EVAAA: A Virtual Environment Platform for Essential Variables in Autonomous and Adaptive Agents

Sungwoo Lee[1,2,8*] **Jungmin Lee**[1,2,8*] **Sohee Kim**[1,3,8] **Hyebhin Yoon**[9] **Shinwon Park**[5]

Junhyeok Park[6] **Jaehyuk Bae**[7] **Seok-Jun Hong**[1,2,3,4,8†] **Choong-Wan Woo**[1,2,3,8†]

[1]Center for Neuroscience Imaging Research, Institute for Basic Science (IBS), South Korea, [2]Department of Biomedical Engineering, Sungkyunkwan University, South Korea, [3]Department of Intelligent Precision Healthcare Convergence, Sungkyunkwan University, South Korea, [4]Center for the Developing Brain, Child Mind Institute, NY, USA, [5]Autism Center, Child Mind Institute, NY, USA, [6]Graduate School of Metaverse, KAIST, South Korea, [7]Department of Brain and Cognitive Sciences, KAIST, South Korea, [8]Life-inspired Neural Network for Prediction and Optimization Research Group, South Korea, [9]Neurocore Co., Ltd., Seoul, South Korea

Abstract

Reinforcement learning (RL) agents have demonstrated strong performance in structured environments, yet they continue to struggle in real-world settings where goals are ambiguous, conditions change dynamically, and external supervision is limited. These challenges stem not primarily from the algorithmic limitations but from the characteristics of conventional training environments, which are usually static, task-specific, and externally defined. In contrast, biological agents develop autonomy and adaptivity by interacting with complex, dynamic environments, where most behaviors are ultimately driven by internal physiological needs. Inspired by these biological constraints, we introduce EVAAA (Essential Variables in Autonomous and Adaptive Agents), a 3D virtual environment for training and evaluating egocentric RL agents endowed with internal physiological state variables. In EVAAA, agents must maintain essential variables (EVs)—e.g., satiation, hydration, body temperature, and tissue integrity (the level of damage)—within viable bounds by interacting with environments that increase in difficulty at each stage. The reward system is derived from internal state dynamics, enabling agents to generate goals autonomously without manually engineered, task-specific reward functions. Built on Unity ML-Agents, EVAAA supports multimodal sensory inputs, including vision, olfaction, thermoception, collision, as well as egocentric embodiment. It features naturalistic survival environments for curricular training and a suite of unseen experimental testbeds, allowing for the evaluation of autonomous and adaptive behaviors that emerge from the interplay between internal state dynamics and environmental constraints. By integrating physiological regulation, embodiment, continual learning, and generalization, EVAAA offers a biologically inspired benchmark for studying autonomy, adaptivity, and internally driven control in RL agents. Our code is publicly available at `https://github.com/cocoanlab/evaaa`

[*]Equal contribution: sungwoo320@gmail.com, jungmin.lee@g.skku.edu

[†]Corresponding authors: hongseokjun@skku.edu, waniwoo@skku.edu

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Track on Datasets and Benchmarks.

1 Introduction

Reinforcement learning (RL) has demonstrated impressive performance in various simulated and structured tasks, such as arcade games or robotic control, yet it still struggles to autonomously select goals and adapt to novel, dynamic environments. These limitations become critical in embodied settings, where real-world agents—ranging from household robots, search-and-rescue drones, to planetary rovers [1, 2]—must operate under non-stationary environments without continuous human guidance. In such settings, externally defined reward functions are brittle, and manual reward engineering is neither scalable nor sustainable.

Biological systems provide a compelling model of autonomy, as they inherently generate their own goals based on internal homeostatic needs required for survival, rather than relying on external incentives. This process—known as homeostatic regulation—not only supports self-directed behavior in novel and uncertain environments but also has a longstanding place in the study of adaptive behavior. Ross W. Ashby [3] proposed that intelligent agents maintain their viability by regulating essential internal variables, ensuring these variables remain within bounded ranges despite environmental disturbances. This idea was later formalized in the *Two-Resource Problem* [4], demonstrating how agents must navigate trade-offs between competing essential internal needs, such as refueling and task execution, under limited energy and time constraints. Failure to manage these can lead to behavioral instability. This framework has since been extended to embodied agents in both physical and virtual environments [5, 6, 7], underscoring internal regulation as a foundation for autonomous behavior. More recent work highlights how interoceptive signals that monitor internal states—such as hunger, thirst, or fatigue—inform the regulation of physiological homeostasis and also support cognitive processes including learning, exploration, and risk sensitivity [8, 9, 10, 11]. These regulatory functions are mediated by neuromodulatory systems in the brain, which dynamically influence synaptic plasticity, activation pattern and neural responsiveness in context-sensitive ways. Computationally, such neuromodulatory processes have been likened to adaptive mechanisms in RL, where agents adjust their learning and behavior in response to contextual signals [12, 13]. In particular, recent studies in meta-learning demonstrate that incorporating contextual information, such as reward history or temporal-difference signals of internal states, can guide the tuning of parameters like learning rates or discount factors, thereby enabling more flexible and adaptive behavior [14, 15, 16, 17]. These insights underscore the central role of interoceptive inputs as internal context, which not only regulates homeostasis but also shapes learning and behavior, functioning as a computational underpinning for adaptive, goal-directed behavior in dynamic environments.

This connection motivates our introduction of EVAAA (Essential Variables in Autonomous and Adaptive Agents), a simulation benchmark that places internal state regulation at the core of autonomy and adaptivity by utilizing internal physiological variables (i.e., essential variables) as sources of both intrinsic rewards and modulatory signals. In EVAAA, agents must maintain four essential variables (EVs) to survive—satiation, hydration, body temperature, and tissue integrity (the level of damage)—by interacting with virtual environments progressively complex along the different stages. Unlike previous benchmarks such as MuJoCo [18], DeepMind Lab [19], or ThreeDWorld [20]—which rely on handcrafted, task-specific rewards—EVAAA utilizes a single reward function that is grounded in the internal states, thus preserved across all environments.

To evaluate autonomy, EVAAA features three unique components:

- **Unifying survival-based reward design:** Agent receives intrinsic reward signals essential for survival.
- **Egocentric, multimodal interaction through an embodied agent design:** Embodied 3D agent with egocentric vision, olfaction, thermoception, and collision sensors.
- **Two-tiered environment architecture:** *(i)* A naturalistic training curriculum with stages of increasing complexity, and *(ii)* unseen testbeds to evaluate zero- and few-shot generalization under internal state constraints.

2 Background

The motivation for developing EVAAA arose from a longstanding interest in creating agents that act not only in response to extrinsic goals but also based on internal needs—signals that provide stable,

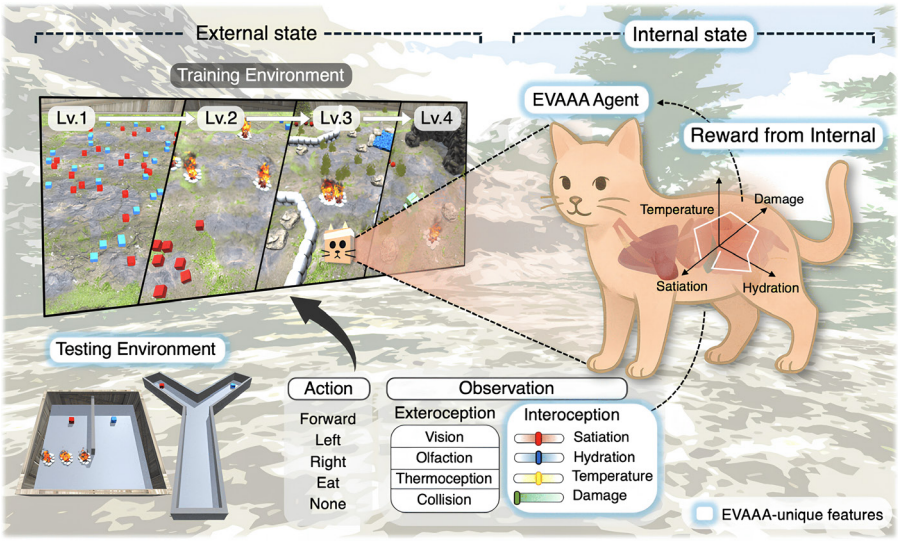

Figure 1: **Overview of the EVAAA simulation environment.** EVAAA integrates an egocentric RL agent with both external and internal state representations. The training environment consists of four levels of incremental complexity (Lv1-Lv4), progressively introducing food, water, obstacles, and predators. Agents navigate and experience the world through multimodal **exteroception** (vision, olfaction, thermoception, and collision) and **interoception** (satiation, hydration, body temperature, and tissue damage). The viable zone (graded colors in the "Interoception" box) of internal variables represents the agent's homeostatic status, and serves as rewards. Testing environments introduce novel, controlled scenarios for evaluating agents' adaptive behaviors and decision-making based on internal states. Components highlighted with the sky-blue outlines indicate EVAAA-unique features.

intrinsic guidance for survival in dynamic and uncertain environments. While much of RL has focused on task-specific supervision, previous studies have explored the role of intrinsic signals in driving behavior. Notably, research on intrinsic motivation [21, 22, 23, 24] and homeostatic RL has framed internal physiological states as reward sources. Keramati and Gutkin [25, 26] proposed a normative theory that unifies reward maximization with the regulation of internal variables by defining reward as a reduction in deviation from a homeostatic set point. Yoshida [7, 27] further extended these ideas by implementing embodied agents in the virtual 3D environment, showing how survival-oriented, autonomous behaviors can emerge through internal-state regulation. While these contributions offer compelling theoretical insights, prior studies have focused primarily on narrow, task-specific implementations, and thus, there remains no benchmark environment for the systematic investigation of a rich array of adaptive, goal-directed behaviors emerging from homeostatic regulation.

EVAAA models agents' behaviors using an extended Markov Decision Process (MDP), defined as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, with the state space factored as $\mathcal{S} = \mathcal{S}_{\text{ext}} \times \mathcal{S}_{\text{int}}$. The external state $\mathcal{S}_{\text{ext}}$ includes exteroceptive sensory inputs (e.g., vision, olfaction, thermoception, collision), while the internal state $\mathcal{S}_{\text{int}}$ consists of four essential variables—*satiation*, *hydration*, *temperature*, and *tissue damage*—which evolve through time-dependent decay (i.e., the internal variable s_i^{int} gradually deviates from its setpoint s_i^* between time steps t and $t+1$, even when the agent takes no action) and agent-environment interactions. The reward function is defined as the temporal reduction in deviation from homeostatic set-point: $R(s_i^{\text{int}}(t), s_i^{\text{int}}(t+1)) = \sum_i \beta_i \left[D_i(s_i^{\text{int}}(t)) - D_i(s_i^{\text{int}}(t+1)) \right]$, where $D_i(s) = (s - s_i^*)^2$. Agents act based on egocentric, multimodal observations composed of exteroceptive inputs and interoceptive signals from $\mathcal{S}_{\text{int}}$ [25, 26].

3 Related works

It is worthwhile to shed light on the key strengths of EVAAA in comparison to recent advances in various benchmark studies for embodied AI and RL in the following three domains: 3D embodiment, survival environment, and animal-inspired benchmarks.

3D Embodiment. Simulation platforms such as MuJoCo [18], DeepMind Lab [19], GibsonEnv [28], Habitat Lab [29], and ThreeDWorld [20] support the training of embodied agents with visual and physical realism. However, these environments depend on externally defined objectives and task-specific rewards, without mechanisms for internal state regulation. EVAAA introduces embodied control anchored in homeostatic regulation, enabling agents to develop adaptive strategies without handcrafted rewards under changing task conditions (see **Table 1**).

Survival Environments. Prior benchmarks such as Crafter [30], Malmo [31], MineRL [32], and Neural MMO [33] simulate survival through health and resource mechanics, with some providing egocentric perception (e.g., Malmo, MineRL), but still lack fully embodied sensorimotor control. Avalon [34] adds first-person 3D embodiment with a scalar energy-based reward, yet abstracts physiological state into a single variable—depleted by movement or damage and replenished only by eating—limiting its potential as a rich, dynamic context for modulating learning and behavior. MARS [35] and AROE [36] emphasize open-world reasoning and task inference, but they lack internal regulation. EVAAA addresses all these limitations by integrating agents in progressively generated 3D environments with multi-dimensional regulation of internal states—including the level of food-supplied nutrients and water, body temperature, and the level of damage—each sensed through a distinct multi-modal channel. These distinctions position EVAAA as a complementary benchmark that broadens the design space for survival environments by enabling richer evaluation of adaptive behavior and internal-state-driven control (see **Table 1**).

Animal-Inspired Benchmarks. Animal-AI Olympics [37, 38], O-PIAGETS [39], and GIBSONA [40] evaluate agents on biologically grounded tasks, such as object permanence, delayed gratification, and spatial elimination. EVAAA advances this direction by introducing two complementary environments: a naturalistic survival environment driven by internal physiological regulation, and an experimental testing environment composed of novel cognitive tasks (e.g., goal manipulation, two-resource choice task, etc). These distinct environments challenge agents to generalize behaviors learned through homeostatic control from survival environments to novel scenarios, linking internal state dynamics with adaptive decision-making (**Table 1**).

Table 1: **A comparison of EVAAA with existing RL environments**

Environment	Survival Task	Multi-Internal States	Unified Reward	Egocentric View	3D Embodiment	Multimodal	Animal-Like Tasks	Progressive Environment	Novel Task Generation	Customizable
Malmo [31]	✓	×	×	×	✓	×	×	×	×	✓
Neural MMO [33]	✓	×	×	×	×	×	×	×	×	✓
Animal-AI Olympic [37]	×	×	×	✓	✓	×	✓	×	×	✓
MineRL [32]	✓	×	×	✓	✓	×	×	×	×	✓
Crafter [30]	✓	✓	✓	×	×	×	×	×	×	×
Avalon [34]	✓	×	✓	✓	✓	×	×	✓	×	✓
O-PIAGETS [39]	×	×	×	✓	✓	×	✓	×	×	×
GIBSONA [40]	×	×	×	✓	✓	×	✓	×	×	✓
EVAAA (ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

4 EVAAA benchmark overview

EVAAA is a simulation platform for studying the autonomous and adaptive behaviors of an agent in dynamically changing environments. Built with Unity and ML-Agents, it integrates RL with a biologically inspired agent design, characterized by multimodal sensory and interoceptive perception (**Fig. 2a**). The platform supports full training pipelines and allows for rich environmental customization, enabling experiments on the regulation of internal state variables, exteroceptive and interoceptive perception, decision-making, and embodied cognition.

4.1 Unity interface

Simulation Environment. The environment is highly customizable, with configurable components such as terrain types, object generation and a temperature system. A complete description of all environmental entities is provided in **Appendix A** of the supplementary material.

Unity Machine Learning Agents Toolkit. Our simulation environment is built on the Unity ML-Agents Toolkit, an open-source platform developed by Unity Technologies for training agents using RL and imitation learning.Apache 2.0 License, which permits use, modification, and distribution under standard open-source terms.

