ALOHA: AGENTIC LLM-DRIVEN ONLINE HYPERPARAMETER ADAPTATION FOR REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Hyperparameter optimization for reinforcement learning is computationally expensive. Offline methods such as Bayesian optimization require hundreds of complete training runs while population-based methods demand dozens of parallel agents, making both approaches impractical when simulation costs are high relative to the available compute budget. We introduce ALOHA (Agentic LLM-driven Online Hyperparameter Adaptation), a method for online hyperparameter optimization within a single training run, that embeds a large language model directly into the RL training loop to adaptively adjust hyperparameters at fixed intervention points based on online training metrics, requiring no preprocessing and no parallel agents. At each intervention, ALOHA produces updated hyperparameter values alongside natural-language reasoning traces that explicitly justify each adjustment, making each decision interpretable. We evaluate ALOHA across three RL algorithms (PPO, DQN, SAC) and a diverse suite of environments spanning continuous locomotion, discrete control, and grid navigation, benchmarked using ARLBench. Our results demonstrate that ALOHA matches or exceeds Bayesian optimization in 9 of 14 environments and outperforms population-based training across all 14, requiring no offline search budget compared to Bayesian optimization SMAC's 400 total training runs (80 trials $\times$ 5 seeds) per environment, without any preprocessing runs, parallel agents, or domain-specific tuning.

1 INTRODUCTION

Reinforcement learning algorithms are notoriously sensitive to hyperparameter configuration (Eimer et al., 2023). Small changes to the learning rate, entropy coefficient, or clipping threshold can determine whether an agent converges to a strong policy or collapses entirely (Eimer et al., 2023; Andrychowicz et al., 2021). This sensitivity has motivated a substantial body of work on automated hyperparameter optimization (HPO) in RL, ranging from Bayesian optimization (Akiba et al., 2019) to Population-Based Training (Jaderberg et al., 2017) and its variants (Parker-Holder et al., 2020; Bai & Cheng, 2024). These methods are powerful but share a common assumption that substantial computational resources are available for preprocessing before deployment. Bayesian optimization can require hundreds of complete training runs to identify a good starting configuration. Population-based methods require running 8 to 32 agents in parallel throughout training. In settings where this assumption fails, existing methods provide no practical solution.

Such settings are common. Fine-tuning a large language model with reinforcement learning from human feedback (Ouyang et al., 2022) may cost thousands of dollars per run, making multi-trial offline search prohibitive. Deploying a robot policy on physical hardware carries safety and wear costs that preclude parallel exploration. Training in high-fidelity simulation where each timestep is computationally expensive imposes strict budgets on total environment interactions. In all these cases, the question is not which HPO method achieves the highest asymptotic performance given unlimited compute, but which method achieves the best performance *within a single training run*.

We ask whether an LLM agent, reasoning over live training diagnostics, can answer this question. Rather than searching over configurations before training or maintaining a population of agents during training, our framework embeds an LLM directly into the training loop. At fixed intervention

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.

points, the agent observes the past history of diagnostic metrics including policy loss, value loss, entropy, KL divergence, clip fraction, and episodic return, and produces updated hyperparameter values accompanied by a natural-language reasoning trace explaining each decision per hyperparameter. This trace, citing specific metric values and established RL failure modes, makes every intervention auditable and interpretable. The framework requires no preprocessing, no parallel agents, and no architectural changes to generalize across algorithms or environments, the algorithm and task context are injected into the prompt at runtime.

We evaluate the framework across 3 RL algorithms and 14 environments drawn from the ARLBench benchmark (Becktepe et al., 2026), under two ablation conditions ranging from zero-shot LLM intervention to hybrid retrieval augmentation with a structured knowledge graph. All results are reported as mean and standard deviation across 10 independent seeds. Our central finding is that **zero-shot LLM intervention, requiring no preprocessing of any kind, matches or exceeds Bayesian optimization across several environments while consuming orders of magnitude fewer total environment steps**. For example on LunarLander, the best zero-shot result achieves a mean episodic return of 288.6 compared to 194.4 for Bayesian optimization. A structured ablation further reveals that retrieval augmentation does not consistently improve over zero-shot, suggesting that the parametric knowledge of modern open-weight LLMs is sufficient for this class of task (Chen et al., 2025).

Contributions - This paper makes the following contributions:

- **An online agentic HPO framework** that operates within a single training run at zero preprocessing cost, producing interpretable natural-language reasoning traces at every intervention point. The framework is algorithm and task agnostic by design, generalizing across RL algorithms (here PPO, DQN, and SAC are shown in results) through runtime prompt configuration.
- **A comprehensive empirical evaluation** across 3 RL algorithms, 14 environments spanning locomotion, discrete control, and grid navigation, with all results reported over 10 independent seeds. Main paper results use **gemma4-31B**, selected as the best-performing model across environments, full per-model results for all 5 open-weight LLMs are provided in Appendix C.
- **A compute-budget-controlled comparison** demonstrating that the agentic method achieves competitive performance with offline HPO methods at a fraction of their total environment interaction cost, as illustrated in Figure 2.
- **Evidence that LLM parametric knowledge is sufficient for online RL HPO**, validated through the hybrid RAG ablation demonstrating robustness to any context injection.

2 METHOD

ALOHA embeds a large language model directly into the reinforcement learning training loop as an online hyperparameter controller. Rather than searching for a fixed configuration before training begins, the agent observes live training metrics at regular intervention points and issues updated hyperparameter values accompanied by natural-language justifications for each decision. The framework requires no preprocessing runs, no parallel agents, and no architectural modifications to the underlying RL algorithm. Algorithm and environment context are injected into the prompt at runtime, making the method agnostic to both the choice of RL algorithm and the task domain. We evaluate ALOHA across three RL algorithms: Proximal Policy Optimization (PPO) (Schulman et al., 2017), Deep Q-Network (DQN) (Mnih et al., 2015), and Soft Actor-Critic (SAC) (Haarnoja et al., 2018), covering both discrete and continuous action spaces. Figure 1 illustrates the complete pipeline.

