	**-0.111**	**0.069**
FAB	WMAttack	**-0.355**	**-0.332**	-0.236	-0.070
PhysCond-WMA	Claudini	-0.224	-0.182	0.171	0.259
PhysCond-WMA	Random	-0.260	-0.200	-0.099	0.010
PhysCond-WMA	WMAttack	**0.796**	**0.801**	**0.925**	**1.015**

Table 18: Historical DMC search trajectories. Entries are task-mean running-best Utility over scout and confirmation records.

Attack	Method	Record 1	Record 2	Record 4	Record 8
APGD-CE	Claudini	-0.303	-0.241	-0.196	-0.158
APGD-CE	Random	-0.273	-0.238	-0.175	-0.131
APGD-CE	WMAttack	**-0.039**	**-0.011**	**0.024**	**0.037**
APGD-DLR	Claudini	-0.365	-0.282	-0.196	-0.168
APGD-DLR	Random	-0.226	-0.206	-0.170	-0.151
APGD-DLR	WMAttack	**-0.062**	**-0.045**	**0.005**	**0.065**
Square	Claudini	-0.545	-0.476	-0.448	-0.400
Square	Random	**-0.526**	**-0.472**	**-0.437**	**-0.375**
Square	WMAttack	-0.621	-0.577	-0.515	-0.440
PhysCond-WMA	Claudini	-0.410	-0.377	-0.326	-0.275
PhysCond-WMA	Random	-0.384	-0.338	-0.320	-0.297
PhysCond-WMA	WMAttack	**-0.106**	**-0.094**	**-0.069**	**-0.016**

Across both suites, WMAttack has higher stage-mixed running-best Utility for APGD-CE, APGD-DLR, and PhysCond-WMA. The baselines are more competitive for Atari FAB and DMC Square. These historical trajectories characterize recorded search progress; the controlled comparisons in Section 5.4 examine the contributions of retrieval and feedback.

G Ablation Details

Complete module comparison. Table 3 reports all initial and final Utility, Drop, and Flip values for the four variants. Initial and final values refer to separate evaluation stages; Full has the highest final Drop and Utility, but not the highest Flip. SCAS improves final outcomes when RGAR is present, while SCAS alone underperforms the variant without either module.

G.1 Component-Level Controlled Ablation of SCAS

Purpose and variants. We separate proposer and Auditor contributions on the same 46 DreamerV3 tasks as RGAR (26 Atari, 20 DMC), using APGD-CE, APGD-DLR, FAB, and PhysCond-WMA. The four arms are A: proposer + Auditor; B: programmatic selector + Auditor; C: proposer only; D: programmatic selector only. The selector takes the top three ranked candidates. All arms disable RGAR and retain numerical history; C/D remove only Auditor-generated analysis and reflection.

Matched setup. All arms share construction, validity, deduplication, and ranking rules and the 32-configuration, 10-pixel menu and evaluation allocation in Appendix J.2. Confirmation Utility selects the frozen configuration before independent testing. The 46 tasks, four families, three repetitions, and four arms yield 552 paired blocks and 2,208 units, executed on the same GPU in cyclically balanced order. Within each block, arms share checkpoints and evaluation seeds. Candidate lists may subsequently diverge through selection and Auditor feedback, while the programmatic search framework and historical access remain controlled.

Evaluation and statistics. Final attack effectiveness is measured by normalized return degradation on the independent test episodes:

$$D = \frac{\bar{R}_{\text{clean}} - \bar{R}_{\text{attack}}}{|\bar{R}_{\text{clean}}| + 1},$$

where larger values indicate stronger attacks. A–B is the primary contrast; A–C, A–D, and B–D are exploratory. We condition on task–family strata and resample repetitions and paired test episodes using shared indices and equal stratum/repetition weights. Each of 5,000 resamples (seed 20260922) recomputes D and its differences; percentiles give 95% paired CIs.

Table 19: Controlled component comparisons of SCAS across 46 tasks: 552 complete four-variant paired blocks and 2,208 search units. Positive differences indicate greater independent-test return degradation; intervals are 95% paired bootstrap confidence intervals.

Comparison	Component increment	Mean ΔD	95% paired CI
A–B	LLM proposer versus programmatic selection, with Auditor retained	$+0.018$	$[+0.007, +0.028]$
A–C	Auditor increment with proposer retained	$+0.011$	$[+0.003, +0.020]$
A–D	Full SCAS versus program-only search	$+0.023$	$[+0.011, +0.035]$
B–D	Auditor increment under programmatic selection	$+0.005$	$[-0.007, +0.014]$

LLM calls per search unit: A = 15, B = 12, C = 3, and D = 0; all four variants use 18 online attack episodes.