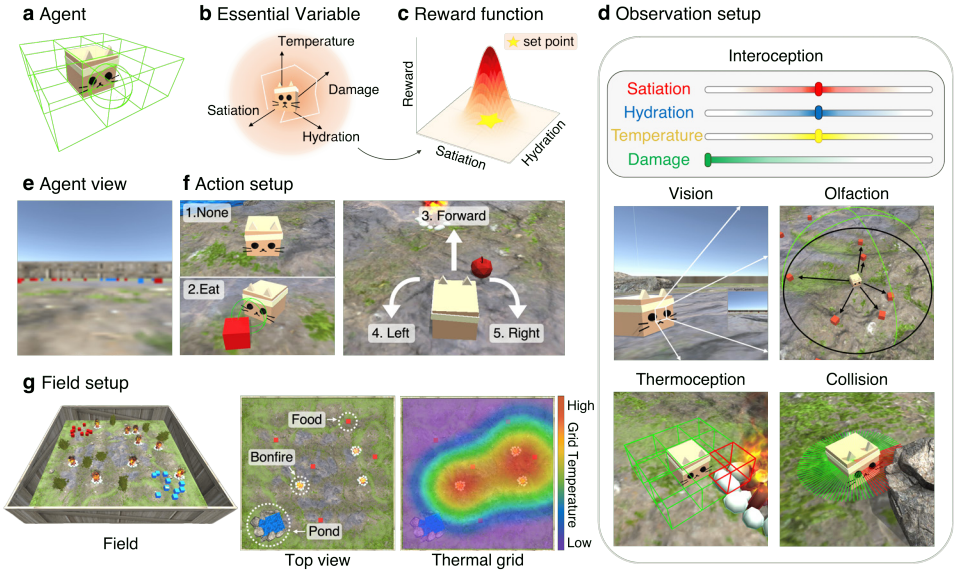

Figure 2: **EVAAA agent design and environmental components.** (a) Agent interaction and thermoception zones, illustrating the spatial regions within which the agent can interact with environmental objects and perceive temperature. (b) Internal state with four essential variables: satiation, hydration, temperature, and damage. (c) Reward is computed based on deviation from homeostatic set points. (d) Interoception requires regulating four EVs: satiation, hydration, temperature (–15 to 15, a set point at 0), and damage (0 to 100, a set point at 0). All EVs at setpoints grant maximum reward. Multimodal exteroceptive and interoceptive observations include vision, a thermal sensing grid, olfaction, and raycasts for collision detection. (e) Egocentric RGB (red, blue, green) vision from a first-person camera. (f) Discrete action set: none, eat, move forward, turn left/right. (g) Sample environment with 3D layout (left), top-down view of resource locations (middle), and a smoothed thermal field (right) reflecting the bonfire heat sources and cooling effects of the pond.

4.2 Agent settings

Essential Variables. The agent maintains four internal variables—satiation, hydration, temperature, and damage—that evolve through interaction with external environments (**Fig. 2b**). An interoceptive reward signal is computed based on deviation of these variables from predefined set points (**Fig. 2c**), allowing goals to emerge from internal dynamics rather than task-specific rewards. The full reward formulation and EV dynamics are detailed in **Appendices C and D** of the supplementary material.

Observations. The agent receives multimodal sensory inputs across both interoceptive and exteroceptive channels, enabling it to respond to its internal state as well as to the external environment (**Fig. 2d**). Detailed descriptions of each observation modality are provided in **Appendix D** of the supplementary material.

- **Interoception:** The agent continuously monitors its internal physiological state through its EVs. These variables change dynamically in response to actions or inaction. For example, consuming food cubes increases the satiation level, while remaining inactive leads to a gradual decline in satiation.
- **Vision:** At every timestep, the agent receives egocentric visual input rendered from a monocular perspective (**Fig. 2e**).
- **Olfaction:** 10-dimensional chemical vectors are aggregated from nearby resources via distance-weighted sampling.
- **Thermoception:** The agent is embedded within a 3×3 grid of thermal sensors that sample the surrounding field temperature.
- **Collision detection:** 100 radial rays detect contact with obstacles and threat (i,e., predator) across a 360 degree field.

Actions. The agent selects from five discrete actions: 'no action', 'move forward', 'turn left', 'turn right', and 'eat'. Eating is triggered when a resource enters an invisible spherical interaction zone positioned in front of the agent (**Fig. 2f**).

4.3 Environment settings

Field setup. The environment is a 100 × 100 unit field with customizable terrain, including food, water, thermal sources, obstacles, and predators (**Fig. 2g**).

Resources. Food and water vary in visual form and spatial distribution across levels (e.g., red/blue cubes vs. apples/ponds). Food respawns randomly, while the pond remains fixed.

Thermal grid. A dynamic 100 × 100 temperature grid simulates spatial heat gradients via heat-emitting objects, such as bonfires, using Gaussian smoothing (**Fig. 2g**).

Obstacles. Bushes, rocks, and bonfires are randomly placed in the field and remain fixed during each stage, while walls and trees retain fixed positions across all stages. Walls block vision, and trees act as local landmarks.

Predator. Across stages, the predator transitions through four behavior states—Resting, Searching, Chasing, and Attacking—using Unity's navigation system and inflicts damage upon contact with the agent.

Day/Night cycle. A five-phase light cycle—Day, Sunset, Night, DeepNight, and Dawn—modulates visibility, temperature, and fog, requiring agents to adapt to temporally varying environmental conditions. Full implementation details are provided in **Appendix E** of the supplementary material.

4.4 Configuration

EVAAA offers a flexible configuration system through JSON files, supporting easy customization of environmental and agent parameters without modifying the codebase. This flexibility extends to the agent's reward function, which is modular and supports alternative sparse-reward configurations. Configuration files specifying parameters such as terrain layout, resource distribution, obstacle placement, and dynamic properties are described in **Appendix E** of the supplementary material.

5 Naturalistic training environment design

To scaffold adaptive survival behavior, EVAAA's training environments are structured as a sequence of progressively challenging levels grounded in internal-state regulation (**Fig. 3a**). Each level is designed to introduce distinct behavioral demands—ranging from simple resource acquisition to obstacle navigation and threat (i,e., predator) avoidance. While these levels were initially designed to elicit biologically inspired behaviors under internal-state regulation, they also serve as targeted evaluations of core RL capabilities. For example, Level 3-2 emphasizes exploration by requiring agents to search for dynamically relocated resources in the absence of fixed cues, while Level 3-1 demands spatial navigation and path planning in semi-structured environments. Level 4-1 introduces an adversarial threat that forces agents to plan strategically to avoid a predator while still securing essential resources. Overall, these scenarios extend EVAAA's relevance beyond homeostatic control, providing a diverse evaluation for assessing exploration strategies, planning capabilities, and generalization performance of RL agents. Full specifications across all levels are provided in **Appendix F** of the supplementary material.

Level 1. Basic resource foraging. Agents learn to regulate their consumption of food and water based on internal signals. In Level 1-1, resources are randomly distributed in an open field with variable temperatures, establishing baseline consumption behavior. In Level 1-2, resources are relocated to corners, requiring the agent to learn spatial targeting based on internal need. Both sublevels are obstacle-free to train EV-driven foraging behaviors first.

Level 2. Obstacle-resource mapping. Environmental complexity increases with obstacles. Level 2-1 features bonfires that create localized heat, supporting temperature homeostasis but causing harm if the agent becomes overheated. Rocks and bushes obstruct navigation and inflict damage upon contact. In Level 2-2, consistent layouts (e.g., pond location, food near trees) encourage agents to associate specific cues with essential resources.

Figure 3: EVAAA training curriculum and test environments. (a) Naturalistic training environments: Progressive training levels (Levels 1-4) span from basic resource foraging to environments involving wandering predators and time-varying conditions. These levels are designed to scaffold adaptive behaviors through internal- state regulation under increasingly complex environmental challenges. (b) Unseen experimental testing environments: the testbeds include basic regulation tasks for individual essential variables, as well as advanced novel scenarios (e.g., risk taking, multi-goal planning, etc.) to evaluate generalization and adaptive decision-making in novel contexts.

Level 3. Terrain exploration and navigation. Agents must explore and navigate semi-structured terrain while balancing competing EVs. In Level 3-1, food is located in tree regions partially enclosed by walls, requiring efficient path planning under temperature and damage constraints. In Level 3-2, food appears in only one region per episode, necessitating dynamic search and continual strategy revision as availability shifts.

Level 4. Dynamic threats. Agents must exhibit behavioral flexibility to avoid external threats and adapt to temporal variations. In Level 4-1, a patrolling predator inflicts damage on contact and must be evaded. In Level 4-2, a day-night cycle modulates both temperature and predator behavior requiring time-sensitive planning and adaptation to shifting external demands.

6 Experimental testbed design

While naturalistic training settings are helpful to induce rich survival behavior, quantitatively probing its result (i.e., agent's capacity for autonomy and adaptivity in terms of internal-state-driven control) is rather challenging in this environment with a high degree of freedom. To address this issue, we introduced a suite of controlled testbed environments—each designed to isolate specific decision-making processes and challenges. By systematically manipulating environmental conditions while minimizing potentially irrelevant extrinsic cues, these testbeds enable precise quantification of how agents generalize from a unified internal reward and exhibit adaptive behavior across diverse tasks. Tasks are organized into two levels—(1) Basic Homeostatic Regulation and (2) Advanced Adaptive Skills (**Fig. 3b**). Full task specifications are provided in **Appendix G** of the supplementary material.

6.1 Basic level

Two-resource choice task. Agents must choose between two resources (e.g., food vs. water) based on their essential variable levels, while operating under partial observability created by a wall that physically separates the two resources.

Collision avoidance task. Agents must reach a visible resource while avoiding narrowly spaced obstacles, testing their fine-grained movement control while minimizing damage.

6.2 Advanced level

Risk-taking task. To reach the resource, agents must pass bonfires, which pose a potential temperature risk. Agents must either pass quickly while enduring the heat or cool down before approaching. This task tests anticipatory planning and internal-state-based risk assessment.

Y-maze (spatial navigation). Two resources are placed at each end of a Y-shaped layout separately, requiring the agent to plan the navigation according to dynamically changing internal needs. This task evaluates the agent's ability to monitor internal states, recall resource locations, and flexibly sequence actions.

Goal manipulation under volatile internal-state conditions. A transparent boundary triggers a sudden and involuntary change of EV levels, requiring the agent to rapidly adjust its behavior. Specifically, the agent must detect the internal-state shift and re-prioritize without external cues–testing internal-state-driven goal adjustment.

Multi-goal planning. Agents must infer urgency across multiple EVs and act in a prioritized sequence—testing coordination and sequential regulation of internal states.

Predators with day/night. Agents must autonomously suppress actions during the day to avoid the predator, even when internal needs are high. At night, they should shift to active foraging as the predators reduce pursuit activity. This task tests adaptive, temporally sensitive survival behavior.

7 Evaluation metrics

We evaluate agent performance across three core components: (1) internal-state regulation, (2) learning efficiency during training, and (3) generalization to novel tasks. Metrics are standardized across algorithms and contextualized using human gameplay data. To benchmark algorithmic performance, we implement three representative reinforcement learning algorithms: DQN [41], PPO [42], and DreamerV3 [43]. These span a range of inductive biases—from DQN as a reactive value-based learner, to PPO for its robust on-policy optimization in dynamic settings, to DreamerV3, a model-based agent capable of memory integration and planning under partial observability. Full implementation details and evaluation protocols are provided in **Appendix H** of the supplementary material. In the naturalistic training environment, we measure survival steps per episode as a proxy for adaptive behavior. Agents are trained either incrementally across levels or directly on complex tasks. In the experimental testbed, success is defined differently for each task—typically based on whether agents maintain all internal variables within viable bounds by episode completion. To further contextualize task difficulty and interpretability, we collected gameplay data from eight human participants. Each completed eight training and ten testing tasks (approximately two hours in total). Due to simulation-to-real-time discrepancies (~40 minutes per 10k steps), human participants were trained only until they understood the task rules, then proceeded to the testing phase using the same tasks as the RL agents.

8 Results

We began by validating the learnability of the EVAAA environment using DreamerV3, given its current state-of-the-art status in the field and well-suited for partially observable and complex environments like EVAAA. This preliminary evaluation helped assess the environment's difficulty and scalability. In general, the number of survival steps reflected increasing task difficulty across levels, with performance dropping sharply from Level 3-2 onward (**Fig. 4a, left**), confirming EVAAA as a challenging benchmark for adaptive control. We also compared DreamerV3 to PPO and DQN,

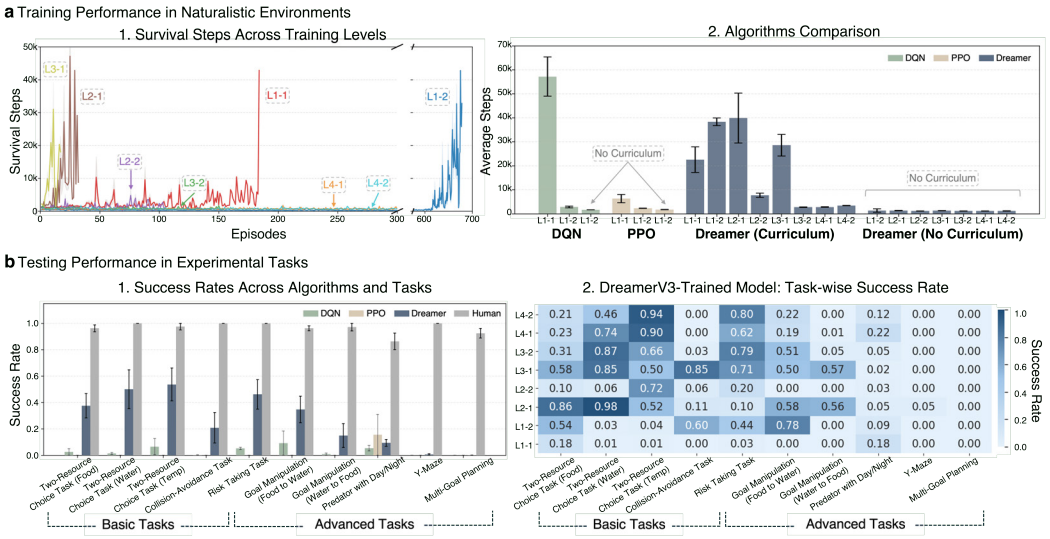

Figure 4: **Agent performance across training and test environments.** (a) Training performance in naturalistic environments. (left) Survival steps across training levels: DreamerV3 learns to survive across levels, but the performance drops beyond Level 3-2, suggesting that EVAAA is a scalable and challenging benchmark for adaptive control. (right) Comparisons of average survival steps among RL algorithms: Agents that undergo the curriculum learning outperform the ones without the curriculum learning; DreamerV3 consistently achieves the highest number of survival steps, followed by DQN and PPO. DQN learns fast in Level 1-1, but fails to generalize beyond Level 1-2. Error bars indicate standard error across random seeds. (b) Testing performance in unseen testbeds. (left) Human participants achieve near-ceiling performance, outperforming all RL agents. DreamerV3 performs best among agents but struggles in certain tasks. Error bars show standard error across training levels (agents) and subjects (humans). (right) Higher training difficulty does not lead to consistent gains in generalization.

with and without curriculum. Results showed that DreamerV3 consistently achieved the highest number of survival steps (**Fig. 4a, right**), benefiting from its use of planning and memory under partial observability. PPO and DQN demonstrated successful survival only in Level 1-1 and failed to scale. Notably, DQN converged faster than DreamerV3 in Level 1-1, but failed to generalize. Supplementary results from additional PPO+LSTM training are provided in **Appendix I**.