2.1 PROBLEM SETUP

Let $\mathcal{E}$ denote a reinforcement learning environment with state space $\mathcal{S}$ and action space $\mathcal{A}$. A policy π_θ parameterized by θ is trained using an RL algorithm $\mathbf{ALG} \in \{\text{PPO, DQN, SAC}\}$ with a hyperparameter vector $\mathbf{h} \in \mathcal{H} = \prod_i [h_i^{\min}, h_i^{\max}]$, where $\mathcal{H}$ is a bounded hyperparameter search space with per-parameter log-scale or linear bounds depending on the parameter type (see Appendix B). Training proceeds in N fixed intervals, where each one corresponds to a contiguous block of environment interaction steps. Let $\mathbf{m}_t \in \mathbb{R}^d$ denote a vector of d scalar training statistics observed at the end of each interval t (detailed in Section 2.2), $\mathcal{W}$ the space of windows of such vectors, $\mathcal{R}$

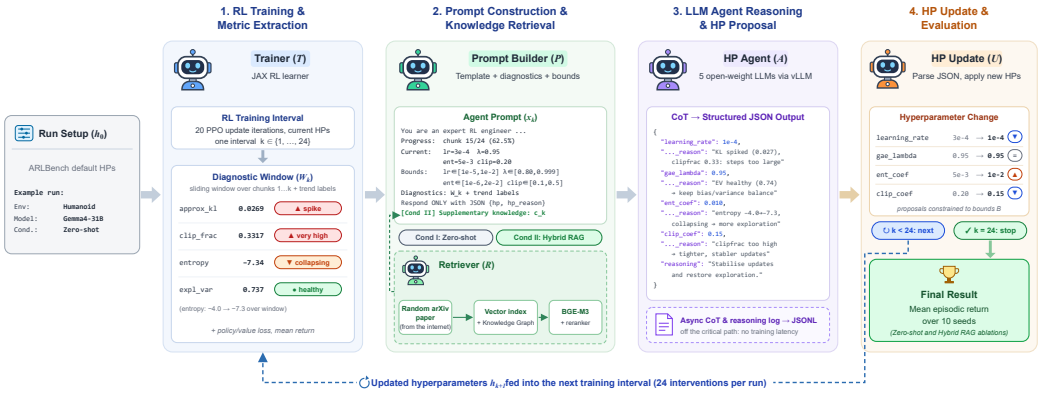


Figure 1: System architecture of ALOHA. The online training loop (blue) repeats $N = 24$ times across the training horizon. At each intervention point, the Metric Extractor assembles a diagnostic window from live training signals, the LLM Intervention Agent reasons over these signals and produces updated hyperparameters with natural-language justifications, and the updated configuration is applied to the next training interval. For the Hybrid RAG ablation, the Retriever (green) supplies additional context from a single retrieved paper processed through vector search and a knowledge graph; this component is absent in the zero-shot condition.

the space of retrieved text context (with $\mathcal{R} = \{\emptyset\}$ for the zero-shot condition), and $\mathcal{T}$ the space of natural-language reasoning traces.

Given a single training run on $\mathcal{E}$ with algorithm $\mathbf{ALG}$, the goal is to produce a sequence of hyperparameter vectors $\mathbf{h}_1, \ldots, \mathbf{h}_N$ that maximizes the final episodic return $J(\pi_{\theta_N})$, using only information available within the run itself and without any preprocessing runs, parallel agents, or pre-optimized starting configurations.

2.2 Pipeline Components and Training Procedure

The agentic pipeline decomposes into three components operating at each interval boundary $t \in \{1, \ldots, N\}$, followed by a training step that applies the updated configuration.

Metric Extractor E. After each training interval, the metric extractor observes the current training state and assembles a diagnostic vector:

$$\mathbf{m}_t = \mathbf{E}(\pi_{\theta_t}, \mathcal{E}) \in \mathbb{R}^d, \tag{1}$$

where $\mathbf{m}_t$ contains eight scalar statistics: policy loss, value loss, entropy, approximate KL divergence, clip fraction, explained variance, and mean and standard deviation of episodic return over the current interval. The full history of all previous intervention points is maintained:

$$\mathbf{W}_t = [\mathbf{m}_1, \ldots, \mathbf{m}_t] \in \mathbb{R}^{t \times d}, \tag{2}$$

so the agent conditions on the complete record of training dynamics from the start of the run. In addition, derived signals Δ_t are computed from $\mathbf{W}_t$: these include the per-step rate of change for value loss, policy loss, and episodic return (δvalue_loss, δpolicy_loss, δreturn), and discrete trend labels (increasing / stable / decreasing) for entropy, KL divergence, and episodic return. These derived signals compress the temporal structure of the window into tokens the LLM can directly reason over without needing to perform numerical differentiation itself.

LLM Intervention Agent F. The core component maps the diagnostic window, derived signals, current hyperparameters, previous reasoning trace, and optional retrieved context to a new hyperparameter vector and a natural-language reasoning trace:

$$(\mathbf{h}_{t+1},\ \tau_t) = \mathbf{F}(\mathbf{W}_t,\ \Delta_t,\ \mathbf{h}_t,\ r_t,\ \tau_{t-1}) \in \mathcal{H} \times \mathcal{T}, \tag{3}$$

where $\tau_t \in \mathcal{T}$ is a structured natural-language trace comprising a per-hyperparameter justification and an overall summary, and τ_{t-1} provides the agent with continuity across successive decisions by making previous reasoning visible at the next intervention.

The agent is implemented as a single-turn LLM call with a two-part prompt. The *system prompt* establishes the role of an expert RL engineer and specifies the required output format: a single valid JSON object containing the updated hyperparameter values and per-parameter reasoning strings, with no additional text or markdown. The *user prompt* is constructed at each intervention and supplies: the environment name and algorithm, the current training progress as a fraction of total steps, the current hyperparameter values, the metric window $\mathbf{W}_t$ formatted as named lists, the derived signals Δ_t, and the previous intervention's reasoning trace τ_{t-1} where available. For the Hybrid RAG ablation, retrieved context r_t is appended after the metric block. Full prompt templates are provided in Appendix A.