Results. Full SCAS has the highest mean independent-test attack effectiveness, as shown in Table 19. The primary contrast A–B supports the LLM proposer over programmatic selection with the Auditor retained ($\Delta D = +0.018$, 95% CI $[+0.007, +0.028]$). The exploratory contrasts also favor retaining the Auditor with the proposer (A–C: $+0.011$, $[+0.003, +0.020]$) and full SCAS over program-only search (A–D: $+0.023$, $[+0.011, +0.035]$). In contrast, the Auditor increment under programmatic selection remains statistically unresolved (B–D: $+0.005$, $[-0.007, +0.014]$). Because the variants share candidate-construction and ranking rules, the A–D contrast supports an additional benefit from LLM-based feedback interpretation and candidate selection within this programmatic search framework. The exploratory intervals are unadjusted for multiplicity. The historical SCAS $\times$ RGAR comparison in Table 3 separately examines retrieval conditions, while Table 2 evaluates full-pipeline search gains.

G.2 Controlled RGAR Retrieval Protocol

Purpose and coverage. Table 4(a) compares No history, Uniform history, Behavior history, and RGAR on 26 Atari and 20 DMC tasks, four families (APGD-CE, APGD-DLR, FAB, PhysCond-WMA), and search seeds 0, 1, and 2. All planned units completed: 552 paired four-arm blocks (312 Atari, 240 DMC), totaling 2,208 units.

Shared evaluation. All arms retain SCAS and share checkpoints, menus, online budgets, and the independent-test protocol in Appendix J.2. They differ only in historical-information selection and provision; held-out returns never enter retrieval or search.

History access. The pool is frozen before evaluation. History-enabled arms share eligible same-family records: Uniform samples with a prespecified seed; Behavior retrieves by policy-output and clean-rollout statistics; RGAR adds encoder, recurrent-state, latent-state, and actor-hidden features. Retrieval and LLM historical context are limited to six records. No history receives neither configurations nor historical context.

Ranking. Behavior and RGAR share $s_i = 6[\cos(\bar{\psi}_\tau, \bar{\psi}_i) + 1] + 0.5 \tanh(U_i)$, using original historical utilities rather than target-test scores. Uniform ignores similarity. Unlike the general method's positive-score screening, this comparison retains the common eligible pool without an extra score filter; fixed record identifiers break ties.

Queries and projections. Each task–seed pair shares five clean query episodes across families and arms, with acquisition cost counted separately. Feature summaries use the first 32 sampled decision points of the prespecified first episode, without return-based trajectory selection. Each feature vector contributes its mean, standard deviation, mean absolute value, and dimension-normalized norm $\|x\|_2/\sqrt{\dim(x)}$; the mean and standard deviation of each quantity across sampled points yield eight statistics per group. Behavior concatenates the policy-output (`logits`) block with three shared statistics: mean clean policy margin, mean return, and return standard deviation, giving $8 + 3 = 11$ dimensions. RGAR concatenates encoder, recurrent, stochastic, actor-hidden, and policy-output blocks in that order, followed by the same three statistics, giving $5 \times 8 + 3 = 43$ dimensions. The shared statistics use all five episodes. For each descriptor type, query and eligible-history descriptors are jointly centered and projected by a separately fitted PCA with at most 16 components, without scaling, whitening, or imputation. A fixed mapping sends retrieved configurations to the common menu while retaining the target's 10-pixel radius.

Statistics. For each suite, we average paired RGAR-minus-comparator differences equally over families within each task–seed pair, then resample tasks and seeds within tasks with replacement. Equal-weight means from 5,000 paired hierarchical resamples (seed 831) give percentile 95% CIs without multiplicity correction. Positive differences favor RGAR.

Aggregation and paired comparisons. Atari and DMC are summarized separately, with equal weights for tasks, attack families, and repetitions. Paired differences are defined as RGAR Drop minus the corresponding comparator's Drop; positive values indicate greater return degradation under RGAR. Means and paired differences use the same data, metric definition, and aggregation weights. All six unadjusted 95% paired confidence intervals in Table 20 lie above zero, supporting a suite-level mean advantage of RGAR over each comparator on both Atari and DMC.

Table 20: Mean paired Drop differences (ΔD, RGAR minus comparator) and unadjusted 95% paired bootstrap confidence intervals for the controlled retrieval comparison in Table 4. Positive values indicate stronger attacks under RGAR.

	Atari		DMC	
Comparison	Mean ΔD	95% paired CI	Mean ΔD	95% paired CI
RGAR − No history	$+0.056$	$[+0.029, +0.086]$	$+0.092$	$[+0.051, +0.128]$
RGAR − Uniform history	$+0.057$	$[+0.031, +0.085]$	$+0.081$	$[+0.039, +0.124]$
RGAR − Behavior history	$+0.034$	$[+0.011, +0.059]$	$+0.047$	$[+0.018, +0.074]$

History scope and interpretation. Record-level target-task/checkpoint exclusion and target-independent generation of historical records are distinct controls. The latter is unverified, so the evidence concerns retrieval within the evaluated frozen pool. RGAR jointly changes internal feature groups and the fitted projection relative to Behavior history. Joint centering and PCA provide a common numerical representation; semantic correspondence across independently trained models is not assumed. Retrieval guides initialization and subsequent candidate generation and ranking. Table 20 reports the suite-level mean advantages under this 46-task protocol.