To contextualize agent performance, we collected human gameplay data from eight participants. As shown in the left panel of **Fig. 4b**, human participants achieved near-ceiling performance across all testing tasks and outperformed all RL agents. This highlights that there is still a significant gap between the state-of-the-art model-based RL and human performance, and more importantly, the utility of EVAAA to develop more intelligent survival-centered algorithms such as the interoceptive AI [44] or homeostatic RL. While DreamerV3 outperforms other baselines across experimental testing tasks, training on increasingly difficult levels does not lead to monotonic improvements in generalization. For example, tasks such as the two-resource choice and the collision avoidance did not exhibit consistent performance gain with increased training complexity (**Fig. 4b, right**). These findings indicate that increasing task difficulty or training complexity alone is not sufficient to achieve robust, consistently adaptive behavior. Thus, EVAAA is more than a benchmark for task success, in that it enables systematic evaluation of how agents adapt, where they fail to generalize, and when behavioral stability degrades under novel, internally driven demands.

9 Emergent behaviors

During evaluation in the experimental test environments, we observe a range of emergent behaviors as agents attempt to solve each task. In the two-resource choice task, agents trained with easily visible, randomized resource placement (e.g., Level 1-1) failed to search beyond their visual range, unlike

others that explored actively. After encountering Level 3-2, they adopted self-terminating behavior following unsuccessful exploration and generalized this sub-optimal policy to other tasks, such as the Y-maze.

To evaluate the relative importance of each modality, we conducted an additional ablation study assessing the impact of their removal on learning (see **Appendix I**). The findings revealed that vision and the essential variables together constitute the most critical features for successful learning, while the importance of non-visual modalities varied—thermoception and collision were more impactful than olfaction. These results underscore the need for further detailed analyses of the distinct functional roles of different modalities in the agent's learning and behavior.

10 Conclusion and limitations

EVAAA is a biologically inspired, virtual 3D ego-centric environment and simulation benchmark designed to place internal-state regulation rather than relying on externally defined rewards. This intrinsic reward formulation supports unified internal reward design , and by integrating egocentric embodiment, multimodal perception, EVAAA bridges gaps between ecological validity models of adaptive behavior and existing RL frameworks. EVAAA comprises a curriculum-based naturalistic training environment and a suite of controlled experimental testbeds, enabling systematic evaluation of learning, generalization, and adaptivity under varying internal and external constraints. Across the structured environments, we demonstrated that RL agents, particularly DreamerV3, were able to acquire robust survival behaviors, but generalization to novel tasks remains somewhat inconsistent and warrants further investigation. In this sense, EVAAA serves as both a benchmark and an analytical tool for understanding RL agents' adaptive behavioral patterns. Looking ahead, EVAAA's architecture was deliberately designed to support the addition of richer, higher-order task types beyond the current version. Because EVAAA is built on a modular Unity ML-Agents framework, it can readily accommodate advanced training and testing settings, such as multi-agent scenarios for studying cooperation or competition (i.e., social tasks), memory-dependent reasoning (e.g., recalling resource locations across changing terrains), and inference-based tasks (e.g., inferring predators' future locations). These future expansions will extend EVAAA's focus from survival to more complex cognitive behaviors, linking internal-state regulation to advanced capabilities such as social functioning, memory-based planning, and context-sensitive decision-making.

Limitations. The limitation of the current work mainly includes the relative paucity of tasks targeting higher-order cognitive functions, such as planning, inference, and social reasoning, and reliance on only fixed (non-dynamical) homeostatic set points. Future extensions incorporating interdependent internal state variables and richer agent embodiments, such as adaptive set points, would enhance biological realism and enable deeper analyses of intrinsic motivation and flexible, biologically inspired decision-making in artificial agents. Also, while EVAAA currently uses a simplified agent with a basic action space (move forward, turn, eat) to ensure reproducibility and training stability, future iterations will integrate more complex Unity ML-Agents embodiments such as the Crawler quadruped—a creature with four arms and four forearms—while retaining the simpler agent as a reproducible baseline.

Acknowledgements

This work was supported by IBS-R015-D2 (Institute for Basic Science, South Korea; to C.-W.W. and S.-J.H) as well as by RS-2023-00217361 (Ministry of Science and ICT — Basic Research Laboratory, South Korea; to S.-J. H).

References

[1] Ranasinghe, K., Sabatini, R., Gardi, A., Bijjahalli, S., Kapoor, R., Fahey, T., & Thangavel, K. (2022). Advances in Integrated System Health Management for mission-essential and safety-critical aerospace applications. Progress in Aerospace Sciences, 128, 100758.

[2] Gao, Y., & Chien, S. (2017). Review on space robotics: Toward top-level science through space exploration. Science Robotics, 2(7), eaan5074.

[3] Ashby, W. R. (1952). Design for a brain. Wiley.

[4] McFarland, D., & Spier, E. (1997). Basic cycles, utility and opportunism in self-sufficient robots. Robotics and Autonomous Systems, 20(2–4), 179–190.

[5] Avila-Garcia, O., & Cañamero, L. (2004). Using hormonal feedback to modulate action selection in a competitive scenario. From animals to animats, 8, 243–252.

[6] Lones, J., Lewis, M., & Cañamero, L. (2017). A hormone-driven epigenetic mechanism for adaptation in autonomous robots. IEEE Transactions on Cognitive and Developmental Systems, 10(2), 445–454.

[7] Yoshida, N. (2017). Homeostatic agent for general environment. Journal of Artificial General Intelligence, 8(1), 1.

[8] Corrales-Carvajal, V. M., Faisal, A. A., & Ribeiro, C. (2016). Internal states drive nutrient homeostasis by modulating exploration-exploitation trade-off. Elife, 5, e19920.

[9] Sgammeglia, N., & Sprecher, S. G. (2022). Interplay between metabolic energy regulation and memory pathways in Drosophila. Trends in Neurosciences, 45(7), 539–549.

[10] Nässel, D. R., & Zandawala, M. (2022). Endocrine cybernetics: neuropeptides as molecular switches in behavioural decisions. Open Biology, 12(7), 220174.

[11] Teckentrup, V., & Kroemer, N. B. (2024). Mechanisms for survival: Vagal control of goal-directed behavior. Trends in Cognitive Sciences, 28(3), 237–251.

[12] Doya, K. (2002). Metalearning and neuromodulation. Neural networks, 15(4–6), 495–506.

[13] Lee, S., Liebana, S., Clopath, C., & Dabney, W. (2024). Lifelong Reinforcement Learning via Neuromodulation. arXiv preprint arXiv:2408.08446.

[14] Luketina, J., Flennerhag, S., Schroecker, Y., Abel, D., Zahavy, T., & Singh, S. (2022, November). Meta-gradients in non-stationary environments. In Conference on Lifelong Learning Agents (pp. 886–901). PMLR.

[15] Xu, Z., van Hasselt, H. P., & Silver, D. (2018). Meta-gradient reinforcement learning. Advances in neural information processing systems, 31.

[16] Xu, Z., van Hasselt, H. P., Hessel, M., Oh, J., Singh, S., & Silver, D. (2020). Meta-gradient reinforcement learning with an objective discovered online. Advances in Neural Information Processing Systems, 33, 15254–15264.

[17] Flennerhag, S., Schroecker, Y., Zahavy, T., van Hasselt, H., Silver, D., & Singh, S. (2021). Bootstrapped meta-learning. arXiv preprint arXiv:2109.04504.

[18] Todorov, E., Erez, T., & Tassa, Y. (2012, October). Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ international conference on intelligent robots and systems (pp. 5026–5033). IEEE.

[19] Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., ... & Petersen, S. (2016). Deepmind lab. arXiv preprint arXiv:1612.03801.

[20] Gan, C., Schwartz, J., Alter, S., Mrowca, D., Schrimpf, M., Traer, J., ... & Yamins, D. L. (2020). Threedworld: A platform for interactive multi-modal physical simulation. arXiv preprint arXiv:2007.04954.

[21] Chentanez, N., Barto, A., & Singh, S. (2004). Intrinsically motivated reinforcement learning. Advances in neural information processing systems, 17.

[22] Barto, A. G., Singh, S., & Chentanez, N. (2004, October). Intrinsically motivated learning of hierarchical collections of skills. In Proceedings of the 3rd International Conference on Development and Learning (Vol. 112, p. 19).

[23] Singh, S., Lewis, R. L., Barto, A. G., & Sorg, J. (2010). Intrinsically motivated reinforcement learning: An evolutionary perspective. IEEE Transactions on Autonomous Mental Development, 2(2), 70–82.

[24] Barto, A. G. (2012). Intrinsic motivation and reinforcement learning. In Intrinsically motivated learning in natural and artificial systems (pp. 17–47). Berlin, Heidelberg: Springer Berlin Heidelberg.

[25] Keramati, M., & Gutkin, B. (2011). A reinforcement learning theory for homeostatic regulation. Advances in neural information processing systems, 24.

[26] Keramati, M., & Gutkin, B. (2014). Homeostatic reinforcement learning for integrating reward collection and physiological stability. Elife, 3, e04811.

[27] Yoshida, N., Daikoku, T., Nagai, Y., & Kuniyoshi, Y. (2024). Emergence of integrated behaviors through direct optimization for homeostasis. Neural Networks, 177, 106379.

[28] Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., & Savarese, S. (2018). Gibson env: Real-world perception for embodied agents. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 9068-9079).

[29] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., ... & Batra, D. (2019). Habitat: A platform for embodied ai research. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 9339–9347).

[30] Hafner, D. (2021). Benchmarking the spectrum of agent capabilities. arXiv preprint arXiv:2109.06780.

[31] Johnson, M., Hofmann, K., Hutton, T., & Bignell, D. (2016, July). The Malmo Platform for Artificial Intelligence Experimentation. In Ijcai (Vol. 16, pp. 4246–4247).

[32] Milani, S., Topin, N., Houghton, B., Guss, W. H., Mohanty, S. P., Vinyals, O., & Kuno, N. S. (2020). The minerl competition on sample-efficient reinforcement learning using human priors: A retrospective. Journal of Machine Learning Research, 1, 1–10.

[33] Suarez, J., Du, Y., Isola, P., & Mordatch, I. (2019). Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents. arXiv preprint arXiv:1903.00784.

[34] Albrecht, J., Fetterman, A., Fogelman, B., Kitanidis, E., Wróblewski, B., Seo, N., ... & Qiu, K. (2022). Avalon: A benchmark for rl generalization using procedurally generated worlds. Advances in Neural Information Processing Systems, 35, 12813–12825.

[35] Tang, X., Li, J., Liang, Y., Zhu, S. C., Zhang, M., & Zheng, Z. (2024). Mars: Situated Inductive Reasoning in an Open-World Environment. Advances in Neural Information Processing Systems, 37, 17830–17869.

[36] Xu, M., Jiang, G., Liang, W., Zhang, C., & Zhu, Y. (2023). Active reasoning in an open-world environment. Advances in Neural Information Processing Systems, 36, 11716–11736.

[37] Crosby, M., Beyret, B., & Halina, M. (2019). The animal-ai olympics. Nature Machine Intelligence, 1(5), 257–257.

[38] Voudouris, K., Slater, B., Cheke, L. G., Schellaert, W., Hernández-Orallo, J., Halina, M., ... & Crosby, M. (2025). The Animal-AI Environment: A virtual laboratory for comparative cognition and artificial intelligence research. Behavior Research Methods, 57(4), 107.

[39] Voudouris, K., Donnelly, N., Rutar, D., Burnell, R., Burden, J., Hernández-Orallo, J., & Cheke, L. G. (2022). Evaluating object permanence in embodied agents using the Animal-AI environment.

[40] Rutar, D., Cheke, L. G., Hernández-Orallo, J., Markelius, A., & Schellaert, W. (2024). General interaction battery: Simple object navigation and affordances (GIBSONA). Available at SSRN 4924246.

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

[42] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[43] Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2025). Mastering diverse control tasks through world models. Nature, 1-7.

[44] Lee, S., Oh, Y., An, H., Yoon, H., Friston, K. J., Hong, S. J., & Woo, C. W. (2023). Life-inspired interoceptive artificial intelligence for autonomous and adaptive agents. arXiv preprint arXiv:2309.05999.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly articulated the major contributions of the current study, including (1) the introduction of EVAAA as a simulation environment centered on internal-state regulation, (2) the integration of multimodal egocentric perception, and (3) the design of structured, incrementally complex training environments alongside controlled testbeds for evaluation.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses its limitations in the "Conclusion and Limitations" section (**Section 10**). These include the current lack of tasks targeting higher-order cognitive functions (e.g., planning, inference, social reasoning) and the reliance on fixed, non-dynamic homeostatic set points. It also suggests future research directions, such as incorporating adaptive set points and interdependent internal variables to enhance biological realism.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not present formal theoretical results or mathematical proofs.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides detailed descriptions of agent architecture including its essential variables and exteroceptive observations, environmental settings and configuration, testbed design, and evaluation metrics through **Sections 4-7** and **Appendix E**.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available through a link provided in the abstract. The paper and code repository provide sufficient documentation and configuration instructions in **Appendix E**.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed specifications for training/test environment structure, agent hyperparameters, and configurations for DQN, PPO, and DreamerV3 are provided in **Sections 5-7** and **Appendix H**.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The paper reports key performance metrics (i.e., survival steps and success rates) along with appropriate measures of variability (e.g., confidence intervals or standard deviations), as visualized in the plots in **Figure 4**.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: **Appendix H** provides detailed information about the computational resources used, including GPU and CPU specifications, memory requirements, and execution times. This allows readers to estimate and replicate the compute demands necessary to reproduce the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This study fully adheres to the NeurIPS Code of Ethics. It involves no ethically sensitive experiments or data and ensures transparency and reproducibility as described throughout the paper.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification:The paper discusses broader societal impacts in **Appendix J**, including both positive (e.g., promoting the development of more autonomous and adaptive agents) and potential risks (e.g., misuse that may cause unintended harm).

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The environments and models described in this paper do not involve high-risk applications or sensitive data. As such, no specific safeguards were necessary or applicable.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The Unity ML-Agents Toolkit is used in accordance with its Apache 2.0 license, as noted in **Section 4**. In addition, all prefab assets used in EVAAA are sourced from the Unity Asset Store and licensed under the Standard Unity Asset Store EULA, allowing royalty-free use for both commercial and non-commercial purposes. This licensing details are described in **Appendix A**.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The newly developed EVAAA simulation environment and configurations are thoroughly documented with comprehensive guidelines included in the code repository and the **Appendix E**.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes]

Justification: Full details of the human-subject experimental procedures, including participant instructions, compensation rates, and session duration, are described in **Section 7** and **Appendix H**.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: The human-subject experiment included in the paper received approval from the Institutional Review Board (IRB). Potential risks were minimal, fully disclosed to participants, and appropriately managed in accordance with the approved IRB protocol.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs are not used in any part of the core methodology in this research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Appendix

Content Overview

- **Appendix A** describes the Unity-based interface implemented in EVAAA, including an environment setup, prefab structures, and object instantiation.
- **Appendix B** provides a comprehensive introduction to Essential Variables (EVs), including their design, dynamics, and role in internal state regulation.
- **Appendix C** explains the implementation of the reward system and its connection to the balance of internal states.
- **Appendix D** details the agent's observation system, covering both interoceptive inputs (EVs) and exteroceptive sensors (i.e., vision, olfaction, thermoception, and collision detection).
- **Appendix E** outlines the modular configuration to generate EVAAA environments, along with the instructions for environment customization.
- **Appendix F** presents the structure and progression of naturalistic training environments.
- **Appendix G** describes the design of unseen experimental testbeds for evaluation.
- **Appendix H** details the computational resources, evaluation metrics, and human experiment.
- **Appendix I** provides analyses of agent behavior across training and test environments, including emergent behavioral patterns.
- **Appendix J** discusses broader impacts, including potential applications and ethical considerations of EVAAA.