The LLM is required to perform chain-of-thought reasoning over all metrics before committing to output values. The output JSON is parsed deterministically and each value is clamped to $\mathcal{H}$ before application, ensuring outputs are always valid regardless of model behaviour. A concrete example of an intervention output is:

> {"learning_rate": 0.0001, "learning_rate_reason": "approx_kl spiked to 0.027 and clipfrac reached 0.33, indicating excessively large policy updates. Reducing the learning rate is necessary to stabilise the policy.", "clip_coef": 0.15, "clip_coef_reason": "Clipfrac of 0.33 is extremely high; reducing clip_coef in conjunction with lower learning rate will ensure more controlled updates.", ..., "reasoning": "Policy experiencing high-magnitude updates and rapid entropy collapse; reducing LR and clip while increasing entropy coefficient to stabilise and restore exploration."}

We evaluate 5 open-weight LLMs served locally via vLLM: **gemma4-31B**, **gemma4-26B-A4B**, **Qwen3.5-9B**, **Qwen3.6-35B-A3B**, and **Qwen3.6-27B**. All models are loaded in bfloat16 precision and served via an OpenAI-compatible vLLM endpoint, with temperature set to 0 for deterministic output. Main paper results use **gemma4-31B**, which performed best across the majority of environments, full per-model results are provided in Appendix C.

Retriever $\mathbf{Ret}$ (Hybrid RAG ablation only). For the Hybrid RAG ablation, a retriever maps the current diagnostic window to relevant external context drawn from a single paper retrieved from arXiv at runtime:

$$r_t = \mathbf{Ret}(\mathbf{W}_t,\ \Delta_t,\ \mathcal{C},\ \mathcal{G}) \in \mathcal{R}, \tag{4}$$

where $\mathcal{C}$ is the text of a single paper retrieved from arXiv at runtime, and $\mathcal{G}$ is a knowledge graph constructed from $\mathcal{C}$ using gemma4-31B to extract structured triplets of the form (subject, predicate, object, confidence, source text). Context is retrieved via both dense vector search (FAISS index with BGE-M3 embeddings) and knowledge graph traversal, and the combined candidates are reranked by a cross-encoder before the top-k passages are appended to the user prompt. For the zero-shot condition, $r_t = \emptyset$ and this component is absent entirely.

Training Step $\mathbf{Train}$. The updated hyperparameters are applied to the next training interval:

$$\theta_{t+1} = \mathbf{Train}(\theta_t,\ \mathbf{h}_{t+1},\ \mathcal{E},\ \mathbf{ALG}), \tag{5}$$

where $\mathbf{Train}$ executes one interval of C environment interaction steps under algorithm $\mathbf{ALG}$ and returns the updated policy parameters. Hyperparameters are hot-swapped between intervals with no reinitialization of policy weights or optimizer state, preserving training continuity.

The complete procedure iterates these components $N = 24$ times across the training horizon, producing a final policy π^* and a full sequence of reasoning traces $\{\tau_t\}_{t=1}^{N}$ that constitutes the interpretability record of the run:

$$\pi^* = \mathbf{AgenticRL}\big(\mathcal{E},\ \mathbf{ALG},\ \mathbf{h}_0,\ \{(\mathbf{W}_t,\ \Delta_t,\ r_t) \to (\mathbf{h}_{t+1},\ \tau_t)\}_{t=1}^{N}\big)\,. \tag{6}$$

2.3 Ablation Conditions

Two conditions vary the retriever component $\mathbf{Ret}$ while holding all other components fixed, allowing the contribution of external context to be isolated.

Condition I (Zero-Shot): $r_t = \emptyset$ for all t. No external context is supplied; the agent relies entirely on live training metrics and the parametric knowledge of the LLM. This condition tests whether the LLM's internalized understanding of RL training dynamics is sufficient for effective online HPO without any retrieval infrastructure.

Condition II (Hybrid RAG): A single paper is retrieved from arXiv and processed through the full retrieval pipeline, dense vector search over FAISS-indexed BGE-M3 embeddings combined with knowledge graph traversal with retrieved passages reranked and appended to the prompt. This condition tests whether supplying structured external context beyond parametric knowledge improves intervention quality, and whether the pipeline is robust to variation in the relevance of the retrieved paper.

3 Experiments

3.1 Experimental Setup

Benchmark and Environments. All experiments are conducted using ARLBench (Becktepe et al., 2026), a JAX-accelerated benchmark for HPO in RL providing efficient implementations of DQN, PPO, and SAC compatible with multiple environment frameworks. We evaluate across 14 environments spanning four domains: continuous locomotion (Brax (Freeman et al., 2021b): *humanoid*, *halfcheetah*), discrete control (Gymnax (Lange, 2022): *Acrobot-v1*, *MountainCarContinuous-v0*; EnvPool (Weng et al., 2022): *LunarLander-v2*, *BattleZone-v5*, *Phoenix-v5*, *DoubleDunk-v5*, *BipedalWalker-v3*), and grid navigation (XLand-MiniGrid (Nikulin et al., 2024): *MiniGrid-EmptyRandom-5x5*, *MiniGrid-FourRooms*). We use ARLBench's recommended environment set in our experiments: for PPO using LunarLander-v2, brax/humanoid, BattleZone-v5, Phoenix-v5, and MiniGrid-EmptyRandom-5x5; for DQN on Acrobot-v1, MiniGrid-EmptyRandom-5x5, BattleZone-v5, DoubleDunk-v5, and MiniGrid-FourRooms; and for SAC using BipedalWalker-v3, brax/humanoid, MountainCarContinuous-v0, and brax/halfcheetah. All runs use 10 M total environment steps.

Baselines. We compare against three baselines, all run under the ARLBench framework to ensure a controlled comparison:

Default HP: A single training run using ARLBench's default hyperparameter configuration for each algorithm, derived from the Stable-Baselines3 zoo (Raffin et al., 2021) and Brax hyperparameter sweeps. This represents the zero-cost baseline requiring no HPO.

Bayesian Optimization (SMAC): Sequential Model-based Algorithm Configuration (Lindauer et al., 2022b) as implemented in ARLBench's built-in HPO interface, using a random forest surrogate model over the full ARLBench hyperparameter search space for each algorithm. We match a budget of approximately 80 trials with 5 seeds per trial, consistent with established HPO evaluation practice (Eimer et al., 2023). This baseline represents the offline HPO regime: all search budget is consumed before a final configuration is selected and evaluated. For PPO this covers learning rate, GAE lambda, entropy coefficient, and policy clipping, for DQN, learning rate, initial and target epsilon, and exploration fraction, and for SAC, learning rate, entropy temperature (α), target network rate (τ), and discount factor (γ). Full search space bounds are provided in Appendix B.