Independent four-task supplementary comparison. A separate experiment on Alien, Pong, acrobot swingup, and cheetah run uses a frozen pool of 168 historical records with direct exclusion

of all four target tasks and checkpoints. It contains 48 complete four-arm blocks (192 method-level units). RGAR-minus-comparator mean differences are 0.0192, 0.0063, and 0.0002 for no history, uniform history, and behavior-based retrieval, with 95% paired intervals $[-0.0004, 0.0407]$, $[-0.0104, 0.0239]$, and $[-0.0160, 0.0155]$. The positive mean gains, largest against no history, are not statistically resolved in this fixed four-task setting. As shown in Table 21, the four-task and 46-task experiments are analyzed separately, using their own cohorts, histories, execution records, and inferential scopes.

Table 21: Protocols for two separately analyzed RGAR cohorts. Both evaluate composite descriptors with fitted projections.

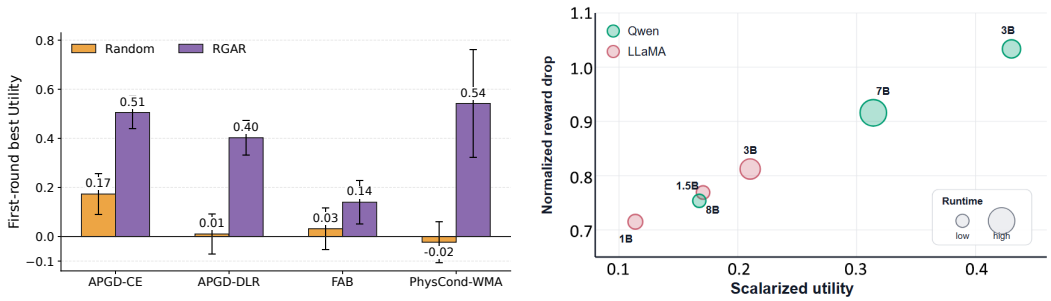
Item	46-task comparison	Independent four-task comparison
Coverage	26 Atari + 20 DMC; four families; three search seeds	Alien, Pong, acrobot swingup, cheetah run; four families; three repetitions
Paired units	552 four-arm blocks; 2,208 method-level units	48 four-arm blocks; 192 method-level units
History	Frozen eligible same-family pool; no evaluation-time updates	Frozen 168-record pool; all four target tasks/checkpoints directly excluded; no updates
History scope	Conditional on the evaluated pool; provenance as described in the text	Conditional on the evaluated pool; provenance as described in the text
Descriptors	11-dimensional behavior or 43-dimensional composite features; separate jointly centered PCA, at most 16 components	Same descriptor dimensions and centering/PCA rule, fitted to this experiment's query and eligible pool
Query and context	Five clean episodes per task–seed; first 32 sampled decisions of the first episode for feature summaries; at most six history records	Same query rule; 12 shared queries (60 physical episodes); at most six history records
Search and testing	32 configurations per family, 10-pixel radius; 18 online attack episodes; frozen winner tested on ten attack and ten paired clean episodes	Same menu, radius, and per-unit episode allocation; query acquisition shared across arms and families
Inference	Separate Atari/DMC task-and-seed paired bootstrap; 5,000 resamples, seed 831	Fixed task–family strata; paired repetition-and-episode bootstrap; 5,000 resamples, seed 20260915

G.3 Additional Retrieval Analyses

Scope. The supplementary RGAR analysis for Section 5.4 covers 26 DreamerV3 Atari tasks and four families (104 task–attack pairs). Variants share SCAS but differ in retrieval access during initialization and later search. Pairs are descriptive within-task units.

Table 22: Per-attack RGAR ablation on DreamerV3 Atari. Each cell reports Utility / Drop / Flip averaged across 26 tasks.

Attack	Variant	First Utility / Drop / Flip ↑	Final Utility / Drop / Flip ↑
APGD-CE	Random init.	0.173 / 0.522 / 0.443	-0.243 / 0.395 / 0.439
APGD-CE	RGAR init.	**0.505 / 0.813 / 0.658**	**0.206 / 0.795 / 0.650**
APGD-DLR	Random init.	0.010 / 0.412 / 0.396	-0.354 / 0.322 / 0.393
APGD-DLR	RGAR init.	**0.402 / 0.769 / 0.614**	**0.086 / 0.714 / 0.606**
FAB	Random init.	0.032 / 0.363 / 0.227	-0.315 / 0.288 / 0.237
FAB	RGAR init.	**0.140 / 0.483 / 0.255**	**-0.284 / 0.331 / 0.251**
PhysCond-WMA	Random init.	-0.023 / 0.483 / 0.509	-0.389 / 0.393 / 0.514
PhysCond-WMA	RGAR init.	**0.542 / 1.017 / 0.767**	**0.074 / 0.819 / 0.764**



(a) RGAR warm-start effect

(b) LLM backbone sensitivity

Figure 6: Design analysis of WMAttack on DreamerV3 Atari. Panel (a) reports mean first-round best Utility across 26 Atari tasks for each attack family; error bars show ± 1 standard error of the mean, computed as the sample standard deviation across tasks divided by $\sqrt{26}$. Panel (b) reports joint proposer–Auditor backbone sensitivity.

