AMaze: an intuitive benchmark generator for fast prototyping of generalizable agents

Kevin Godin-Dubois[1]*, Karine Miras[1] and Anna V. Kononova[2]

[1]Computer Science Department, Vrije Universiteit Amsterdam, Amsterdam, Netherlands, [2]LIACS, Leiden University, Leiden, Netherlands

Traditional approaches to training agents have generally involved a single, deterministic environment of minimal complexity to solve various tasks such as robot locomotion or computer vision. However, agents trained in static environments lack generalization capabilities, limiting their potential in broader scenarios. Thus, recent benchmarks frequently rely on multiple environments, for instance, by providing stochastic noise, simple permutations, or altogether different settings. In practice, such collections result mainly from costly human-designed processes or the liberal use of random number generators. In this work, we introduce AMaze, a novel benchmark generator in which embodied agents must navigate a maze by interpreting visual signs of arbitrary complexities and deceptiveness. This generator promotes human interaction through the easy generation of feature-specific mazes and an intuitive understanding of the resulting agents' strategies. As a proof-of-concept, we demonstrate the capabilities of the generator in a simple, fully discrete case with limited deceptiveness. Agents were trained under three different regimes (one-shot, scaffolding, and interactive), and the results showed that the latter two cases outperform direct training in terms of generalization capabilities. Indeed, depending on the combination of generalization metric, training regime, and algorithm, the median gain ranged from 50% to 100% and maximal performance was achieved through interactive training, thereby demonstrating the benefits of a controllable human-in-the-loop benchmark generator.

KEYWORDS

benchmark, human-in-the-loop, generalization, mazes, Reinforcement Learning

1 Introduction

Based on the need to fairly compare algorithms (Henderson et al., 2018), benchmarks have proliferated in the Reinforcement Learning (RL) community. These cover a wide range of tasks, from the full collection of Atari 2,600 games (Bellemare et al., 2013) to 3D simulations in Mujoco (Laskin et al., 2021). However, in recent years, the focus of research has changed from producing more complex environments to producing a *range* of environments. Although undeniable progress has been made with respect to the capabilities of trained agents, much remains to be done for their capacity to generalize (Mnih et al., 2015). In practice, agents "will not learn a general policy, but instead a policy that will only work for a particular version of a particular task with particular initial parameters" (Risi and Togelius, 2020).

Thus, a recurring theme in modern RL research is the training of agents in various situations to avoid overfitting. Although some algorithms have built-in solutions to smooth out the learning process, e.g., TD3 (Fujimoto et al., 2018) (where small perturbations are applied to the actions), providing such a diversity of experience primarily originates

from the environments themselves. To this end, numerous benchmarks now consist of a collection with varying degrees of homogeneity. Some of them have a similar structure, as in the Sonic benchmark (Nichol et al., 2018) where levels are small areas taken from three games in the franchise. In other cases, environments share very little: In the Arcade Learning Environment (ALE), the single common factors are the dimensions of the observation space (Bellemare et al., 2013). Intermediate test suites with distinct but complementary sets of "skill-building" tasks have also been designed, for example, with the Mujoco simulator (Wawrzyński, 2009; Yu et al., 2019; Laskin et al., 2021) or Meta-World (Yu et al., 2019). However, all of these examples share a common feature: the set of environments is predefined, generally the result of a costly human-tailored design procedure, e.g., Beattie et al. (2016).

To solve this generalization problem, agents must face sufficiently diverse situations so that the underlying principles are learned instead of a specific trajectory. Naturally, this requires generating environments that exhibit such diversity while still offering the same core challenges. For instance, in the context of maze navigation, reaching e.g., the exit might be the goal while the actual topology of said maze is only relevant insofar as giving agents a wide sample of states to learn from. One common way to address this later point is to use procedural generation (Beattie et al., 2016; Kempka et al., 2016; Harries et al., 2019; Juliani et al., 2019; Tomilin et al., 2022) or complementary techniques such as evolutionary algorithms (Alaguna and Gomez, 2018; Wang et al., 2019). For example, ProcGen (Cobbe et al., 2020) encompasses 16 different types of environment and serves as a generalizable alternative to ALE. Adapting more recent video game environments, either directly (Synnaeve et al., 2016) or in a light format (Tian et al., 2017), can help further push adaptability by providing finer-grained perceptions and actions. While such an approach can be used to create large training sets, the main difficulty becomes the design of a sufficiently tunable generator, i.e., one in which desirable features are easy to introduce.

Considering the challenges of generating a panel of demanding training environments, the contribution of this article is two-fold:

1. We introduce AMaze,[1] a generator for generic, computationally inexpensive environments of unbounded complexity that focus on generalization (via environmental diversity) and intelligibility (intuitive human understanding).
2. We demonstrate how such a generator is helpful in leading to more generalized performance (robust behavior w.r.t. unseen tasks) and how it can benefit from human input (e.g., to dynamically adjusting difficulty).