All code and data are publicly available at: `https://github.com/cocoanlab/evaaa`

B Unity Interface

B.1 Prefabs

Environmental elements such as terrain, resources, obstacles, and predators are implemented as reusable and configurable Unity prefabs. These assets are sourced from the Unity Asset Store, and we selected only freely available assets typically used in both research and applied projects. This modular system enables rapid prototyping, task generation, condition randomization, and reproducible scene setup. Prefabs can be customized through the Unity Editor or programmatically at runtime, and reused across scenes without manual rebuilding. Figure 5 provides a visual overview of the following available prefab types:

- **Agent** prefabs include both the main agent and predators.
- **Environment** prefabs define the simulation area based on terrain fields.
- **Material** prefabs offer color customization for visual or functional distinction and can be applied to any object type. Available colors include black, red, light green, yellow, orange, and more.

C Essential Variables

Essential Variables (EVs) represent the agent's internal state and are central to the design of EVAAA. Four scalar EVs are defined: satiation, hydration, body temperature, and tissue integrity (the level of damage). These variables evolve over time based on the agent's actions and environmental conditions, and are continuously updated during simulation. EVs influence the agent's observation vector, reward computation, and termination conditions. Over time, even in the absence of activity, levels of satiation and hydration gradually decline, reflecting natural metabolic processes that lead to hunger and dehydration. To continue an episode, all EVs must remain within their viable ranges; deviations lead to negative rewards or early termination. This framework anchors the learning process in survival-relevant dynamics without hard-coded objectives.

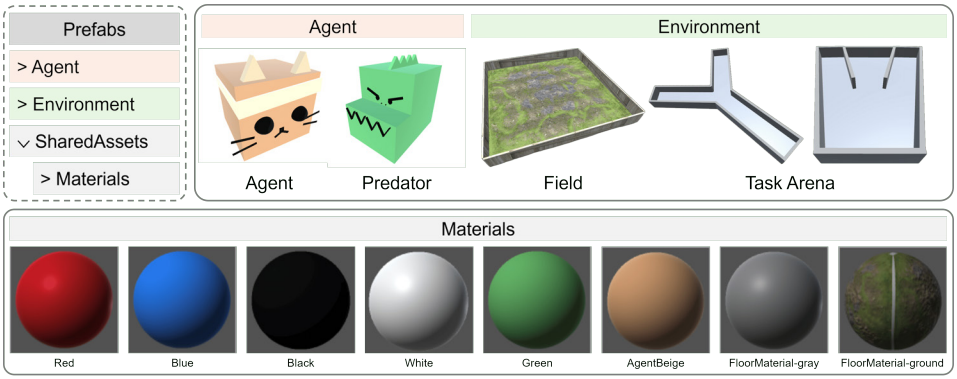

Figure 5: **Overview of prefab categories and object types used in EVAAA.** Prefabs are grouped into Agents, Environment, and Materials. Each category includes reusable components for constructing and customizing interactive scenes: Agents (main agent and predators), Environment (terrain and containers), and Materials (varied textures and colors for visual distinction).

D Reward system

The reward system in EVAAA is driven by the agent's internal state (EVs), rather than relying on task-specific external objectives. At each timestep, the agent receives a reward signal reflecting how closely its internal state matches predefined homeostatic setpoints. The reward r_t is calculated as the negative normalized Euclidean distance between the current EVs and their target values:

$$r_t = -\sqrt{\sum_{i=1}^{4} \left(\frac{EV_i(t) - \mu_i}{\sigma_i} \right)^2}$$

Where:

- $EV_i(t)$ is the value of the i-th essential variable at time t.
- μ_i is the setpoint (target value) (e.g., 0 for satiation/hydration/temperature/damage)
- σ_i is the allowed deviation, derived from researcher-defined valid ranges for each EV

The negative reward increases with greater deviations from the setpoints, encouraging the agent to get back to the balanced internal states (i.e., homeostasis). For instance, if the agent becomes dehydrated, overheated, or injured, it receives increasingly negative rewards.

E EVAAA Observation details

E.1 Interoception

I. EV update dynamics Each EV is modeled as a scalar internal state $\text{EV}_i \in \mathbb{R}$ and updated at every timestep using a linear function: $\text{EV}_i(t+1) = \text{EV}_i(t) + \beta_0 \cdot D_i + \sum_{j=1}^{3} \beta_j \cdot S_{ij}(t) + \beta_{i,e} \cdot \Delta_i(t)$

Term Definitions:

- $\beta_0 \cdot D_i$: constant passive decay scaled to the EV's range
- $S_{ij}(t)$: EVs input
- β_j: coefficients for inter-EV influences (e.g., $\text{foodCoefficient.change}_j$)
- $\beta_{i,e} \cdot \Delta_i(t)$: discrete event-driven changes from resource interaction (e.g., eating food, drinking water)

Although the current setting has no interdependency among EVs, non-linear interactions among EVs are implemented as one of the potential configurations for future use.

II. EV ranges and setpoints Each EV operates within a researcher-defined viable range and is anchored to a homeostatic set point representing the agent's optimal internal state.

- Satiation, hydration, and temperature use symmetric ranges from -15 to 15, with a setpoint at 0. Deviations in either direction indicate homeostatic imbalance (e.g., overheating or starvation).
- Damage operates on a 0 to 100 scale, with a setpoint at 0 representing no damage. Higher values indicate increased damage resulting from environmental hazards.

An episode is terminated if any EV falls outside its defined range.

E.2 Vision

At each timestep, the agent receives egocentric visual input rendered from a first-person monocular view using Unity ML-agents' built-in CameraSensor function. Observations are encoded as 64x64 RGB images by default, represented in a 3D-tensor format (height, width, channels). Resolution and color depth are configurable.

E.3 Olfaction

Each time a resource object is instantiated—whether at the beginning of an episode or after being consumed and respawned—it is assigned a fixed 10-dimensional feature vector that encodes its identity (e.g., food vs water), with additional variation introduced through uniform random sampling (**Fig. 6**). These vectors remain constant throughout the episode. The agent uses a spherical olfactory sensor based on Unity's Physics OverlapSphere to detect nearby resources within a configurable radius. For each detected object, its feature vector is divided by its distance to the agent. If multiple objects are detected (e.g., several food sources at distances d_1, d_2, d_3, and d_4), their distance-weighted vectors are summed to make a single 10-dimensional observation at each timestep. This process informs the agent of the existence of nearby resources without revealing their exact positions.

E.4 Thermoception

The agent perceives ambient temperature through a 3x3 grid thermal sensor, with the agent located at the center (**Fig. 7**). These sensors sample values from an invisible two-dimensional 100x100 temperature grid that overlays the environment. Each cell encodes a scalar temperature value but is not directly observable. At each timestep, if a sensor intersects with a grid cell, it records the corresponding temperature value. The nine sensors together produce a 9-dimensional observation vector that captures the spatial temperature distribution around the agent. This enables detection of heat gradients and environmental temperature changes without relying on explicit spatial coordinates. Although the temperature field remains static during an episode, it can be reconfigured to simulate dynamic conditions such as day-night cycles or local heat sources (e.g., bonfires).

E.5 Collision Setting

The agent is equipped with a radial raycasting system that emits 100 evenly spaced rays across a full 360-degree field around its body (**Fig. 8**). These rays are divided into 10 angular segments, with each segment aggregating binary collision feedback. At every step, rays are cast outward from the agent's position and are checked for intersections with nearby objects, excluding the agent itself. If any ray within a segment collides with an obstacle or predator, the corresponding entry in a 10-dimensional 'collisionObservation' vector is set to a non-zero value. This setup allows the agent to detect nearby hazards and avoid collisions during navigation. Upon contact, the agent receives a scalar damage value proportional to the impulse magnitude of the collision. This value is capped at a predefined maximum to prevent extreme updates and then subtracted from the agent's internal damage EV.

F EVAAA implementation details

EVAAA provides a modular configuration system using structured JSON files, allowing researchers to flexibly adjust agent properties, environmental layout, resource generation, and interaction parameters.

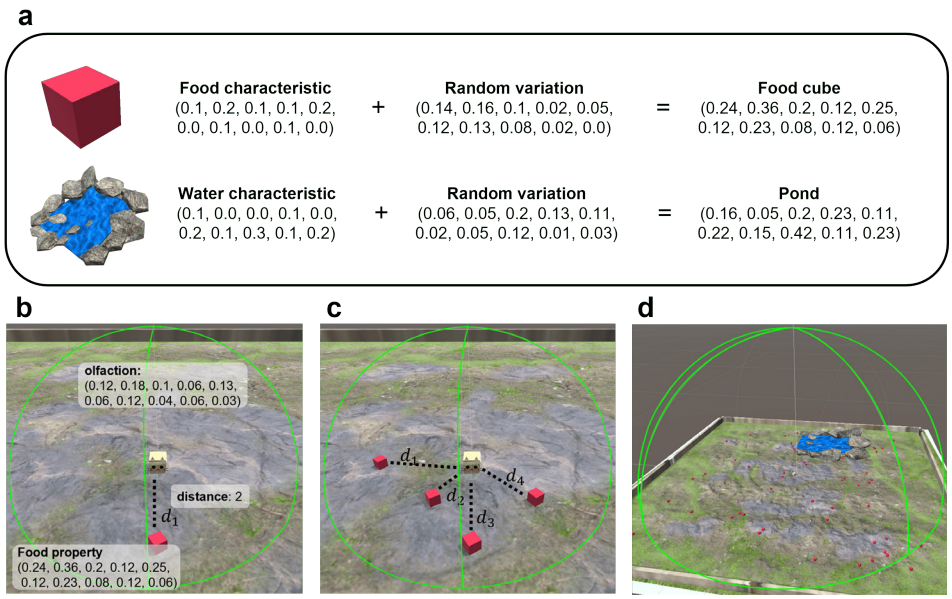

Figure 6: **Overview of the olfactory mechanism.** (a) Each resource object emits a unique 10-dimensional chemical feature vector. (b) The agent senses nearby objects using a spherical detection field. (c) Multiple resources are encoded through the distance-dependent aggregation of their feature vectors. (d) The olfactory radius can be extended to encompass larger areas and heterogeneous resource types.

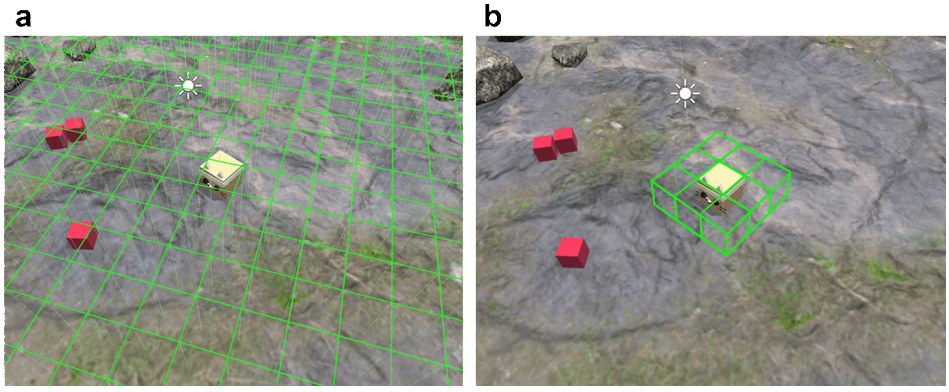

Figure 7: **Overview of the thermoception mechanism.** (a) A 100x100 invisible temperature grid is placed over the terrain. (b) The agent samples local temperatures via a 3×3 array of thermal sensors centered on its position. Each sensor maps to overlapping grid cells, producing a 9-D temperature observation at each timestep.

All configuration files are grouped under the Config directory and organized into reusable components, supporting both training and evaluation through a unified structure (**Fig. 9**). Core configuration categories include:

- E.1 **Agent config** defines action settings (i.e., movement speed, eating action), internal state initialization, and sensor modalities.
- E.2 **Court config** controls terrain size, boundary settings, and floor materials.
- E.3 **Resource config** specifies food/water generation, spawn intervals, and resource placement behaviors (e.g., random distribution, static positioning, respawn strategy).
- E.4 **ThermoGrid config** sets the resolution and values of the ambient temperature field.

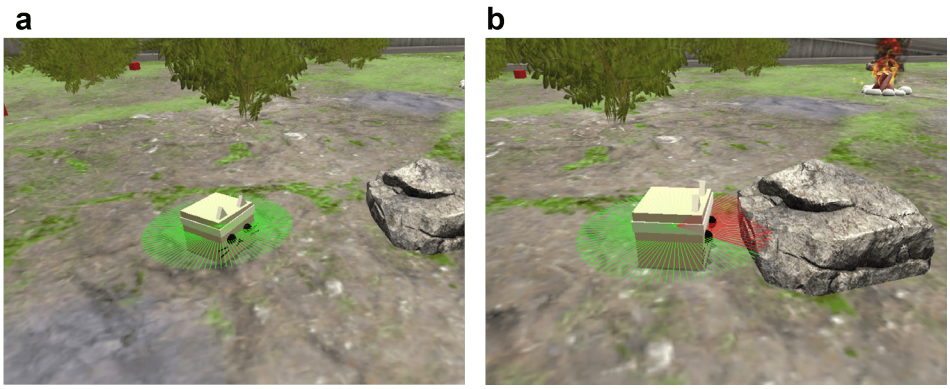

Figure 8: **Collision detection using radial raycasting.** (a) In obstacle-free conditions, rays (green) return no contact. (b) When rays intersect an object (e.g., a rock), contact is detected (red), updating the agent's 10-D collision observation vector.

- E.5 **Obstacle config** determines the type and placement of physical obstacles.
- E.6 **Landmark config** defines the valid boundary for predators to navigate.
- E.7 **Predator config** configures predator behavior, aggression levels, and spawn patterns.
- E.8 **Day/Night cycle config** manages time-of-day cycles and lighting conditions.

Each subsection below details the available parameters, default values, and example use cases for configuring EVAAA environments.

Figure 9: **Overview of the configuration system of EVAAA.** (a) Modular config hierarchy, organized by components such as agent, court, resource, and obstacle. (b) Example instantiations for training (e.g., train-level-1.1, train-level-2.1) and testing (e.g., exp-Ymaze, exp-damage) via the configFolderName parameter.