Population-Based Training (PBT): PBT (Jaderberg et al., 2017) as implemented in ARLBench, run with population sizes of 5, 10, and 20 agents. All three population sizes are included in the compute budget comparison (Section 3.3); the main results table reports PBT with 20 agents as it achieves the strongest performance.

Evaluation Protocol. All results are reported as mean episodic return and standard deviation across 10 independent seeds. ALOHA results use a single training run of 10 M steps with no preprocessing budget.

Setting		Benchmarking		Agentic (ALOHA)	
Environment	**Baseline**	**Bayesian Opt**	**PBT**	**Zero-shot**	**Hybrid RAG**
PPO					
LunarLander-v2	$-53.9_{\pm 230.4}$	$194.4_{\pm 23.2}$	$149.8_{\pm 54.8}$	$\mathbf{288.6_{\pm 5.96}}$	$279.4_{\pm 12.1}$
Humanoid	$5355.4_{\pm 859.3}$	$\mathbf{5953.7_{\pm 200.8}}$	$4625.5_{\pm 809.2}$	$5121.8_{\pm 265.2}$	$5123.2_{\pm 215.3}$
Battlezone-v5	$15829.2_{\pm 9206.9}$	$\mathbf{20429.2_{\pm 11742.4}}$	$15946.8_{\pm 6786.9}$	$20178.4_{\pm 8325.1}$	$19965.8_{\pm 7648.2}$
Phoenix-v5	$4615.2_{\pm 4682.1}$	$10586.7_{\pm 5949.8}$	$5897.9_{\pm 6892.5}$	$\mathbf{11536.3_{\pm 3705.2}}$	$11481.5_{\pm 3325.7}$
Empty Random	$0.906_{\pm 0.2}$	$0.965_{\pm 0.0}$	$0.965_{\pm 0.01}$	$0.969_{\pm 0.0}$	$\mathbf{0.97_{\pm 0.0}}$
DQN					
Acrobot-v1	$-117.7_{\pm 105.9}$	$-80.8_{\pm 8.1}$	$-82.5_{\pm 4.6}$	$\mathbf{-75.6_{\pm 25.6}}$	$-81.2_{\pm 15.3}$
Empty Random	$0.214_{\pm 0.34}$	$0.653_{\pm 0.40}$	$0.277_{\pm 0.41}$	$0.697_{\pm 0.25}$	$\mathbf{0.714_{\pm 0.32}}$
Battlezone-v5	$11822.9_{\pm 6799.7}$	$\mathbf{20960.4_{\pm 6983.9}}$	$11992.7_{\pm 8763.8}$	$20364.7_{\pm 6231.2}$	$20431.5_{\pm 5897.0}$
DoubleDunk-v5	$-18.49_{\pm 3.03}$	$-0.974_{\pm 2.64}$	$-1.218_{\pm 0.66}$	$\mathbf{-0.625_{\pm 1.78}}$	$-0.796_{\pm 2.24}$
FourRooms	$0.026_{\pm 0.04}$	$0.054_{\pm 0.04}$	$0.038_{\pm 0.02}$	$\mathbf{0.071_{\pm 0.04}}$	$0.068_{\pm 0.05}$
SAC					
BipedalWalker-v3	$34.57_{\pm 144.5}$	$154.48_{\pm 158.8}$	$134.47_{\pm 191.2}$	$\mathbf{298.76_{\pm 25.05}}$	$286.78_{\pm 37.6}$
Humanoid	$4337.0_{\pm 1125.2}$	$\mathbf{4736.1_{\pm 765.5}}$	$4503.1_{\pm 649.4}$	$4583.7_{\pm 230.8}$	$4613.3_{\pm 141.7}$
MountainCarC-v0	$12.645_{\pm 33.53}$	$-0.003_{\pm 0.006}$	$0.0_{\pm 0.0}$	$\mathbf{18.98_{\pm 40.01}}$	$14.46_{\pm 63.8}$
Halfcheetah	$4503.8_{\pm 4299.8}$	$\mathbf{8641.8_{\pm 1950.4}}$	$5922.6_{\pm 1099.9}$	$7825.6_{\pm 1304.9}$	$8194.2_{\pm 1056.7}$

Table 1: Mean episodic return ($\pm$ std, 10 seeds, 10M timesteps) across all methods and environments. "MountainCarC-v0" refers to MountainCarContinuous-v0. All baselines (Default HP, Bayesian Opt via SMAC, and PBT) were run using the ARLBench framework (Becktepe et al., 2026). PBT reports 20-agent results as the strongest configuration; results for 5 and 10 agents are included in the compute budget analysis (Section 3.3). ALOHA results use gemma4-31B; per-model comparisons across all 5 open-weight LLMs are in Appendix C.

ALOHA Configuration. ALOHA is evaluated under two conditions as described in Section 2: zero-shot (Condition I) and Hybrid RAG (Condition II). The backbone LLM for all main paper results is **gemma4-31B**, selected as the best-performing model across environments in a model comparison study, full per-model results for all five open-weight LLMs are provided in Appendix C. The intervention schedule uses $N = 24$ equally spaced intervention points across the 10 M step training horizon, with hyperparameter bounds as specified in Appendix B.

3.2 Main Results

Table 5 reports mean episodic return and standard deviation for all methods across all 14 environments. ALOHA zero-shot matches or exceeds the SMAC baseline on 9 of 14 environments while consuming a single training run with no tuning budget, compared to SMAC's approximately 400 total training runs (80 trials $\times$ 5 seeds) per environment. Hybrid RAG does not consistently improve over zero-shot, consistent with our finding that LLM parametric knowledge is sufficient for this task and that injected context can act as noise (Section 3.4).

3.3 Compute Budget Analysis

Figure 2 plots mean episodic return against total environment steps consumed by each method, including preprocessing overhead. For SMAC, the x-axis position accounts for all trials used across the full search phase (80 trials). For PBT, the x-axis reflects the total steps consumed across all agents in the population (population size $\times$ 10 M steps), shown separately for 5, 10, and 20 agents. ALOHA occupies a single point at 10 M steps with no preprocessing overhead.