Table 23: Pair-level win counts for RGAR over random initialization on DreamerV3 Atari. Each entry reports the number of task–attack pairs where RGAR is strictly better than random initialization.

Metric	First round	Final result	Total pairs
Utility	85	87	104
Reward drop	84	85	104
Flip rate	96	92	104

Retrieval-enabled runs attain higher recorded initial and final metrics on most task–attack pairs, with larger differences for APGD-CE, APGD-DLR, and the two-stage adaptation than for FAB. Figure 6(a) shows the corresponding first-round Utility differences. These descriptive comparisons encompass historical-information access and retrieval-assisted initialization and subsequent search.

H Additional Main-Result Metrics

Scope and metrics. Table 25 expands Section 5.2 with aggregate Flip, Utility, and evaluation time for DreamerV3 and additional architectures, each using its own task subset. Runtime and variability enter Utility, so its ordering can differ from attacked return.

I Clean and Attacked Returns

Evaluation setup. Tables 26 and 27 give clean/attacked returns in task-specific units for all 46 DreamerV3 main tasks. Each mean averages five confirmation-stage episodes for the configuration selected by Utility, as shown in Appendix E. Methods retain their separately measured clean baselines. Freeway and pendulum swingup have zero measured clean and attacked returns, yielding zero measured performance degradation.

Table 24: Per-attack final pair-level win counts for RGAR over random initialization. Each entry reports the number of tasks where RGAR achieves a higher final metric than random initialization.

Attack	Utility	Drop	Flip
APGD-CE	25 / 26	25 / 26	25 / 26
APGD-DLR	25 / 26	25 / 26	25 / 26
FAB	12 / 26	10 / 26	16 / 26
PhysCond-WMA	25 / 26	25 / 26	26 / 26

Table 25: Main adversarial evaluation results.

Victim	Suite	Tasks	Attacks	Method	Drop ↑	Flip ↑	Utility ↑	Time (min)
DreamerV3	Atari	26	4	Claudini	0.497	0.400	-0.132	77.2
DreamerV3	Atari	26	4	Random	0.446	0.377	-0.163	73.4
DreamerV3	Atari	26	4	WMAttack	**1.034**	**0.660**	**0.430**	106.2
DreamerV3	DMC	20	5	Claudini	0.319	0.705	-0.137	55.4
DreamerV3	DMC	20	5	Random	0.352	0.721	-0.123	59.0
DreamerV3	DMC	20	5	WMAttack	**0.682**	**0.800**	**0.174**	120.2
DreamerV2	Atari	10	4	WMAttack	0.539	0.554	-0.021	242.9
DreamerV2	DMC	10	5	WMAttack	0.397	0.762	0.046	253.5
TD-MPC2	DMC	10	5	WMAttack	0.621	0.786	0.152	68.1
IRIS	Atari	10	4	WMAttack	0.550	0.474	-0.214	830.1

Table 26: DreamerV3 Atari: mean clean and attacked episodic return for the Utility-selected configurations.

	WMAttack		Random		Claudini-style	
Task	Clean	Attacked	Clean	Attacked	Clean	Attacked
Alien	974.00	320.00	1198.00	662.00	1084.00	672.00
Amidar	194.00	18.60	189.40	145.80	198.00	139.00
Assault	699.20	336.00	720.00	655.20	701.20	676.00
Asterix	980.00	220.00	880.00	530.00	1010.00	400.00
BankHeist	1130.00	28.00	1090.00	220.00	1108.00	278.00
BattleZone	15200.00	4400.00	12400.00	7800.00	14800.00	7800.00
Boxing	94.80	1.60	93.60	27.60	80.60	29.20
Breakout	32.00	3.00	33.00	2.20	34.60	1.60
ChopperCommand	1520.00	240.00	1160.00	920.00	1660.00	960.00
CrazyClimber	92040.00	22540.00	79280.00	55340.00	64320.00	56240.00
DemonAttack	326.00	84.00	206.00	153.00	264.00	168.00
Freeway	0.00	0.00	0.00	0.00	0.00	0.00
Frostbite	420.00	104.00	396.00	316.00	430.00	324.00
Gopher	4040.00	620.00	2140.00	1636.00	3912.00	1836.00
Hero	13531.00	4827.00	13517.00	12214.00	13527.00	13331.00
Jamesbond	380.00	60.00	350.00	220.00	270.00	220.00
Kangaroo	4440.00	480.00	2680.00	1400.00	3280.00	1480.00
Krull	7534.00	2932.00	7358.00	6210.00	6432.00	5348.00
KungFuMaster	15280.00	1480.00	18140.00	12500.00	21920.00	12280.00
MsPacman	1456.00	440.00	1152.00	646.00	1200.00	734.00
Pong	19.60	-21.00	20.40	-21.00	19.80	-20.60
PrivateEye	41.60	-262.80	182.00	-53.60	183.60	-270.60
Qbert	2265.00	75.00	2095.00	480.00	635.00	510.00
RoadRunner	24840.00	2800.00	22900.00	17060.00	24100.00	15560.00
Seaquest	348.00	252.00	320.00	276.00	344.00	268.00
UpNDown	5736.00	1668.00	5284.00	4054.00	5320.00	4166.00

Table 27: DreamerV3 DMC: mean clean and attacked episodic return for the Utility-selected configurations.