F.1 Agent config

The agent configuration defines how the agent moves, senses, and maintains internal homeostasis (**Fig. 10**). Each EV has a bounded range and an initial value, adjustable through parameters like 'startFoodLevel'. Consuming resources increases the corresponding EV based on values such as 'resourceFoodValue', 'resourceWaterValue', and so on. Movement is configured using 'moveSpeed' and

'turnSpeed', and the agent's initial position and orientation are set using 'randomPositionRange' and 'initAgentAngle'. The 'autoEat' true/false flags control whether the agent eats automatically when near a resource, and 'eatingDistance' specifies the required proximity. Sensory modalities—touch, collision, olfactory, and thermal—can be toggled individually, with 'olfactorySensorLength' controlling olfactory range. Collision-related settings include sensor range and 'damageConstant', which affects how damage is applied upon impact. EV decay rates are determined by coefficients such as 'foodCoefficient', which define how quickly each EV decreases over time.

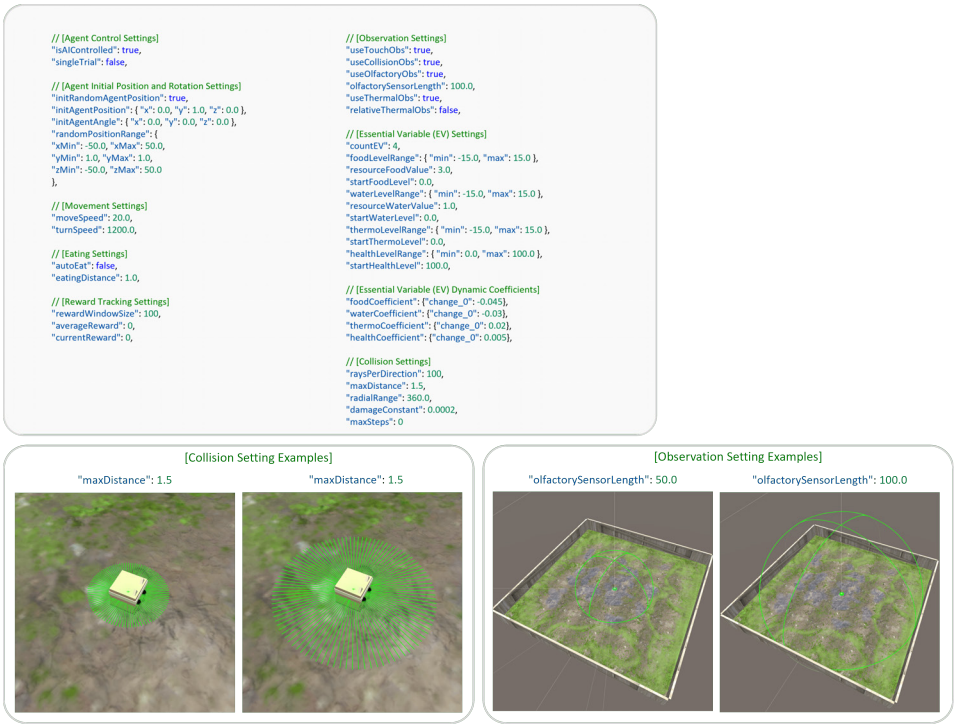

Figure 10: **Agent configuration parameters in EVAAA.** Top: JSON snippets illustrating control, movement, eating, essential variable (EV) decay, observation, and collision settings. Bottom: Visualization of selected parameters, including collision sensing range (maxDistance) and olfactory sensor range (olfactorySensorLength). These settings govern how agents perceive the environment and manage internal state variables.

F.2 Court config

The court configuration defines the size, layout, and visual appearance of the simulation field (**Fig. 11**). By default, the field is a 100x100 unit square centered at the origin, with coordinates ranging from -50 to 50 along the x and z axes. It acts as the base area for placing all environmental items, including resources, thermal grids, obstacles (e.g., rocks, trees, bonfires), and predators. Wall height ("wallHeight") and surface materials (e.g., "floorMaterialName", "wallMaterialName") are configurable, allowing environments to range from naturalistic (e.g., grassy textures) to controlled lab-like settings.

F.3 Resource config

The resource configuration file defines the placement, quantity, and appearance of interactive resources such as food and water (**Fig. 12**). Each resource group is assigned a 'prefabName', a spawn count, and randomized value ranges for position, rotation ('rotationRange'), and scale ('scaleRange'). In early training stages, resources are instantiated as simple red or blue cubes placed randomly. As the environment becomes more complex, resource shapes diversify into more realistic forms like apples and ponds, and their placement becomes more structured—for example, grouped apples or

fixed-location ponds. Resource types are categorized as ‘Random’, ‘Static’, or ‘GroupedRandom’, which control spatial behavior and respawn logic.

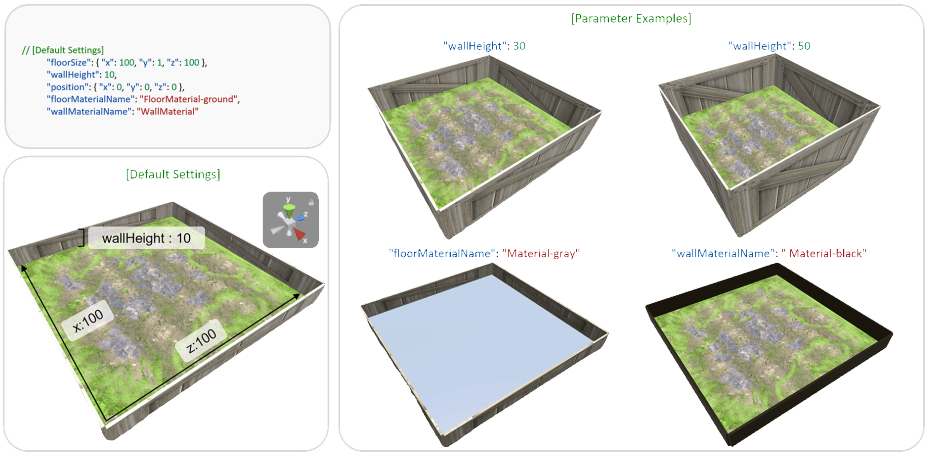

Figure 11: **Court configuration option in EVAAA.** Left: Default court settings define a 100×100 unit area with 10-unit high walls and a ground-like floor texture. Right: Examples illustrating how modifying wallHeight, floorMaterialName, and wallMaterialName changes the spatial and visual characteristics of the court. These parameters enable customization of court geometry and appearance for controlled or naturalistic experimental setups.

F.4 ThermoGrid config

The thermal environment in EVAAA is modeled using a dynamic, grid-based temperature system. The grid consists of a configurable number of square cells that span the full simulation area (**Fig. 13**), with each cell to which a scalar temperature value is assigned. A global baseline temperature (‘fieldDefaultTemp’) is applied across the grid, while localized variations are introduced via "hotspots". Each hotspot has a configurable temperature (hotSpotTemp), size (hotSpotSize), and count (hotSpotCount), allowing precise control over how much heat is applied, where it is placed, and how widely it spreads. To simulate realistic heat diffusion, a 2D smoothing filter (e.g., Gaussian blur) is applied using the smoothingSigma parameter, generating continuous temperature gradients across the field. Two types of hotspot placement are supported:

- **Random hotspot placement:** Enabled via useRandomHotSpot = true, this method distributes temperature sources randomly across the grid using the predefined hotspot parameters.
- **Object-linked hotspot placement:** Enabled via useObjectHotSpot = true, this mode generates thermal gradients around specific obstacles (e.g., bonfires) that act as heat sources. In this case, the temperature field is informed by the positions and values of these thermal objects rather than using random placement.

F.5 Obstacle config

Obstacles are categorized as either randomized or fixed (**Fig. 14**). When the minimum and maximum values of the position range are the same, the obstacle is placed at a fixed location. Otherwise, it is positioned randomly within the specified range. For example, in Figure 14, Trees and RockWalls are placed at fixed positions. Rocks, Bushes, and Bonfires, by contrast, are located in random locations that change at the start of each episode.

F.6 Landmark config

Landmarks are invisible reference points used to guide predators in setting their destinations. They are generated based on a grid layout defined in a landmark JSON configuration file (i.e., ‘landmarkConfig.json’). This configuration specifies landmark positions using a ‘customPattern’ matrix, where

Figure 12: **Resource configuration options in EVAAA.** Top: JSON examples illustrating how to control resource type, quantity, spatial distribution, and visual scale using parameters like count, position, and scaleRange. Middle: Visualizations showing effects of varying object count (left), object size (center), and placement region for different resource types (right). Bottom: Advanced configuration examples including fixed-position static objects (e.g., "Pond") and grouped dynamic objects (e.g., clustered apples in "FoodGroupA") for structured environmental design.

a value of 1 denotes the presence of a landmark at a corresponding grid cell. It also defines parameters such as the size of each landmark (e.g., 'landmarkRadius'), the spacing between landmarks to avoid overlap (e.g., 'overlapPadding'), and a flattened array (e.g., 'customPattern') indicating the number of landmarks to place in each grid cell. To define the predator's moving area, the system computes a convex hull around all placed landmarks (**Fig. 15**). This hull is then converted into a mesh collider, representing the spatial boundary within which the predator is allowed to move. This method restricts predators to specific areas of the environment while preserving flexibility in design layout.

F.7 Predator config

The predator in EVAAA uses Unity's built-in AI navigation system and operates across four discrete behavioral states: Resting, Searching, Chasing, and Attacking. In the Searching state, the predator moves stochastically using Unity's built-in NavMesh—a navigation system that enables pathfinding over walkable surfaces— and the predator's state changes to the Resting state after a configurable number of steps (**Fig. 16**). When the agent enters the predator's vision cone, defined by an adjustable angle and distance, the predator switches to the Chasing state, targeting the agent's position. On physical contact, it enters the Attacking state and increases the agent's internal damage EV. If the agent escapes or breaks contact, the predator returns to Searching.

F.8 Day/Night cycle config

EVAAA implements a configurable day-night cycle that introduces time-dependent environmental changes. The cycle is implemented by rotating a directional light source in the Unity scene (**Fig. 17**). Each phase adjusts environmental parameters such as lighting intensity, fog density, ambient color, and scene exposure. All relevant settings are defined in the daynightConfig.json, including phase durations, light rotation speed ('dayChangeSpeed'), and temperature effects like 'nightTemperatureChange', which lowers the overall thermal field during night phases. For example, temperature gradually rises

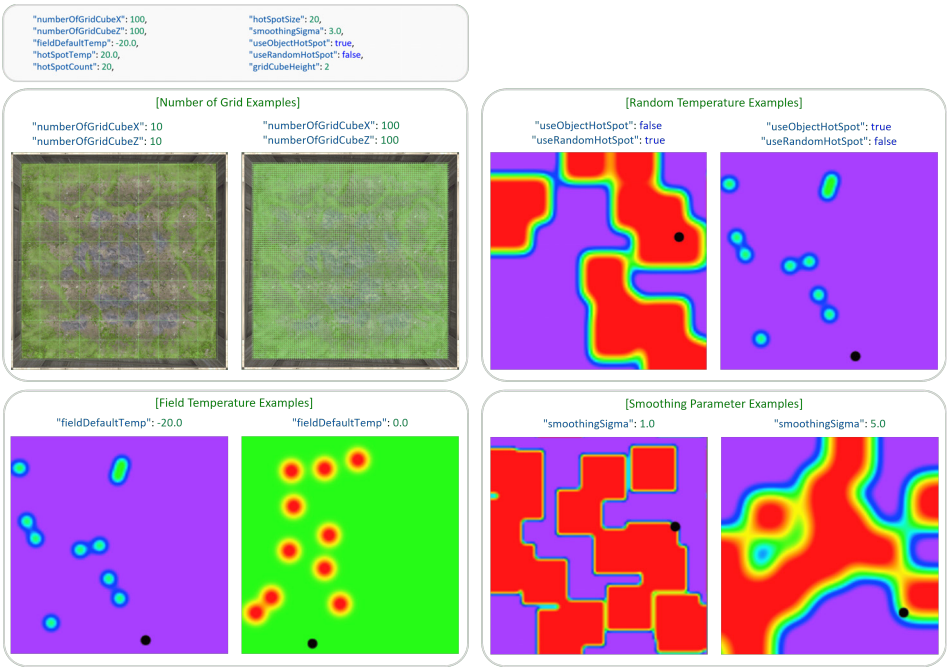

Figure 13: **Thermal field options in EVAAA.** Top left: Grid resolution comparison showing coarse (10x10) and fine (100x100) temperature grids. Top right: Hotspot generation using random placement ('useRandomHotSpot') vs. object-linked placement ('useObjectHotSpot'). Bottom left: Impact of changing default baseline temperature ('fieldDefaultTemp') on the ambient environment. Bottom right: Effect of smoothing parameter ('smoothingSigma') on thermal diffusion, with higher values producing more continuous gradients. Temperature is color-coded from cool (blue) to hot (red).

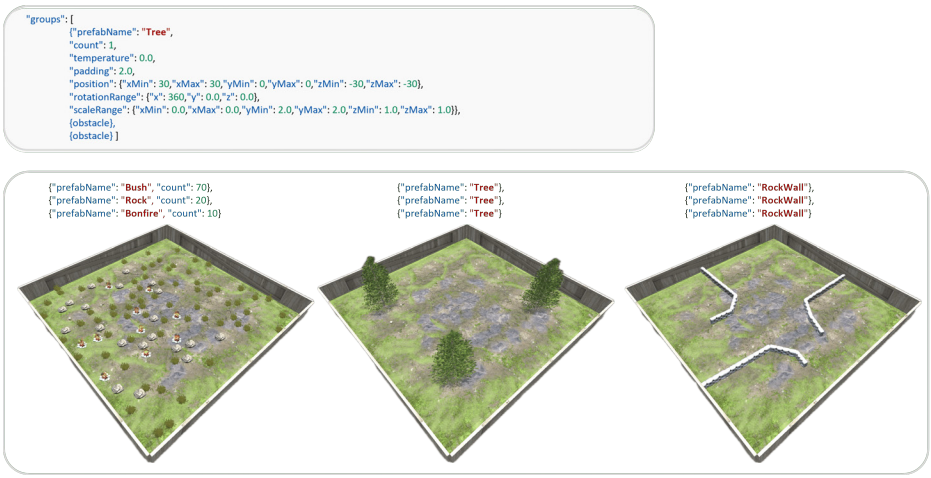

Figure 14: **Obstacle configuration in EVAAA.** Top: JSON configuration specifying parameters for static obstacles such as trees, including position, padding, and scale. Bottom left: Randomized placement of bushes, rocks, and bonfires to create unorganized terrain. Bottom center: Organized (or structured) placement of tree obstacles with consistent spacing. Bottom right: Placement of walls to obstruct navigation or visual perception.