This comparison makes the compute efficiency of ALOHA explicit: achieving competitive performance at the leftmost position on the x-axis, without the overhead of offline search or parallel population training.

3.4 Ablation: Zero-Shot vs. Hybrid RAG

To isolate the contribution of retrieved context, we compare Condition I (zero-shot) against Condition II (Hybrid RAG) across all 14 environments. Hybrid RAG supplies additional context from a

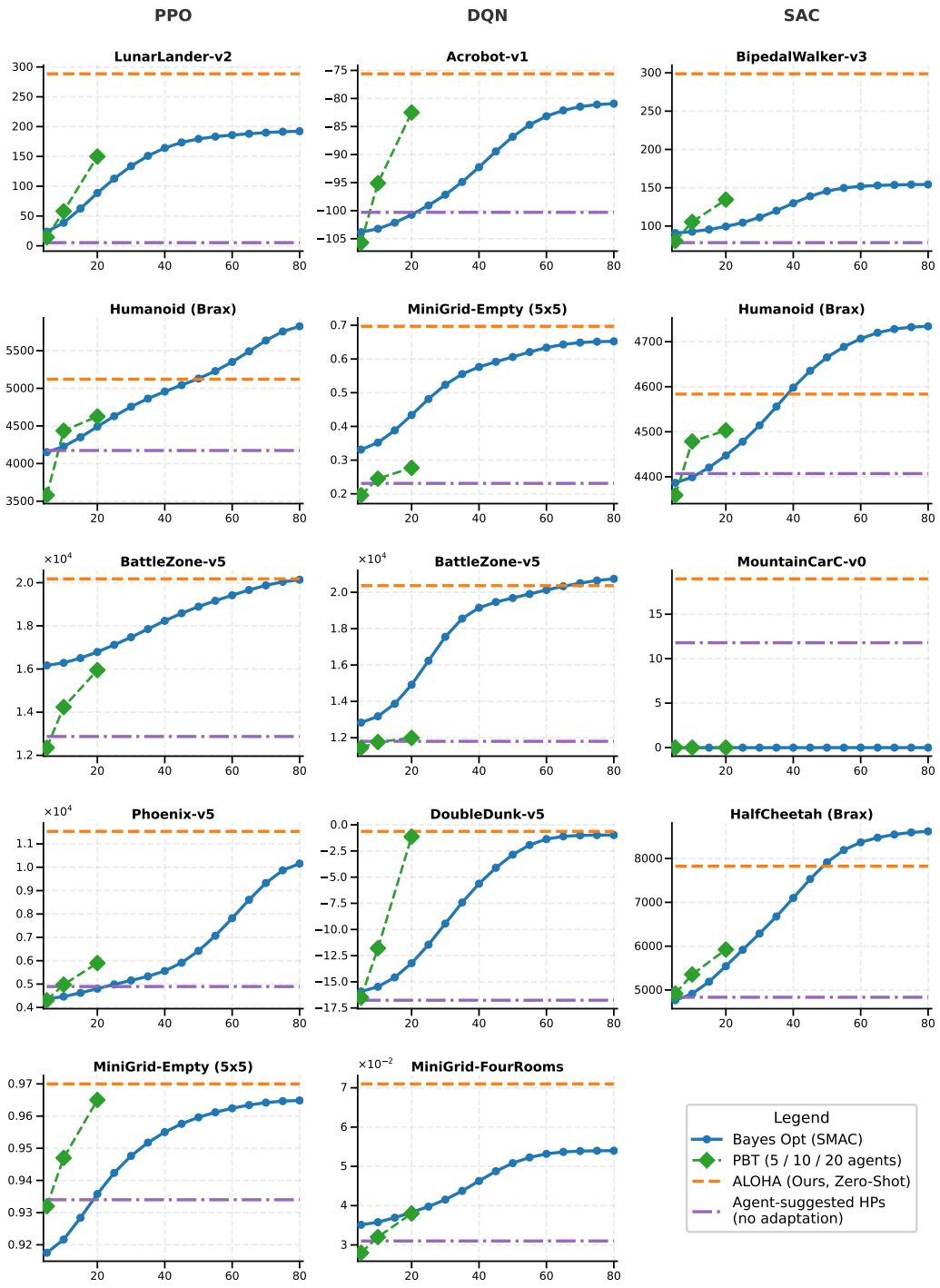


Figure 2: Compute budget comparison across all 14 environments. Best mean episodic return (y-axis) vs. number of offline search trials (x-axis, each evaluated on 5 seeds), where each trial uses 5 evaluation seeds. Bayes Opt (SMAC, blue curve) and PBT (green diamonds - 5, 10, 20 agents) are offline methods whose x-axis position reflects their preprocessing compute cost. ALOHA Zero-Shot (orange dashed) requires no offline trials, it adapts hyperparameters online within a single training run from live metrics alone, shown as a flat line at zero preprocessing cost. Agent-suggested HPs with no adaptation (purple dash-dot) isolates the contribution of online adjustment over a fixed LLM-proposed starting hyperparameters. ALOHA matches or exceeds Bayes Opt in 9 of 14 environments while requiring no offline search budget, compared to SMAC's 80 trials $\times$ 5 seeds $=$ 400 total training runs per environment.

single arXiv paper processed through the full retrieval pipeline, the relevance of this paper to the current training task varies naturally across environments, providing an implicit robustness test.

Across the majority of environments, Hybrid RAG does not improve over zero-shot and in several cases degrades performance, consistent with prior evidence that injected context can act as noise when not precisely aligned with the task at hand. This finding has a practical implication: ALOHA in its simplest zero-shot form, requiring only a locally served LLM and no retrieval infrastructure, is sufficient for effective online HPO.

4 Related Work

Hyperparameter Optimization in Reinforcement Learning: The sensitivity of RL algorithms to hyperparameter configuration has been extensively documented (Henderson et al., 2018; Andrychowicz et al., 2021; Eimer et al., 2023). Static approaches, including random search, grid search, and Bayesian optimization (Akiba et al., 2019; Lindauer et al., 2022a), treat HPO as a black-box search problem, multiple complete training runs are executed offline, and the best-performing configuration is selected for deployment. While Bayesian methods are considerably more sample-efficient than grid search, they remain fundamentally offline, the selected configuration is fixed before training begins and cannot adapt to the non-stationary dynamics that characterize RL training (Eimer et al., 2023). ARLBench (Becktepe et al., 2026) provides a standardized, JAX-accelerated benchmark for comparing HPO approaches across 3 algorithms and 14 environments, and we adopt it as our evaluation framework. Our method differs from all static approaches by requiring no preprocessing runs in its default configuration (zero-shot).