	WMAttack		Random		Claudini-style	
Task	Clean	Attacked	Clean	Attacked	Clean	Attacked
acrobot swingup	339.16	3.64	297.92	101.64	320.15	168.58
ball in cup catch	968.40	129.60	968.80	920.80	969.00	897.60
cartpole balance	993.79	486.20	994.57	676.97	995.05	722.78
cartpole balance sparse	1000.00	46.00	1000.00	352.80	1000.00	722.40
cartpole swingup	860.52	210.15	861.19	626.22	860.91	686.81
cartpole swingup sparse	787.80	8.40	790.80	290.60	786.40	191.40
cheetah run	837.88	128.83	872.59	321.49	807.11	305.89
finger spin	128.00	14.20	100.20	57.20	79.00	35.40
finger turn easy	924.20	412.80	863.80	710.40	859.40	689.80
finger turn hard	368.20	193.80	403.40	319.00	363.00	300.60
hopper hop	280.77	12.96	286.15	45.12	299.54	42.88
hopper stand	694.65	3.98	715.78	2.19	733.20	73.60
pendulum swingup	0.00	0.00	0.00	0.00	0.00	0.00
quadruped run	293.44	104.24	348.59	229.96	363.61	237.35
quadruped walk	267.78	30.22	260.16	159.31	203.55	159.18
reacher easy	894.40	770.00	909.80	876.20	901.40	893.40
reacher hard	634.20	87.80	639.80	598.60	652.00	591.00
walker run	725.89	164.79	732.94	507.46	735.20	523.80
walker stand	929.54	790.54	967.74	946.18	965.20	939.53
walker walk	925.14	511.33	933.31	901.97	949.60	895.78

Relative return reduction and sensitivity analysis. For tasks with positive clean returns, we compute each method's task-level relative return reduction against its own clean baseline:

$$q_{m,t} = \frac{\bar{R}^{\text{clean}}_{m,t} - \bar{R}^{\text{attack}}_{m,t}}{\bar{R}^{\text{clean}}_{m,t}} \times 100\%.$$

The summaries use the task-level mean returns reported in this appendix and weight the 44 positive-clean-return tasks equally. Freeway and pendulum swingup have zero clean returns and are therefore excluded from relative-reduction summaries, but remain in the complete 46-task raw-return tables.

Table 28: Descriptive robustness summaries of historical relative return reduction, computed from the task-level mean returns in this appendix. All entries are percentages; the exclusion analysis is supplementary.

Summary	Tasks	WMAttack	Random	Claudini-style
Mean	44	92.91	42.75	44.23
Median	44	84.21	31.51	32.26
Mean excluding PrivateEye	43	78.05	40.73	39.51

PrivateEye combines a small positive clean return with a negative attacked return, producing a large relative reduction that raises the cross-task mean. WMAttack retains the highest median relative reduction and the highest mean relative reduction after excluding PrivateEye.

Task-wise comparison criteria. The count of 40 out of 46 tasks compares raw attacked returns: WMAttack has a lower mean attacked return than both baselines. The counts of 24 out of 26 Atari tasks and 18 out of 20 DMC tasks instead compare Drop normalized against each method's own clean return. Different clean baselines can yield different task-wise rankings under these two metrics.

J Equal-Budget Search Details

J.1 Historical Evaluation Conditions

Runs and backends. The table below records the executed proposer/Auditor, initialization, and historical access for WMAttack, Random, and Claudini-style. Representation retrieval denotes RGAR; all runs retain numerical feedback and reflection.

Suite	Run label	Proposer/Auditor backend	Initialization	Retrieval
Atari	WMAttack	Qwen2.5-3B-Instruct	Task-conditioned	Representation
Atari	Random	Heuristic	Random	None
Atari	Claudini-style	Heuristic	Random	None
DMC	WMAttack	Qwen2.5-3B-Instruct	Task-conditioned	Representation
DMC	Random	Heuristic	Random	None
DMC	Claudini-style	Heuristic	Random	None

WMAttack uses Qwen2.5-3B-Instruct (Team, 2024) for its proposer and Auditor in both suites; the Atari implementation uses Transformers. Historical Random uses random initialization followed by feedback-conditioned construction and ranking, unlike the uniform selector in Table 2. Claudini-style uses a local autoresearch-inspired heuristic configuration (Panfilov et al., 2026).