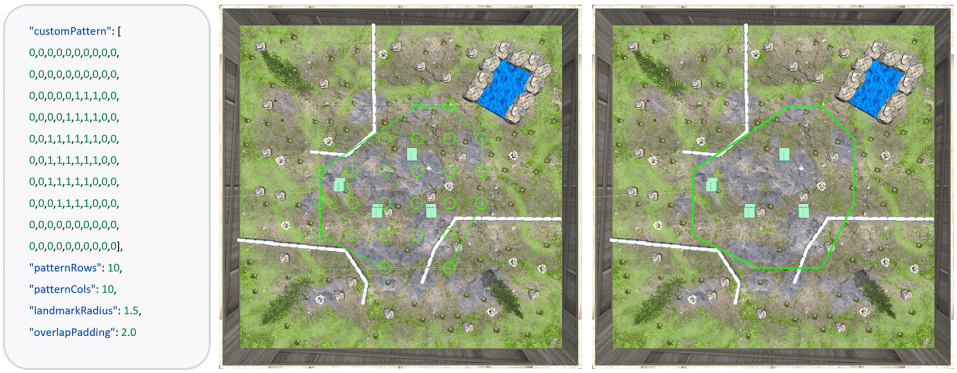

Figure 15: **Landmark-based predator navigation boundary.** Left: A 10x10 array is defined (i.e., 'customPattern'). Middle: Landmarks (depicted in green circles) are placed according to the number specified per grid cell. However, if an obstacle is present that would block predator navigation, the landmark is omitted from that location. A convex hull (represented in green line) is computed around the placed landmarks to define the predator's navigable area. Right: The final convex mesh collider (in green outline) restricts the predator's range of motion.

Figure 16: **Predator deployment configuration in EVAAA.** Left: JSON configuration for a predator agent, including key parameters such as position range, view angle (120 degrees), walk speed, and state transition limits (e.g., 'maxRestingSteps', 'maxSearchingSteps'). Right: Visualization of four predators navigating using Unity's NavMesh system.

during daytime and decreases at night, requiring agents to adjust their thermal regulation behavior over time.

G Naturalistic training level design

EVAAA implements a two-tiered environment structure: (1) Naturalistic training environments and (2) unseen experimental testbeds. The naturalistic training environment is organized as a curriculum that gradually increases environmental complexity, enabling agents to learn homeostatic regulation across various challenges–from basic resource foraging to predator avoidance. To evaluate whether agents have learned homeostatic behaviors and can generalize beyond these basic survival behaviors, we introduce a set of unseen test environments. The unseen test environments are described in the next section G.

Level 1. Basic resource foraging Level 1 focuses on basic survival by training agents to regulate internal states through food and water (random locations) foraging and consumption. In Level 1-1, 50 red food cubes and 50 blue water cubes are randomly distributed across the field with a variable ambient temperature. This setup promotes learning the direct link between resource consumption and increases in EV level. In Level 1-2, resources are relocated to the two opposite diagonal corners of the field, requiring more spatially directed foraging.

Figure 17: **Day/night cycle configuration and visual transition examples.** Top: Configuration parameters controlling the day-night cycle, including fog density, temperature shift, sun angle, and clipping distance. Bottom: Five key visual phases of the cycle: day, sunset, night, deepnight, and dawn. These phases dynamically modulate ambient lighting, visibility, and temperature, influencing agent behavior and sensory inputs.

Level 2. Obstacle-resource mapping Level 2 increases task complexity by introducing environmental hazards and spatial cues. In Level 2-1, bonfires act as localized heat sources, requiring agents to actively regulate body temperature, while rocks and bushes obstruct movement. The agent must avoid collisions to keep the damage level within a viable range. Level 2-2 introduces consistent spatial layouts: water cubes now appear as a stationary pond providing a reliable water source, and food, now depicted as apples, spawns only under trees. These spatial cues encourage the agent to learn predictable environment-resource associations and support more strategic, non-random foraging behavior.

Level 3. Terrain exploration and navigation Level 3 supports flexible navigation and adaptive exploration within partially structured environments. In Level 3-1, three tree regions are positioned in separate corners of the field, each partially enclosed by walls and bushes. Apples are placed beneath these trees, requiring the agent to navigate around obstacles while managing path efficiency and thermal regulation. In Level 3-2, food location varies across time: at the beginning of each episode, apples spawn in only one randomly selected tree region (Fig. 18). After the agent consumes all the food in that region, the apples reappear in a randomly selected tree. This shifting food placement encourages agents to explore, identify the current resource location, and adjust their foraging strategies using spatial navigation skills and rapid decision-making.

Level 4. Dynamic threats Level 4 integrates predator threats with a day/night cycle, challenging agents to coordinate survival behavior with internal regulation. In Level 4-1, predators patrol the field and inflict damage upon contact, but revert to a passive search state once the agent escapes their line of sight. To prevent unavoidable deaths during resource consumption, predators are restricted to pre-defined navigation zones (Fig. 15). In Level 4-2, predator behaviors and ambient temperature shift over time: predators pursue during the daytime but remain inactive at night, while temperature rises and falls accordingly. These conditions require agents to balance safety, thermoregulation, and foraging, promoting time-sensitive and adaptive strategies.

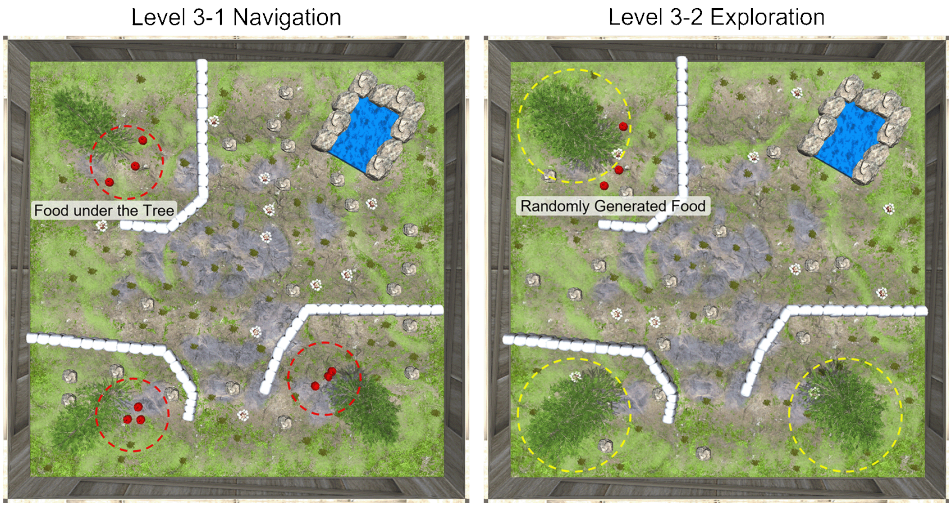

Figure 18: **Level 3 food resources random generation example.** In Level 3-1, the apples are placed at all three tree regions. In Level 3-2, on the other hand, at the start of each episode, food spawns in a randomly selected region (e.g., yellow dotted regions) and relocates to another once consumed.

H Experimental testbed design

H.1 Basic Homeostatic Regulation

Two-resource choice task To assess whether agents can prioritize among competing internal needs, this task presents two distinct resources (e.g., food vs water or food vs bonfire to warm the body temperature) separated by a wall. Once the agent commits to one side of the resource area, access to the other side becomes temporarily obstructed. Successful performance requires selecting the resource most urgently needed to maintain current homeostasis.

Collision avoidance task This task evaluates the agent's ability to anticipate and avoid damage. A visible resource (e.g., a food red cube) is placed behind narrow passages or closely spaced obstacles that induce damage upon contact. Although the resource is visible from the start, the agent must navigate safely through constrained space, demonstrating precise motor control and the capacity to inhibit impulsive actions in favor of safe, state-preserving paths.

H.2 Advanced Adaptive Skills

Risk-taking task In this setting, the agent must reach a needed resource (e.g., food or water) located behind a bonfire that rapidly raises the agent's internal temperature level. The agent must decide whether to endure the rising temperature to preserve other essential variables, which would otherwise decline over time and ultimately lead to failure in survival. This task tests the agent's ability to manage the conflicting needs.

Y-maze (spatial navigation) The Y-maze evaluates agents' ability to make spatial decisions based on internal state comparisons. Two resource types (e.g., food and water) are placed at the ends of a Y maze. At the start of each episode, the agent begins at the base and must choose which arm to explore first. If the satiation level is lower than hydration, the optimal strategy is to visit the food cube located arm first. Once the satiation level is restored and hydration becomes more depleted, the agent must redirect toward the water arm that was previously seen but not initially visited. This task evaluates the agent's ability to monitor internal needs, recall previously encountered spatial locations, and flexibly sequence goal-directed actions in response to evolving interoceptive demands.

Goal manipulation under volatile internal-state conditions This task introduces sudden, unobservable changes to the agent's internal state during ongoing action execution. The change is triggered when the agent crosses a transparent boundary (**Fig. 19**). For example, the agent may begin

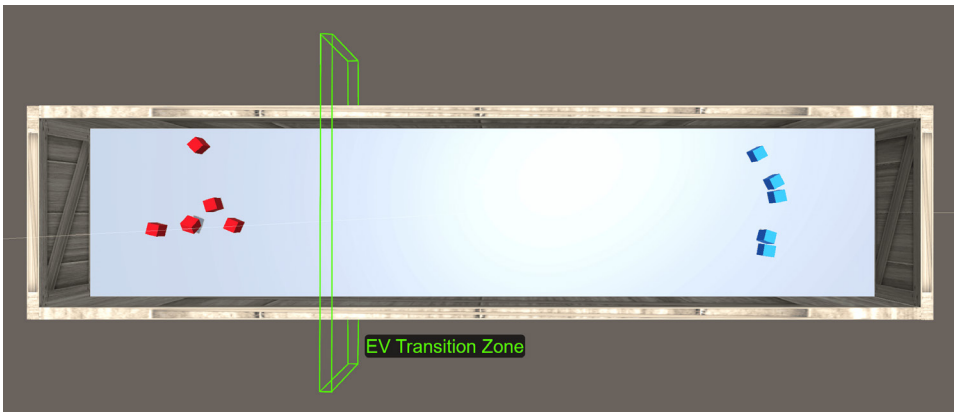

Figure 19: **Goal manipulation under volatile internal-state conditions.** The agent initiates movement in one direction based on its initial EV level. Upon crossing an invisible EV transition zone (indicated by the green box), the EV level is altered, prompting the agent to reverse direction.

moving toward a food resource due to low satiation levels, but after crossing the boundary, hydration becomes the most depleted EVs. The agent must recognize this internal shift, revise its plan, and re-prioritize its goals accordingly. This setting tests the agent's ability to adapt goal-directed actions based on internal state changes and serves as a virtual analogue to real-world behavioral experiments where manipulating internal physiological states is challenging, such as in animal studies.

Multi-goal planning The agent must coordinate actions across three EVs: satiation, hydration, and temperature (e.g., warmth from a bonfire). The agent's behavior is guided by an internal priority that reflects the current levels of EVs. For example, if the current internal state is as temperature < hydration < satiation, the optimal policy is to first seek warmth to restore body temperature, then hydrate, and finally consume food. Success depends on the agent's ability to plan and execute actions according to the relative urgency of each EV while maintaining overall homeostasis. While multiple action sequences may be valid, the agent must keep all EVs within a stable range throughout the test period.

Predators with day/night In this task, the resource is visible, but predators actively move around and attempt to attack the agent during the day, making direct access risky. At night, predators become inactive, providing a safer opportunity to obtain the resources. The optimal strategy is to avoid predators during the daytime and wait until nightfall to approach and take the resource. This setup is designed to assess the agent's ability to balance immediate needs and survival by demonstrating risk-aware, time-sensitive decision-making.

I Evaluation protocol

I.1 Computational resources

All experiments were conducted using a single NVIDIA RTX 3090 GPU. Training times varied depending on task complexity, with most environments requiring approximately 1.5 to 2 days to reach convergence under default curriculum schedules and environment configurations. The system ran under a standard PyTorch + Unity ML-Agents setup with GPU acceleration.

I.2 Architectures and hyperparameters

We implemented three reinforcement learning agents: DQN, PPO, and DreamerV3. For PPO and DreamerV3, we adopted the Sheeprl framework `https://github.com/Eclectic-Sheep/sheeprl`, which provides modular and reproducible implementations aligned with recent benchmarks. In contrast, the DQN agent was developed from scratch to suit our environment's specific requirements. The following sections describe the model architectures for each agent.

DreamerV3 DreamerV3 was implemented using the Sheeprl API with the $dreamer_v3_S.yaml$ configuration. The agent consists of a world model, an actor, and a critic. The world model encodes RGB observations using a four-layer convolutional network with progressively increasing channels ($32 \to 64 \to 128 \to 256$), kernel size 4×4, stride 2, and LayerNorm applied after each layer. Vector observations—including essential variables, olfactory, thermal, and collision features—are processed via a two-layer MLP with 512 units per layer, SiLU activations, and LayerNorm. Latent dynamics are modeled through a recurrent state-space model (RSSM) comprising a feedforward MLP ($1029 \to 512$), a LayerNorm-GRU cell ($1024 \to 1536$), and two additional MLPs representing the transition and representation models ($512 \to 1024$ and $5120 \to 1024$, respectively). The decoder reconstructs visual input using a transposed convolutional network ($256 \to 128 \to 64 \to 32 \to 3$), and reconstructs vector modalities using an MLP decoder with multiple linear heads for different variables. The actor and critic are two-layer MLPs ($1536 \to 512 \to 512$), both with LayerNorm and SiLU activations. The actor outputs the actions (5), while the critic predicts value distributions across 255 bins. All components follow Hafner's initialization scheme and use consistent activation and normalization settings.

PPO PPO was implemented using the Sheeprl API with a dual-branch encoder and separate actor and critic heads. Visual inputs are constructed by stacking four consecutive 64×64 RGB frames, resulting in a 12-channel input image. This consists of three convolutional layers with ReLU activations (8×8, 4×4, and 3×3 kernels; strides 4, 2, and 1), followed by a fully connected projection to a 512-dimensional visual embedding. Vector inputs—including essential variables, olfactory, thermal, and collision signals—are passed through a two-layer MLP with ReLU activation and a final projection to 64 dimensions. The concatenated feature vector (576 dimensions) is shared across the actor and critic networks. The actor is implemented as a one-layer MLP that produces a latent feature vector, which the agent subsequently interprets as the parameters of a diagonal Gaussian distribution over actions. The critic is a two-layer MLP that outputs scalar state values. Both heads use ReLU activations and follow a consistent architectural template defined in the configuration. This design ensures efficient encoding of multimodal state information while maintaining architectural symmetry across policy and value functions.

DQN Visual input is represented as a stack of two consecutive 64×64 RGB frames (6 channels total) and passed through a four-layer convolutional encoder with increasing channel widths ($16 \to 32 \to 64 \to 128$), each followed by BatchNorm and ReLU activations. The resulting visual embedding is projected to 1000 dimensions. Vector observations—including essential variables, olfactory, thermal, and collision signals—are each passed through independent linear encoders that project to 50–100 dimensions depending on the modality. These embeddings are concatenated with the visual feature and passed through a fully connected fusion layer ($1450 \to 400$) before producing action-value estimates via a final output layer ($400 \to 5$). The network is trained using the standard DQN objective with experience replay and a target network. Reward shaping is enabled to enhance learning performance in the presence of sparse or delayed feedback. The design ensures tight integration between perceptual and internal state features through modality-specific encoders and joint feature fusion.

See **table 2** for hyperparameters used for training DQN, PPO, and Dreamer V3.