Dynamic and Population-Based Methods: Population-Based Training (PBT) (Jaderberg et al., 2017) addresses the offline limitation by maintaining a population of agents that evolve their hyperparameters dynamically during training. Lower-performing agents periodically copy weights and hyperparameters from higher-performing agents and apply random perturbations, enabling discovery of adaptive schedules rather than fixed configurations. Subsequent variants have refined the mutation strategy: PB2 (Parker-Holder et al., 2020) replaces random perturbation with Gaussian process-guided selection, and Generalised PBT (Bai & Cheng, 2024) introduces pairwise comparison mechanisms. Despite their strength, these methods require 8 to 32 agents training simultaneously, generating orders of magnitude more environment interactions than a single run. Crucially, the mutation step in all population-based methods is semantically blind, i.e. hyperparameter values are perturbed numerically with no understanding of why an agent is underperforming, and no access to the causal relationships between observed diagnostic signals and appropriate interventions. Our framework replaces this blind mutation with LLM reasoning over live diagnostics, operating within a single training run.

LLM Agents for Optimization: The use of LLM agents for hyperparameter optimization has emerged as a promising direction in automated machine learning. AgentHPO (Liu et al., 2025) demonstrated that LLM agents can match or surpass human expert performance across 12 machine learning tasks by iteratively refining configurations based on historical trial outcomes, with natural-language explanations accompanying each decision. LLAMBO (Liu et al., 2024) replaces key components of Bayesian optimization with separate LLMs acting as surrogate model and candidate sampler. JF-HPO (Gao et al., 2026) proposed a multi-fidelity approach for LLM post-training that leverages a small proxy model to reduce the cost of each optimization trial. A key limitation shared across all these methods is that they operate *between* complete training runs, selecting a configuration before each run begins. They cannot respond to dynamics that emerge *within* a run, and have no access to live diagnostic signals. Agentic frameworks have also been applied to LLM post-training, including Agent Laboratory (Schmidgall et al., 2025) for autonomous research workflows and AutoTrainess (Ye et al., 2025) for autonomous post-training pipeline control, though neither operates on live within-run diagnostics. Our framework is the first to embed LLM reasoning directly into the training loop, intervening at fixed points during a single run based on observed metric trends rather than historical trial outcomes.

Interpretability in Automated HPO: Most HPO methods, including population-based approaches and surrogate-model methods, produce hyperparameter updates that are not directly in-

terpretable, the reasons for a particular configuration change are opaque to practitioners. AgentHPO provides post-hoc explanations of why configurations were selected across trials, but these explanations describe aggregate trial histories rather than specific within-run diagnostic events. Our framework produces a per-intervention reasoning trace at every update point during training, citing specific metric values and identified failure modes, making each hyperparameter decision auditable in real time.

Retrieval Augmentation in Machine Learning Pipelines: Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has been widely adopted to ground LLM outputs in external knowledge. In machine learning optimization pipelines, however, its benefit is less clear. R&D-Agent (Chen et al., 2025) found that RAG consistently degrades performance on standard ML optimization tasks, with retrieval providing benefit only for genuinely novel or out-of-distribution challenges where parametric LLM knowledge is insufficient. We evaluate 2 conditions including a retrieval ablation and find consistent results that retrieval augmentation does not drastically improve over zero-shot LLM intervention for RL HPO, validating the sufficiency of parametric knowledge for this kind of task.

5 Conclusion

We presented ALOHA, an agentic LLM-driven framework for online hyperparameter adaptation in reinforcement learning that operates within a single training run at zero tuning cost. By embedding an LLM directly into the training loop at fixed intervention points, ALOHA produces interpretable natural-language reasoning traces at every decision and adapts hyperparameters from live diagnostic signals alone, without parallel agents, offline search, or domain-specific tuning. Across 14 environments spanning PPO, DQN, and SAC, ALOHA zero-shot matches or exceeds Bayesian optimisation (SMAC) in 9 of 14 environments while consuming 400$\times$ less compute (eliminating 80 trials $\times$ 5 seeds of preprocessing) and outperforms PBT across all 14 environments, demonstrating that the compute-constrained regime where offline HPO methods are simply infeasible is where ALOHA is most practically valuable. A key finding is that retrieval augmentation does not improve over zero-shot: injected context acted as noise to the LLM's parametric knowledge rather than augmenting it, with Hybrid RAG outperforming zero-shot in only 2 of 14 environments, suggesting that modern open-weight LLMs already possess sufficient internalized RL knowledge for effective online HPO without any retrieval infrastructure.

Two limitations are worth noting for future work. First, in this work we treat the network architecture as fixed, but this too could be controlled. Extending ALOHA to dynamic architecture search, for example via a supernet with binary masking that allows structural changes without reinitializing the policy is a natural next step. Second, intervention points are fixed rather than adaptive: JAX compiles computation graphs ahead of time for GPU execution, making live metric polling between steps expensive. Fixed intervals after complete rollout intervals allow the policy to execute under a given configuration long enough to produce meaningful training metrics signal before the LLM is queried. Adaptive triggering based on detected training instability, combined with trajectory-informed interventions that accumulate experience across the run, represents a promising direction toward more responsive online HPO.

AI use statement

We did not use generative AI tools to design or evaluate research methodology, propose or refine hypotheses, conduct literature review, interpret experimental results, formulate or prove mathematical claims, develop theoretical frameworks, or generate synthetic data. AI assistants were used for implementation support, including writing and debugging code for experiment execution and result extraction. All AI-assisted code was reviewed and verified by the authors prior to use in experiments. We take full responsibility for the final content of this work, including all experimental results, analysis, and writing.

ETHICS STATEMENT

This work presents a framework for automated hyperparameter optimization in reinforcement learning using open-weight large language models served locally. No human subjects, personal data, or sensitive information are involved. The open-weight models used in evaluation are publicly available on HuggingFace and are used solely for hyperparameter suggestion within a controlled experimental setting. We foresee no direct ethical concerns with the proposed method; however, we note that any system that automates training decisions in high-stakes RL deployments (e.g. robotics or safety-critical control) should be subject to appropriate human oversight before adoption.