Selection and evaluation. The nominal historical schedule has four rounds, each proposing one configuration for scout evaluation and confirmation: four configurations and eight evaluation records. Each round uses one/three scout episodes on Atari/DMC and five confirmation episodes, giving nominal budgets of 24/32 attack episodes per family; clean evaluation is separate. Some archived task–family runs contain additional records. Figure 4 and Tables 17–18 summarize the first eight records in logged order, counting scout and confirmation separately. The highest confirmation Utility selects the final configuration, with ties retaining candidate order. Reported returns are confirmation measurements under each method's initialization and history-access settings.

J.2 Controlled Search Protocol and Baseline Implementations

Purpose and coverage. Table 2 compares Coarse Grid, Random Search, GP-UCB-style, Successive Halving, and WMAttack on 26 Atari and 20 DMC DreamerV3 tasks, using APGD-CE, APGD-DLR, FAB, PhysCond-WMA, and search seeds 0, 1, and 2. Methods share checkpoints and evaluation seeds within each task–family–seed block. Panel (a) allows history only for full WMAttack; panel (b) disables history for all methods, including WMAttack without RGAR.

Shared evaluation protocol. Each task–family–seed unit uses five clean-search episodes and 18 online attack episodes: nine one-episode scouts and three three-episode confirmations. The highest confirmation-Utility configuration is frozen before ten held-out attack episodes and ten paired clean episodes, using seeds disjoint from search. Held-out outcomes never influence configuration selection, tuning, or feedback. Families are searched and tested separately, then averaged without test-based family selection. Clean and test costs are outside the online budget. This shared scout–confirm allocation fixes stage budgets so the comparison measures candidate selection and feedback use.

Shared candidate menu. All methods use 32 configurations per family at radius 10 pixel units, one restart, and $\rho = 0.75$. APGD-CE/DLR steps are $\{4, 6, 8, 10, 12, 16, 20, 24\}$; FAB/PhysCond-WMA steps are $\{6, 8, 10, 12, 16, 20, 24, 32\}$. Each combines with fixed allocation or adaptive minimum-step fractions 0.25, 0.5, and 0.75. Feedback-guided proposals remain within this menu.

Search schedule. Coarse Grid, Random Search, GP-UCB-style, and WMAttack run three batches of three new candidates, confirming each batch's highest-Utility candidate. Successive Halving instead screens nine candidates together and confirms the top three: a single two-rung bracket. Both schedules use the same nine scout and nine confirm episodes; confirmations allocate additional episodes to selected scout candidates.

Random Search and Coarse Grid. Random Search uniformly shuffles unseen configurations with the assigned seed, independently of evaluation scores; Utility still governs promotion and final selection. Coarse Grid targets the lowest, middle-index, and highest radii/steps, retaining all values when at most three exist. It enumerates modes, radii, and steps in fixed order and maps targets lexicographically by radius distance, step distance, mode mismatch, and configuration key. After deduplication, remaining entries are appended by key and prior proposals excluded. At fixed radius, this deterministically covers steps and modes without evaluating all 32 configurations.

GP-UCB-style. We use a fixed-kernel batch implementation with six min–max-normalized features: radius, steps, restarts, ρ, allocation mode (fixed/adaptive: 0/1), and minimum-step fraction (0 for fixed allocation). Scaling uses observed and available configurations; constant features are zero. Each configuration contributes confirmation Utility when available and scout Utility otherwise. Targets are

standardized, using scale 1 when their standard deviation is at most 10^{-8}. A unit-amplitude RBF kernel uses length scale 0.35, diagonal regularization 10^{-5}, and acquisition

$$\mathrm{UCB}(c) = \mu(c) + 1.5\,\sigma(c),$$

with posterior variance floored at 10^{-8} and parameters fixed independently of test outcomes. Fewer than four observed configurations trigger seeded random selection: the first six candidates are random, and the final batch is GP-guided using two confirmation and four scout targets. Each batch uses one posterior ranking without simulated observations; ties retain candidate order, and linear-algebra failures invoke random fallback. A single GP fits scout and confirmation Utilities with the common standardization above and no stage-specific calibration, using the stage definitions in Appendix C.

Ties and failure handling. The sequential controller uses stable Utility ordering; the controlled cold-start controller breaks promotion/selection ties by configuration key. Its strict interface records and stops on invalid or duplicate proposals, budget violations, or evaluation/structured-output failures, without heuristic substitution or automatic reruns. GP numerical fallback follows the separate rule above.

Selection and interpretation. All methods use the Utility weights in Appendix C. WMAttack retains within-task feedback under cold start, which disables historical reads and writes. The cold-start panel evaluates the remaining pipeline, while Table 4(a) compares RGAR retrieval strategies with SCAS retained.

Cost accounting. Appendix B.1 distinguishes the matched online budget from runtime, gradient evaluations, LLM calls, and memory, clean-evaluation, and test costs.