I.3 Success rate metrics

Success rates were computed independently for each experimental test condition using task-specific criteria aligned with the behavioral objectives of each task. Evaluation strategies fall into two categories: single-trial success and survival-duration success. For survival-based tasks, a fixed maximum episode length (maxStep) is assigned per task, defining the duration the agent must survive to be considered successful.

Single-Trial Success Criteria. These tasks focus on discrete resource acquisition. An episode is marked as successful if the agent completes the designated objective before reaching the maxStep:

- **Two-resource choice tasks (Food, Water):** Success is defined as EpisodeEndType = ResourceConsumed and positive consumption of the relevant resource (FoodConsumed > 0, WaterConsumed > 0). (maxStep = 500)

Table 2: **Hyperparameters used for training**

Parameter	DQN	PPO	DreamerV3-S
Optimizer	Adam	Adam	Adam
Learning rate	0.0001	0.00025	0.0001
Other optimizer params	[l]ϵ-greedy:		
decays 1.0→0.1 (209k)	[l]$\epsilon = 10^{-5}$		
decay=0	[l]$\epsilon = 10^{-6}$		
decay=0			
Grad norm clipping	–	0.5	`actor/critic=100, world_model=1000`
Baseline cost	–	0.5	1
Entropy cost	–	0.01	0.0003
Discount factor (γ)	0.95	0.99	0.997
GAE Lambda (λ)	–	0.95	0.95
IS clip range (ϵ_{clip})	–	0.1	–
Batch size	12	256	16
Unroll length	–	1024	64
Actor gradients	TD (Q-learning)	Analytical (GAE)	REINFORCE

- **Goal manipulation under volatile internal-state conditions:** Success requires adapting to an internal state switch and reaching the newly prioritized resource.(maxStep = 300)
- **Risk-taking task:** Success is defined as EpisodeEndType = ResourceConsumed, reflecting successful resource acquisition under threat. (maxStep = 1000)
- **Collision avoidance task:** Success is defined as EpisodeEndType = ResourceConsumed, requiring the agent to navigate hazard zones to reach the target. (maxStep = 500)
- **Predators with day/night:** Success is defined as EpisodeEndType = ResourceConsumed, requiring the agent to avoid moving predators to reach the target. (maxStep = 300)

Survival-Duration Success Criteria. These tasks emphasize sustained internal regulation and long-horizon planning. An episode is marked as successful if the agent survives for the entire duration, defined by the maxStep limit:

- **Two-resource choice tasks (Temperature):** Success = EpisodeEndType = MaxStepReached, indicating full survival under thermal constraints. (maxStep = 300)
- **Y-Maze (spatial navigation):** Success = EpisodeEndType = MaxStepReached, requiring the agent to navigate and switch goals adaptively. (maxStep = 350)
- **Multi-goal planning:** Success = EpisodeEndType = MaxStepReached, reflecting correct prioritization across multiple EVs over time.(maxStep = 350)

For each agent model and experiment type, the total number of episodes and the number of successful episodes are recorded. The success rate is computed as Success Rate $= \frac{\text{Successful Episodes}}{\text{Total Episodes}} \times 100$

I.4 Human evaluation protocol

To assess human performance on the same tasks as EVAAA agents, we conducted a controlled behavioral study with eight participants. The study was approved by the Institutional Review Board (IRB), and informed consent was obtained from all participants. The whole experiment lasted approximately 2–3 hours and consisted of two phases: a curriculum-based training phase and a test phase. During training, participants experienced the same naturalistic training environments to understand the relationship between environment and EV, and reward feedback. Most participants spend considerable time in Level 1-1, first learning how to interact with the environment—such as using keyboard inputs (e.g., arrow keys, spacebar)—and then exploring how different actions influence EV levels. After each level, experimenters pose diagnostic questions to evaluate participants' understanding of EV regulation (e.g., what actions led to successful survival). These questions serve

as a practical substitute for the full training experience, as it is unrealistic to ask human participants to complete the same number of steps as agents (e.g.,~100,000steps is 400 minutes of continuous play for each level of naturalistic tasks). We advanced to the next level when participants demonstrated a sufficient level of understanding.After completing the curriculum, participants received a brief explanation of the evaluation protocol and completed the same test environments. Participants were compensated at a rate of $11/hour Participation was voluntary, with the option to withdraw at any time without penalty. No personally identifiable or biometric data was collected.

J Results

J.1 Key behavioral-based analysis of agent and human behavior across unseen test tasks

As noted in the main results, we also implemented a recurrent baseline (PPO+LSTM). However, this model failed to demonstrate successful learning (**Fig. 20**). Therefore, the following behavioral-event analysis focuses on the agents trained with DQN, PPO, and DreamerV3. To complement the main result, we present an analysis based on selected key behavioral events across tasks, highlighting key environmental interactions that influence success rates. **Figure 21-24** visualizes normalized event counts (mean ± SE) for each algorithm (e.g., DQN, PPO, DreamerV3), human participants, and training level, offering insights into behavioral patterns such as resource consumption and causes of failure. Across testbeds, we observe distinct strategies in how different agents and human participants manage EV levels and respond to task-specific hazards. In tasks such as *Two-resource choice tasks (Food, Water)*, *Risk-taking task*, *Collision avoidance task*, *Goal manipulation under volatile internal-state conditions*, and *Predators with day/night*, higher success rates can be achieved when agents and humans engage in food or water consumption actions that are aligned with their current essential variable (EV) levels, often through task-specific strategies such as avoiding collision while approaching the resource. In contrast, tasks such as *Two-resource choice tasks (Temperature)*, *Y-Maze (spatial navigation)*, and *Multi-goal planning*, emphasize long-term survival, where success depends on maintaining internal-state balance until the task's maximum step rather than reaching a discrete goal. For instance, in *Two-Resource Choice (Temp)*, successful agents and humans tend to remain near heat sources (e.g., bonfire) as internal temperature drops, maintaining this positioning strategy over time instead of over-consuming food or water. Overall, these analyses provide a detailed framework for dissecting the behavioral dynamics underlying agent success or failure. By analyzing key event patterns, we can identify which action patterns facilitate goal completion as well as maladaptive behaviors that correlate with failure, offering mechanistic insights into agent-environment interactions and internal state regulation.

J.2 Agent emergent behavior

Modality ablation study. As discussed in the main text, we conducted a systematic ablation study to assess the relative importance of each sensory component. We trained DreamerV3 agents with one of five components removed: (1) Essential Variables (EVs), (2) Vision, (3) Olfaction, (4) Thermoception, and (5) Collision. Agents in the 'No EV' and 'No Vision' conditions fail to learn, confirming these features are critical. Agents in the 'No Olfaction', 'No Thermoception', and 'No Collision' conditions demonstrate varying degrees of learning, allowing for an analysis of their differential impact (**Fig. 25**). Following this, the **Figure 26** demonstrates the generalization performance of these agents on the unseen test tasks.

Detouring behavior after misjudgment. In the two-resource choice task (e.g., food vs. water choice), agents trained on Level 1-2 or higher consistently exhibited detouring behavior. After initially approaching the incorrect resource—for instance, heading toward the food cube while dehydrated—they corrected their trajectory by detouring around the wall to reach the appropriate resource. In contrast, agents trained exclusively on Level 1-1, where resources are randomly placed and easily accessible, tended to persist along suboptimal paths without corrective action. The emergence of detouring behavior underscores the role of stable environment-resource associations in facilitating spatial re-planning and policy adjustment.

Self-Termination. Agents trained on Level 3-2 frequently exhibit deliberate episode termination behavior in the Y-maze (spatial navigation) task. Unlike earlier training levels, such as Level 3-1, where food consistently appears in predictable locations, Level 3-2 introduces uncertainty by spawning apples in only one randomly selected tree region. When unable to locate the food area,

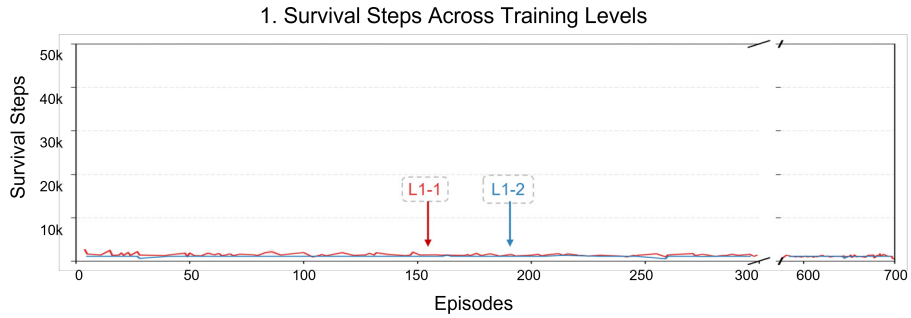

Figure 20: **Training performance of the recurrent baseline (PPO+LSTM).** PPO+LSTM shows survival steps remained minimal throughout training in both Level 1-1 (red) and Level 1-2 (blue). This indicates a failure to learn, justifying the model's exclusion from the detailed behavioral-event analysis.

some agents adopt a strategy of repeatedly colliding with allies to deplete their damage EV and terminate the episode. This behavior reflects a learned policy that favors resetting the environment over persisting in low-reward or difficult-to-recover states.

Videos demonstrating these emergent behaviors are available in the GitHub repository.

K Broader impacts

EVAAA is a simulation platform for training and evaluating agents based on internal survival needs such as satiation, hydration, body temperature, and tissue integrity (the level of damage). By focusing on intrinsic regulation rather than external rewards, EVAAA enables research on adaptive behavior, goal prioritization, and autonomous decision-making under internal pressure. This line of research may inform the design of more robust agents for applications like healthcare support, education, or uncertain environments. However, internal-state-driven learning raises challenges in transparency and control. For example, agents may exhibit opaque strategies when internal signals are not externally visible, or misaligned behaviors if deployed without careful oversight. To support responsible use, all code and tasks are released as open-source. We encourage users to ensure interpretability, avoid overfitting to abstract survival dynamics, and evaluate downstream impacts before applying such systems in real-world contexts.

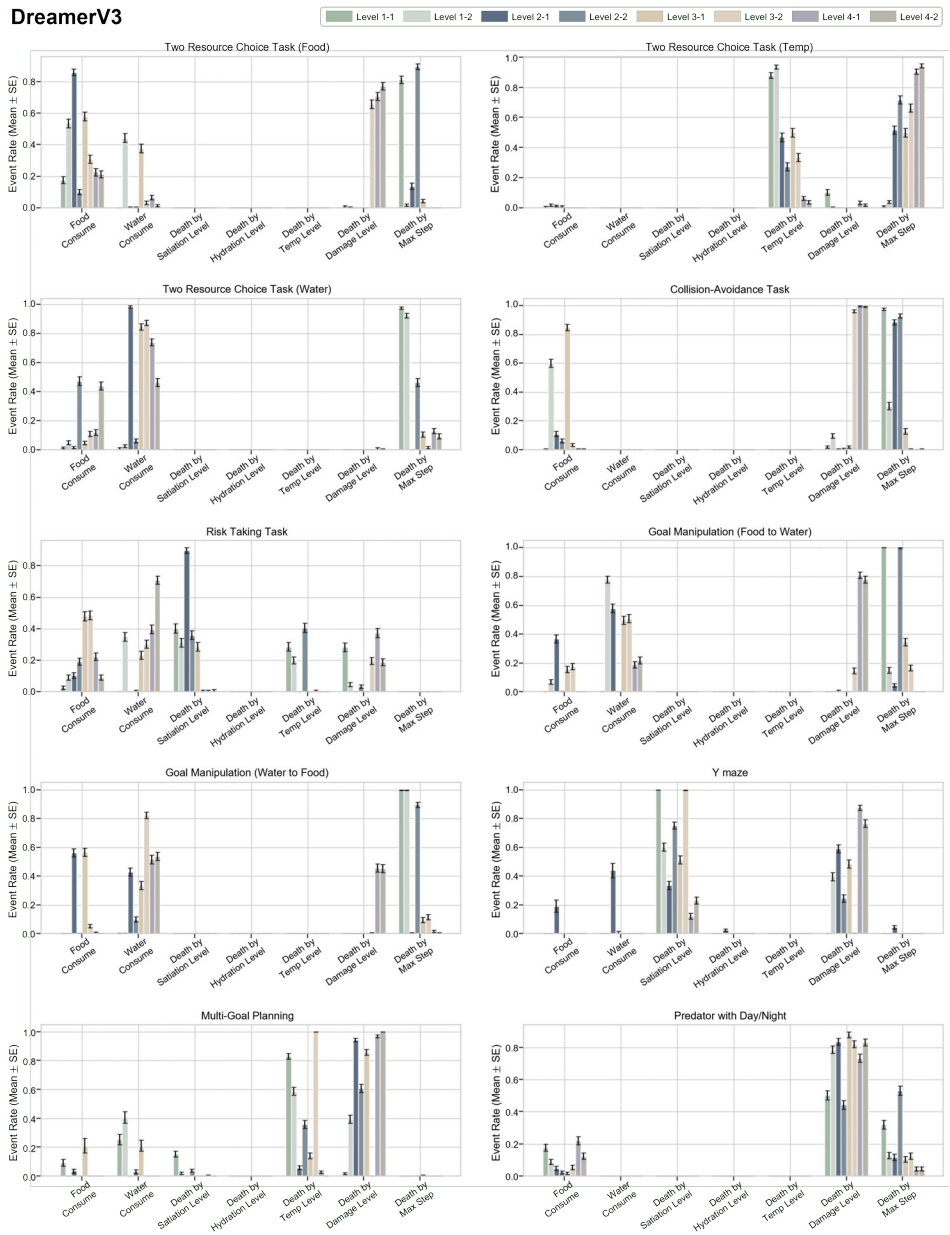

Figure 21: **DreamerV3 event-based behavior across test tasks.** DreamerV3 demonstrates flexible and adaptive strategies across ten test environments. It actively manages internal states by adjusting resource use and positioning, leading to higher success in both simple and complex tasks.