REPRODUCIBILITY STATEMENT

The ALOHA framework, including the full intervention pipeline, prompt templates, hyperparameter bounds, and evaluation protocol, is described in Section 2 and Appendix A. All experiments use publicly available open-weight models served via vLLM and the ARLBench benchmark (Becktepe et al., 2026) with environment frameworks accessible through standard package installation. Hyperparameter configurations for all baselines are reported in Appendix B. To support full reproducibility, we will release the complete codebase, and experiment configurations upon publication.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, 2019.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? A large-scale study. In *International Conference on Learning Representations*, 2021.

Hui Bai and Ran Cheng. Generalized population-based training for hyperparameter optimization in reinforcement learning. *arXiv preprint arXiv:2404.08233*, 2024.

Jannis Becktepe, Julian Dierkes, Carolin Benjamins, Aditya Mohan, David Salinas, Raghu Rajan, Frank Hutter, Holger H Hoos, Marius Lindauer, and Theresa Eimer. ARLBench: Flexible and efficient benchmarking for hyperparameter optimization in reinforcement learning. *Journal of Data-centric Machine Learning Research*, 2026.

Xu Chen et al. R&D-agent: Automating research and development via LLM-based agents. *arXiv preprint arXiv:2505.14738*, 2025.

Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 9104–9149. PMLR, 2023.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation, 2021a. URL `http://github.com/google/brax`.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021b.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. Efficient hyperparameter optimization for LLM reinforcement learning. *arXiv preprint arXiv:2606.03073*, 2026.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022a.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022b. URL http://jmlr.org/papers/v23/21-0888.html.

Siyi Liu, Chen Gao, and Yong Li. Large language model agent for hyper-parameter optimization. In *Conference on Parsimony and Learning*, 2025.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization, 2024. URL https://arxiv.org/abs/2402.03921.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Artem Agarkov, Viacheslav Sinii, and Sergey Kolesnikov. XLand-MiniGrid: Scalable meta-reinforcement learning environments in JAX. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-1390. URL https://arxiv.org/abs/2312.12044.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. Provably efficient online hyperparameter optimisation with population-based bandits. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17200–17211, 2020.

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

Samuel Schmidgall et al. Agent laboratory: Using LLM agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A highly parallel reinforcement learning environment execution engine. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22409–22421, 2022.

Yiwen Ye et al. AutoTrainess: Towards autonomous LLM post-training. *arXiv preprint arXiv:2606.31551*, 2025.

A Prompt Templates and Reasoning Trace Examples

A.1 Full Prompt

```
You are an expert reinforcement learning engineer with deep knowledge
    of RL
training dynamics, hyperparameter sensitivity, and convergence
    behaviour across
continuous control environments.

Your task: analyse the provided RL training metrics and suggest
    improved
hyperparameters to accelerate convergence and stabilise training.

Reason through ALL metrics carefully before deciding. Consider the
    training
progress percentage.

After your internal reasoning, respond ONLY with a single valid JSON
    object -
no other text, no markdown fences, no explanation outside the JSON:

{
  "HP1":        <float within range1>,
  "HP1_reason": "<2-3 sentences: what the metrics showed for HP1,
                  what you changed and exactly why>",
  "HP2":        <float within range2>,
  "HP2_reason": "<2-3 sentences: what the metrics showed for HP2,
                  what you changed and exactly why>",
  "HP3":        <float within range3>,
  "HP3_reason": "<2-3 sentences: what the metrics showed for HP3,
                  what you changed and exactly why>",
  "HP4":        <float within range4>,
  "HP4_reason": "<2-3 sentences: what the metrics showed for HP4,
                  what you changed and exactly why>",
  "reasoning":  "<one sentence overall summary>"
}
```

A.2 Reasoning Trace (example)

```
Condition: Zero-shot | Model: Gemma4-31B | Algorithm: PPO | Env:
    Humanoid
Chunk: 15/24 (62.5% complete)

Training state:
  approx_kl=0.0269  clipfrac=0.3317  entropy=-7.3431
  explained_variance=0.7370  mean_return=0.0000

Hyperparameter change:
  learning_rate : 0.0003 -> 0.0001
  gae_lambda    : 0.95   (unchanged)
  ent_coef      : 0.005  -> 0.010
  clip_coef     : 0.200  -> 0.150
```

```
Reasoning:
  [learning_rate] The approx_kl has spiked to 0.0269 and
  clipfrac surged to 0.3317, indicating excessively large
  policy updates. Reducing the learning rate is necessary
  to stabilize the policy and prevent further divergence.

  [gae_lambda] Explained variance remains relatively healthy
  at 0.7370 despite the instability. Maintaining the current
  lambda preserves the existing balance between bias and
  variance in the advantage estimates.

  [ent_coef] Entropy is dropping precipitously (from -4.0
  to -7.3), indicating the policy is collapsing into a
  deterministic state too quickly. Increasing ent_coef will
  provide the necessary exploration to escape the current
  low-return regime.

  [clip_coef] The clipfrac of 0.3317 is extremely high,
  indicating the policy is being clipped too frequently.
  Reducing clip_coef in conjunction with a lower learning
  rate will ensure more stable, controlled updates.

  [Overall]: The policy is experiencing high-magnitude
  updates and rapid entropy collapse; reducing LR and clip
  while increasing entropy to stabilize and restore
  exploration.
```

B Hyperparameter Search Spaces and Default Configurations

Tables 2–4 show the hyperparameter search spaces used by SMAC (Bayesian optimisation) for each algorithm, reproduced from Becktepe et al. (2026). Default values are those used for the Default HP baseline, derived from the Stable-Baselines3 zoo (Raffin et al., 2021) and Brax hyperparameter sweeps (Freeman et al., 2021a). ALOHA uses the same bounds as hard constraints when clamping LLM-suggested values to valid ranges.

Table 2: PPO hyperparameter search space and default values per environment domain. CC = Classic Control, ALE = Arcade Learning Environment.