Aggregation. Each unit computes $D = (R_{\mathrm{clean}} - R_{\mathrm{attack}})/(|R_{\mathrm{clean}}| + 1)$ from held-out mean returns. We average seeds and families equally within tasks, then tasks equally within each suite. Normalization precedes task aggregation; test episodes provide evaluation replicates within each search unit.

Paired uncertainty. Tables 29 and 30 report method-mean 95% CIs for Table 2(a)/(b). Tables 31 and 32 report $\Delta = \overline{D_{\mathrm{WMAttack}} - D_{\mathrm{baseline}}}$ and paired CIs; positive differences favor WMAttack, without RGAR in cold start. Inference uses paired method differences and their confidence intervals. Within Atari and DMC separately, we average families within task–seed pairs, then resample tasks and seeds within tasks using shared indices across methods and equal task weights. The 2.5th/97.5th percentiles of 5,000 resamples (seed 831) give unadjusted 95% CIs for each paired comparison. Resampling units are tasks and search seeds.

Table 29: Full-method search results corresponding to Table 2(a): mean normalized return degradation and 95% bootstrap CI bounds.

	Atari			DMC		
Searcher	Mean	Lower	Upper	Mean	Lower	Upper
Coarse grid	0.755	0.363	1.157	0.503	0.281	0.695
Random Search	1.163	0.857	1.486	0.624	0.498	0.856
GP-UCB-style	1.289	0.727	1.723	0.767	0.623	0.879
Successive Halving	1.301	0.661	2.018	0.819	0.715	0.891
WMAttack	1.439	1.175	1.820	0.952	0.901	0.984

Paired comparisons support the mean advantage of full WMAttack. As shown in Table 31, full WMAttack achieves positive mean differences in normalized return degradation against all four search baselines on both Atari and DMC, and each paired 95% bootstrap confidence interval excludes zero. Relative to Successive Halving, the baseline with the highest mean degradation, the gains are 0.138 (95% CI: $[0.036, 0.195]$) on Atari and 0.133 ($[0.049, 0.266]$) on DMC. These results provide statistical support for stronger average attacks than the evaluated baseline implementations under the same online rollout budget.

WMAttack retains a mean advantage without historical access. In the cold-start setting, the mean differences against all four baselines and the lower bounds of their paired 95% bootstrap confidence intervals remain positive on both suites, as shown in Table 32. Relative to Successive Halving, WMAttack without RGAR improves normalized return degradation by 0.090 (95% CI:

Table 30: Cold-start search results corresponding to Table 2(b): mean normalized return degradation and 95% bootstrap CI bounds.

	Atari			DMC		
Searcher	Mean	Lower	Upper	Mean	Lower	Upper
Coarse grid	0.608	0.297	0.896	0.446	0.153	0.714
Random Search	1.095	0.791	1.334	0.597	0.352	0.840
GP-UCB-style	1.119	0.738	1.462	0.605	0.358	0.811
Successive Halving	1.121	0.806	1.494	0.623	0.415	0.870
WMAttack w/o RGAR	1.211	0.975	1.372	0.787	0.649	0.884

Table 31: Paired differences in normalized return degradation for full WMAttack versus each baseline in Table 2(a). Entries are mean differences [95% bootstrap CI]; positive values favor WMAttack.

Baseline	Atari difference [95% CI]	DMC difference [95% CI]
Coarse grid	$+0.684$ $[+0.390, +0.803]$	$+0.449$ $[+0.130, +0.591]$
Random Search	$+0.276$ $[+0.046, +0.293]$	$+0.328$ $[+0.076, +0.385]$
GP-UCB-style	$+0.150$ $[+0.072, +0.187]$	$+0.185$ $[+0.076, +0.216]$
Successive Halving	$+0.138$ $[+0.036, +0.195]$	$+0.133$ $[+0.049, +0.266]$

$[0.048, 0.104]$) on Atari and 0.164 ($[0.115, 0.249]$) on DMC. Thus, the search advantage does not depend entirely on historical experience: the remaining pipeline, which retains current-task feedback, remains effective without retrieval. The incremental contribution of RGAR is examined separately through the controlled retrieval comparisons in Table 4.

Table 32: Paired differences in normalized return degradation for WMAttack without RGAR versus each baseline in Table 2(b). Entries are mean differences [95% bootstrap CI]; positive values favor WMAttack without RGAR.

Baseline	Atari difference [95% CI]	DMC difference [95% CI]
Coarse grid	$+0.603$ $[+0.481, +0.906]$	$+0.341$ $[+0.188, +0.585]$
Random Search	$+0.116$ $[+0.037, +0.237]$	$+0.190$ $[+0.040, +0.417]$
GP-UCB-style	$+0.092$ $[+0.071, +0.145]$	$+0.182$ $[+0.147, +0.240]$
Successive Halving	$+0.090$ $[+0.048, +0.104]$	$+0.164$ $[+0.115, +0.249]$

J.3 Execution and Reproducibility Index

Experimental units. Table 33 indexes series by task, checkpoint, family, repetition, and method/arm. Pairing is within each series; clean/attack tests share episode seeds after configuration freezing. Test episodes are evaluation replicates. Full-search, cold-start, RGAR, and SCAS cover the same 46 tasks but remain separate series, as do the four-task retrieval and Flip experiments.