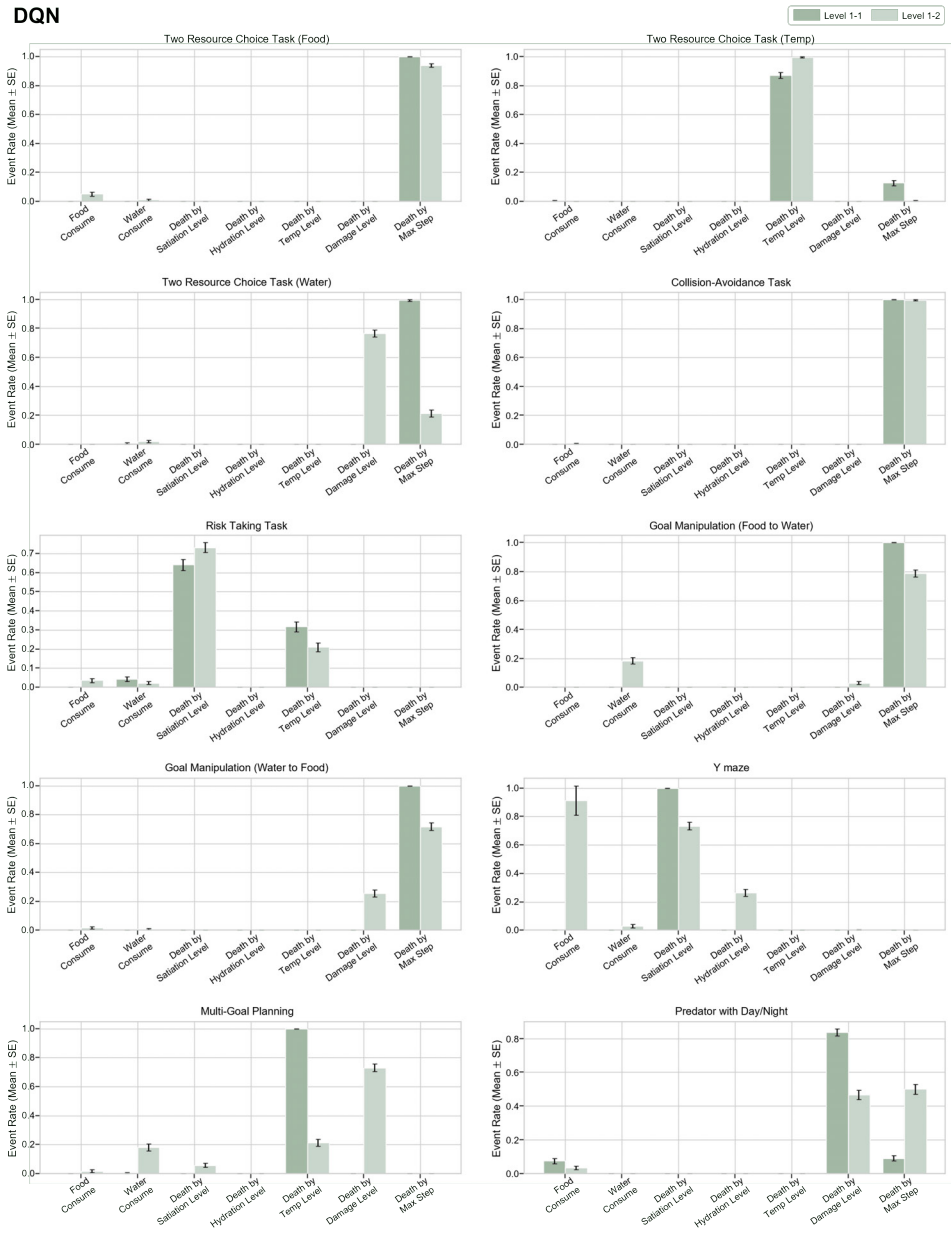

Figure 22: **DQN event-based behavior across test tasks.** DQN exhibits limited adaptability when evaluated in testbed environments. Event patterns reveal frequent failures in appropriate resource consumption, leading to low success rates—primarily due to reaching the maximum allowed steps—indicating the agent's difficulty in regulating internal states and adapting behavior beyond its training experience.

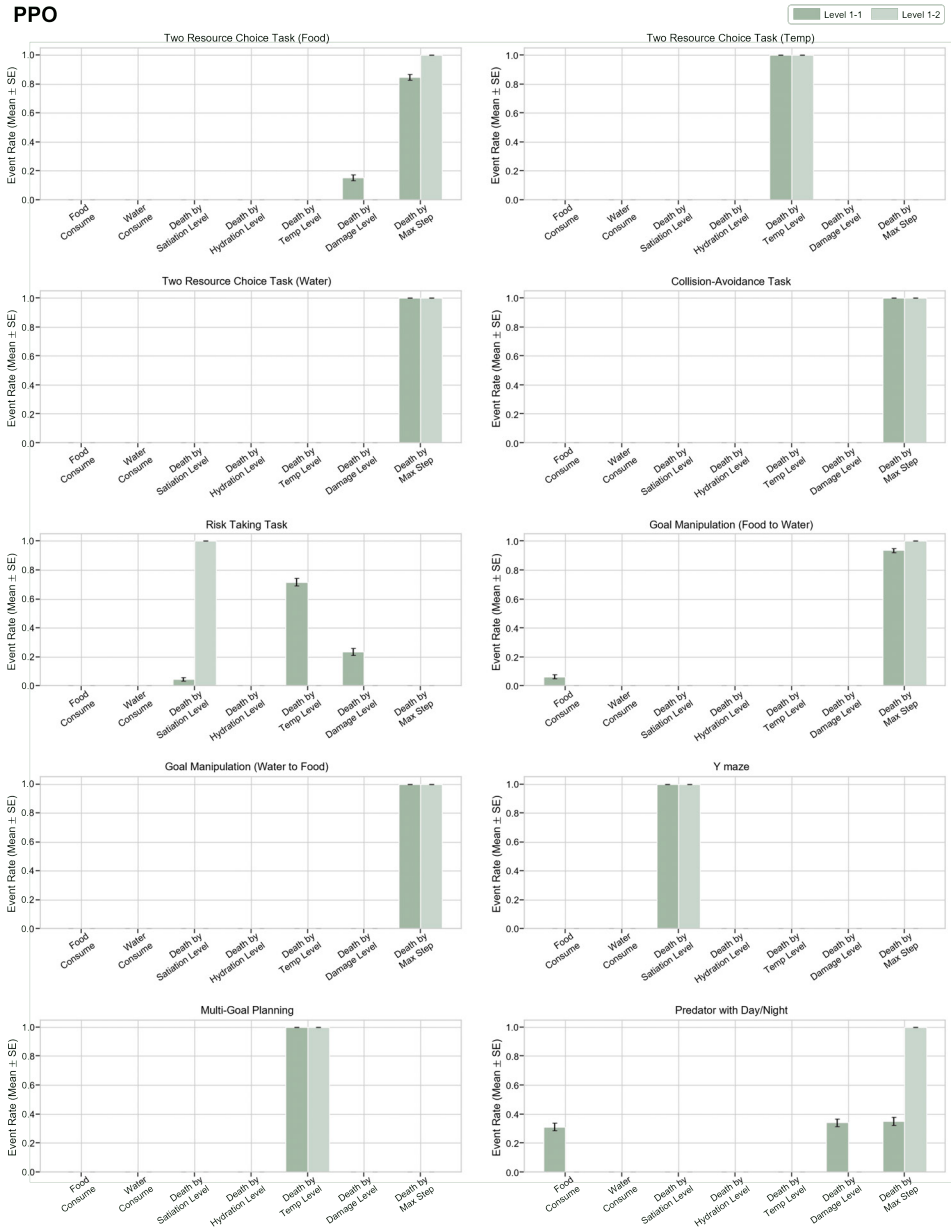

Figure 23: **PPO event-based behavior across test tasks.** PPO demonstrates slightly better adaptability than DQN, yet still fails to regulate internal states effectively. Agent deaths frequently result from insufficient resource consumption, with failures often linked to temperature regulation.

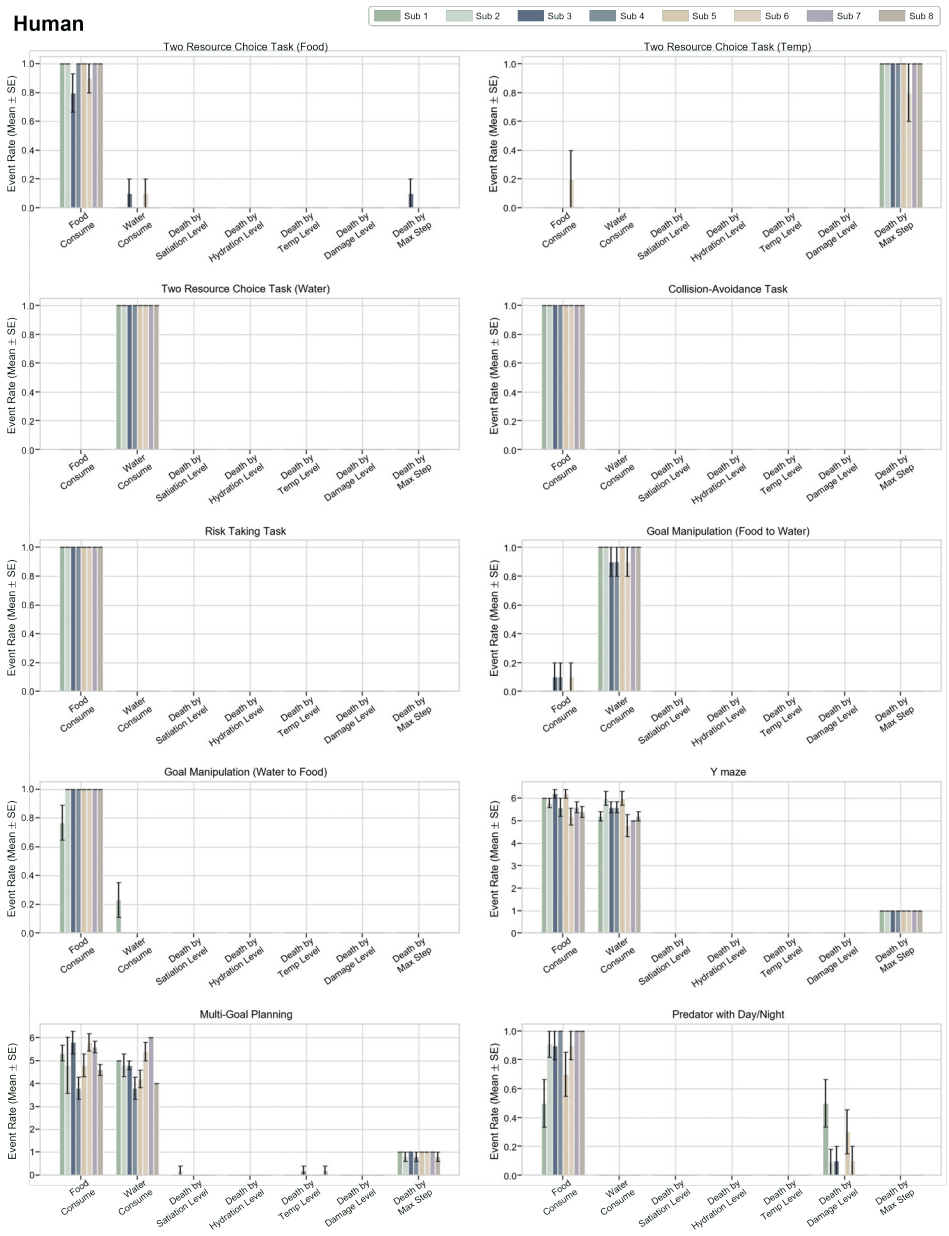

Figure 24: **Human event-based behavior across test tasks.** Human participants consistently show high levels of appropriate resource consumption and low internal failure rates across tasks. Their event patterns indicate proactive management of internal states, such as consistently consuming food and water before critical depletion, which supports high success and flexible adaptation in diverse environments.

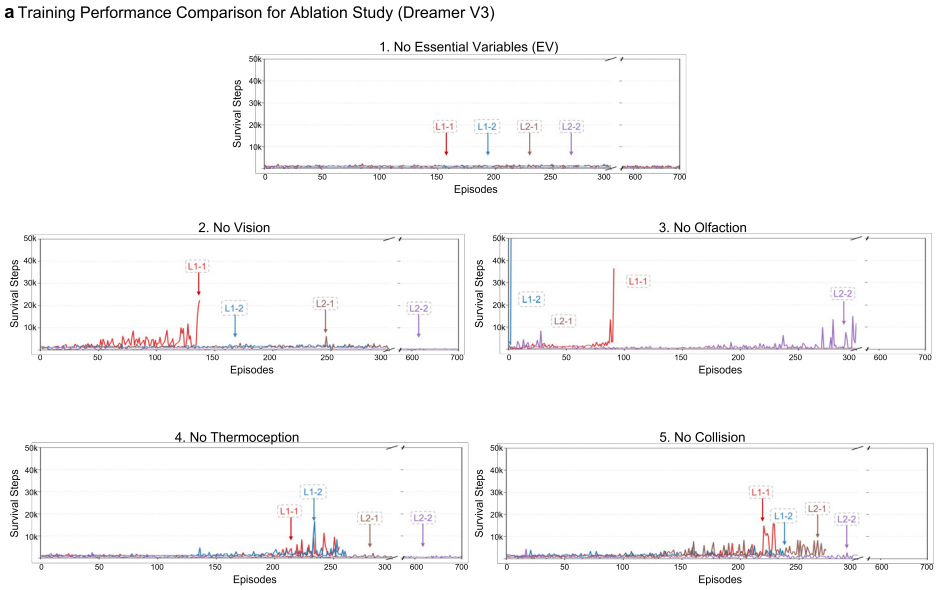

Figure 25: **Training performance comparison for the modality ablation study.** Training performance for agents trained with one sensory component removed. (1) 'No Essential Variables' and (2) 'No Vision' conditions result in minimal successful learning, confirming their critical role. The remaining conditions illustrate the differential impact of non-visual modalities: agents in the (3) 'No Olfaction' condition perform notably better than those in the (4) 'No Thermoception' and (5) 'No Collision' conditions, suggesting thermoception and collision are more impactful for successful training.

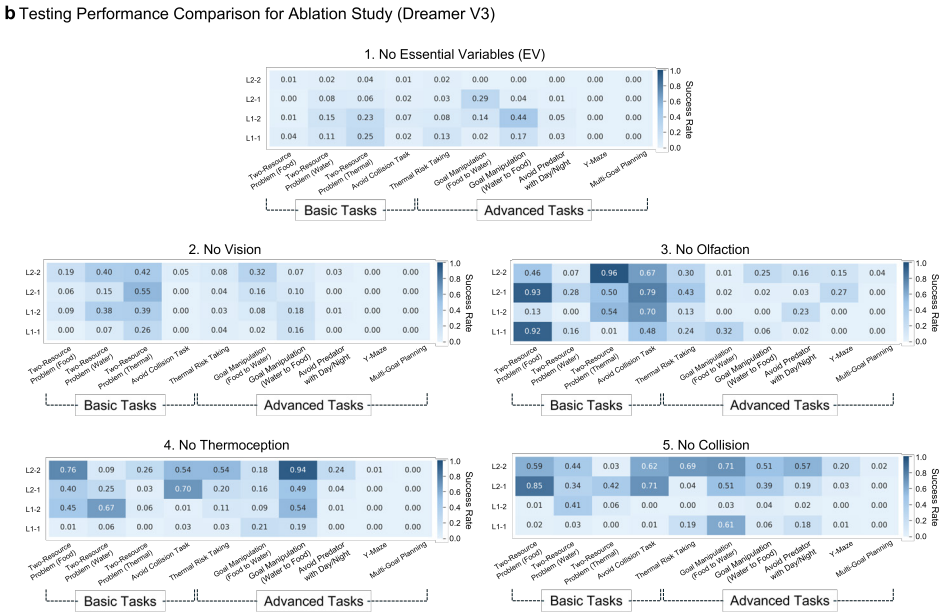

Figure 26: **Testing performance comparison for the modality ablation study.** Testing performance (success rate) on unseen test tasks for the five ablation conditions. Agents in the (1) 'No EV' and (2) 'No Vision' conditions fail to generalize. In contrast, agents in the (3) 'No Olfaction' condition generalize well, especially on Basic Tasks. The (4) 'No Thermoception' and (5) 'No Collision' conditions show markedly poorer generalization than the 'No Olfaction' group, confirming that thermoception and collision are more important for generalization than olfaction.