Hyperparameter	Search Range	Box2D	XLand	ALE	CC	Brax
learning rate	$\log([10^{-6}, 10^{-1}])$	0.0003	0.0003	0.00025	0.0003	0.0003
entropy coefficient	$[0.0, 0.5]$	0.0	0.0	0.01	0.0	0.0
GAE lambda	$[0.8, 0.9999]$	0.95	0.95	0.95	0.95	0.95
policy clipping	$[0.0, 0.5]$	0.2	0.2	0.1	0.2	0.2
value clipping	$[0.0, 0.5]$	0.2	0.2	0.1	0.2	0.2
value fn coef	$[0.0, 1.0]$	0.5	0.5	0.5	0.5	0.5
max gradient norm	$[0.0, 1.0]$	0.5	0.5	0.5	0.5	0.5
normalize advantages	{Yes, No}	Yes	Yes	Yes	Yes	Yes
batch size	domain-specific	{32,64,128}	{128,256,512}	{128,256,512}	{32,64,128}	{512,1024,2048}
num environments	fixed	16	4	8	1	2048
num steps	fixed	1024	32	128	32	512
update epochs	fixed	4	10	4	4	4

C Per-Model Results

Table 3: DQN hyperparameter search space and default values per environment domain. Conditional parameters (target update interval, buffer α, β, ϵ) are only optimised when their parent condition is active.

Hyperparameter	**Search Range**	**ALE**	**Box2D / CC**	**XLand**
learning rate	$\log([10^{-6}, 10^{-1}])$	0.0001	0.0003	0.0003
initial epsilon	$[0.5, 1.0]$	1.0	1.0	1.0
target epsilon	$[0.001, 0.2]$	0.01	0.05	0.05
buffer size	$[1024, 10^6]$	100000	10000	10000
learning starts	$[1, 2048]$	1000	1024	1024
use target network	{Yes, No}	Yes	Yes	Yes
target update interval	$[1, 2000]$ (conditional)	1000	1000	1000
buffer priority sampling	{Yes, No}	False	False	False
buffer α	$[0.01, 1.0]$ (conditional)	0.9	0.9	0.9
buffer β	$[0.01, 1.0]$ (conditional)	0.9	0.9	0.9
buffer ϵ	$\log([10^{-7}, 10^{-3}])$ (conditional)	10^{-6}	10^{-6}	10^{-6}
batch size	domain-specific	{16,32,64}	{64,128,256}	{32,64,128}
num environments	fixed	8	4	4

Table 4: SAC hyperparameter search space and default values per environment domain. τ and buffer priority parameters are conditional on their respective parent hyperparameters being active.

Hyperparameter	**Search Range**	**Box2D**	**CC**	**Brax**
learning rate	$\log([10^{-6}, 10^{-1}])$	0.0003	0.0003	0.0003
alpha (entropy temp.)	$[0.0, 1.0]$	1.0	1.0	1.0
alpha auto-tune	{Yes, No}	Yes	Yes	Yes
buffer size	$[1024, 10^6]$	10000	10000	10000
learning starts	$[1, 2048]$	128	128	128
use target network	{Yes, No}	Yes	Yes	Yes
target network τ	$[0.01, 1.0]$ (conditional)	1.0	1.0	1.0
reward scale	$\log([0.1, 10])$	1.0	1.0	1.0
buffer priority sampling	{Yes, No}	False	False	False
buffer α	$[0.01, 1.0]$ (conditional)	0.9	0.9	0.9
buffer β	$[0.01, 1.0]$ (conditional)	0.9	0.9	0.9
buffer ϵ	$\log([10^{-7}, 10^{-3}])$ (conditional)	10^{-6}	10^{-6}	10^{-6}
batch size	domain-specific	{128,256,512}	{256,512,1024}	{512,1024,2048}
num environments	fixed	1	1	64

Setting / Environment	Per-Model results (ALOHA-Zero shot) Gemma4-31B	Gemma4-26B	Qwen3.5-9B	Qwen3.6-27B	Qwen3.6-35B
PPO					
LunarLander-v2	**288.6±5.96**	265.7±12.9	231.8±7.9	259.4±13.8	274.7±6.9
Humanoid	5121.8±265.2	4911.7±531.9	4223.7±304.4	**5341.2±579.1**	5217.1±184.1
Battlezone-v5	20178.4±8325.1	**20201.5±9681.3**	17841.2±8381.7	18812.3±7923.1	19607.6±6912.9
Phoenix-v5	**11536.3±3705.2**	10149.8±3106.5	7714.5±5305.1	10512.6±2783.5	11460.3±3495.1
Empty Random	**0.969±0.0**	0.949±0.0	0.961±0.02	0.964±0.0	0.965±0.0
DQN					
Acrobot-v1	**−75.6±25.6**	−81.5±14.6	−93.2±3.7	−85.4±23.8	−79.6±27.5
Empty Random	**0.697±0.25**	0.641±0.37	0.591±0.67	0.635±0.31	0.674±0.27
Battlezone-v5	20364.7±6231.2	19452.5±7179.2	17302.8±7482.9	19579.2±6581.7	**20683.4±7158.2**
DoubleDunk-v5	**−0.625±1.78**	−1.135±1.08	−1.934±2.89	−1.246±1.34	−0.813±2.01
FourRooms	**0.071±0.04**	0.058±0.03	0.036±0.05	0.059±0.04	0.065±0.03
SAC					
BipedalWalker-v3	**298.76±25.05**	264.51±45.1	191.24±164.7	239.41±49.81	267.15±86.45
Humanoid	**4583.7±230.8**	4215.7±324.5	4493.1±587.1	4314.6±341.2	4412.5±153.6
MountainCarC-v0	**18.98±40.01**	11.21±59.21	−0.004±0.0	−0.01±0.0	15.18±91.7
Halfcheetah	7712.3±1561.4	7734.5±1743.7	7541.3±2412.5	**7922.3±1564.7**	7812.3±1011.5

Table 5: Mean episodic return (± std, 10 seeds, 10M timesteps) across all methods and environments. "MountainCarC-v0" refers to MountainCarContinuous-v0. All results were run using the ARLBench framework (Becktepe et al., 2026). This table contains results of ALOHA method tested using different LLMs (zero-shot). Main results of ALOHA use gemma4-31B because it performed well for 10 out of 14 environments.