Episode settings. The 46-task DreamerV3 experiments use the same task-specific environment settings for clean, scout, confirmation, and held-out episodes. Atari uses frame skip 4, an outer time limit of 27,000 agent steps, and no termination on loss of a life. DMC uses action repeat 2 and native episode termination or truncation, without an additional outer time limit. Episodes end on environment termination or truncation.

LLM backend and decoding. All enabled WMAttack proposer and Auditor modules use Qwen2.5-3B-Instruct, except in the backbone ablation in Appendix K. This setting applies to full-search and cold-start WMAttack in Table 2, the 46-task and independent four-task RGAR comparisons, and the SCAS and Flip ablations. SCAS arms retain or remove modules as specified in Appendix G.1. Across all 46 tasks in the full-search, cold-start, RGAR, and SCAS series, every enabled LLM module uses greedy constrained decoding (temperature 0) with at most 768 new tokens per call. The same decoding settings apply to the independent four-task RGAR and Flip experiments.

Runtime and calls. SCAS, Flip, and independent four-task RGAR run through Transformers on V100 GPUs. Invalid output or backend failure stops the unit without heuristic replacement or effect-

Table 33: Index of controlled experimental series. All use fixed DreamerV3 victims and three search repetitions per task–family. T, F, and A denote tasks, families, and methods/arms. Each unit uses 18 online attack episodes plus separately accounted clean and held-out evaluation.

Series	$T \times F \times A$	Historical access	Protocol and statistics
46-task full search	$46 \times 4 \times 5$	Full WMAttack only; baselines have none	Appendix J.2; suite-level task/seed bootstrap
46-task cold search	$46 \times 4 \times 5$	Disabled for all methods	Appendix J.2; suite-level task/seed bootstrap
46-task RGAR	$46 \times 4 \times 4$	None, uniform, behavior, or composite retrieval	Appendix G.2; suite-level task/seed bootstrap
Four-task RGAR	$4 \times 4 \times 4$	Same four access rules; separate frozen pool	Appendix G.2; fixed-stratum repetition/episode bootstrap
46-task SCAS	$46 \times 4 \times 4$	Disabled; numerical feedback retained	Appendix G.1; fixed-stratum repetition/episode bootstrap
Flip Utility weight	$4 \times 4 \times 2$	Disabled; Flip feedback/ranking retained	Appendix C.1; fixed-stratum repetition/episode bootstrap

dependent reruns. SCAS arms A/B/C/D make 15/12/3/0 calls per unit; each Flip or four-task RGAR arm makes 15.

Seeds and aggregation. Full-search, cold-start, and 46-task RGAR use search seeds 0, 1, and 2. SCAS and Flip use separate online/held-out seed domains, respectively 23100000/23200000 and 22100000/22200000, before task, family, repetition, and stage offsets. Pre-validation data are excluded. To reconstruct results, group paired clean/attack returns by series, unit, selected configuration, and test seed. Compute unit-level D from mean returns, then apply the declared weighting and paired resampling. Full-search, cold-start, and 46-task RGAR resample tasks/seeds; SCAS, Flip, and four-task RGAR condition on task–family strata and resample repetitions/paired test episodes.

K LLM-Backbone Sensitivity

Setup. The proposer and Auditor use the same backbone and are varied jointly across 26 DreamerV3 Atari tasks through Transformers. The checkpoints are `meta-llama/Llama-3.2-1B-Instruct`, `meta-llama/Llama-3.2-3B-Instruct`, `meta-llama/Meta-Llama-3.1-8B-Instruct`, `Qwen/Qwen2.5-1.5B-Instruct`, `Qwen/Qwen2.5-3B-Instruct`, and `Qwen/Qwen2.5-7B-Instruct`.

Table 34: LLM-backbone sensitivity on 26 DreamerV3 Atari tasks.

Model	Size	Drop ↑	Flip ↑	Utility ↑	Time (s)
LLaMA 3.2	1B	0.715	0.637	0.114	5528
LLaMA 3.2	3B	0.812	0.668	0.210	6803
LLaMA 3.1	8B	0.769	0.629	0.171	5215
Qwen2.5	1.5B	0.754	0.640	0.168	**5164**
Qwen2.5	3B	**1.034**	0.660	**0.430**	6338
Qwen2.5	7B	0.916	**0.675**	0.314	8252

Results. Table 34 and Figure 6(b) compare Qwen (Team, 2024) and LLaMA (Dubey et al., 2024) backbones for both modules. Qwen2.5-3B achieves the highest Drop (1.034) and Utility (0.430). Qwen2.5-7B achieves the highest Flip (0.675) and has the longest recorded evaluation time; Qwen2.5-1.5B has the shortest time but lower Drop. Thus, within Qwen2.5, increasing model size does not consistently improve attack performance.