After highlighting, in Section 2, the niche this generator occupies in the current benchmark literature, we describe its main components in Section 3. Three alternative methodologies for training generalized maze-navigating agents are then detailed in Section 4 alongside two algorithms (A2C and PPO). The resulting performance in handling unknown environments is then thoroughly tested in Section 5, allowing us to draw conclusions

about the relative benefits of the generator, the training processes, and the underlying algorithms.

2 Related benchmarks

To place this generator in perspective, we conducted an extensive comparison with a select number of commonly used benchmarks. As our library is primarily targeted at Python environments, we restricted the set of considered environments to those that could be reliably installed and used on an experimenter's machine. Timing was done on 1000 time steps averaged over 10 replicates on an i7-1185G7 (3GHz) using the Python 3.10 version of all libraries, except for the Unsupervised Reinforcement Learning Benchmark (URLB) (Laskin et al., 2021) and RetroGym (Nichol et al., 2018) which required Python 3.8. In the latter case, we used the ROMs linked in the library's documentation. The scripts, intermediate data and figures are available as part of AMaze's repository.

As detailed in the following section, AMaze can provide environments for fully discrete, fully continuous, and hybrid agents. Table 1 illustrates how the former case allows for fast simulation at the cost of low observable complexity. Based on the time taken to simulate 1,000 timesteps, only the simplest of the gymnasium suite (Sutton and Barto, 2018) is comparable to AMaze which, in addition, provides numerous unique and experimenter-controlled environments. In the hybrid case, where agents perceive images but still only take discrete steps, the library is on par with Classic Control tasks (Barto et al., 1983) such as Mountain Car or Cart Pole. ProcGen (Cobbe et al., 2020) addresses similar concerns as AMaze and is quite comparable in terms of speed, but has a stronger focus on randomness, with difficulty levels being the main way of controlling the resulting environments. DeepMind Lab2D (Beattie et al., 2020), while noticably slower, is also extensively customizable, albeit through lua scripting, and allows for heterogeneous multi-agent experiments.

With respect to the fully continuous case, the most computationally expensive of the three regimes, AMaze performs at a level similar to that of Box2D or Mujoco (Todorov et al., 2012), which, in traditional implementations such as gymnasium (Towers et al., 2024), lack customization capabilities. Purely vision-based benchmarks such as ALE (Bellemare et al., 2013), RetroGym (Nichol et al., 2018), Meta-world (Yu et al., 2019), Unsupervised Reinforcement Learning Benchmark (Laskin et al., 2021), LevDoom (Tomilin et al., 2022), or Maze Explorer (Harries et al., 2019), while offering a more challenging task than AMaze, also exhibit drastically higher costs with variable levels of experimenter control over the environments.

To summarize, AMaze has comparable computational costs with the most simple of environments (e.g., Classic Control or Toy Text) while providing much finer grained control over the challenges proposed to the agent. Conversely, even in its most expensive variation (fully continuous) it outperforms complex settings such as Mujoco-based environments, ALE or Retro-Gym. Combined with its extensive and intuitive parameterization, this makes it an ideal platform for trying out new algorithms, policies or hypotheses before deployment on more demanding contexts.

---

1 AMaze library is available on PyPI at https://pypi.org/project/amaze-benchmarker/. The code for the experiment described thereafter is hosted at https://github.com/kgd-al/amaze_edhucat_2024.

TABLE 1 Comparison between AMaze and related benchmark (suites). All time metrics (performance) correspond to the wall time for 1,000 timesteps of the corresponding environment, averaged over 10 replicates. Qualification of the inputs, outputs, and control levels are taken from the related article or, when unavailable, directly from the sources. Overall, AMaze is competitive with small-scale benchmarks, but provides the experimenter with more control over the characteristics of the targeted environments. Complex environments (e.g., 3D) have much higher computational costs making AMaze an efficient and scalable prototyping platform.

Library	Family	N	Inputs	Outputs	Control	Median	Time (s)
A Maze	Discrete	64	Discrete	Discrete	Extensive	0.010	
	Hybrid	64	Image	Discrete	Extensive	0.025	
	Continuous	64	Continuous	Continuous	Extensive	0.102	
Gymnasium	Toy text	5	Discrete	Discrete	None	0.009	
	Classic control	5	Continuous	Both	None	0.023	
	Mujoco	22	Continuous	Continuous	None	0.090	
	Box2D	8	Continuous	Both	None	0.112	
	ALE	104	Image	Discrete	Modes	0.400	
Miscellaneous	ProcGen	100	Image	Discrete	Modes	0.031	
	Lab2D	11	Both	Discrete	Script	0.056	
	MetaWorld	50	Continuous	Continuous	None	1.228	
	URLB	24	Both	Continuous	None	3.929	
Retro-Gym	GameBoy	27	Image	Discrete	None	0.197	
	Sms	88				0.269	
	Nes	295				0.355	
	Genesis	331				0.472	
	Snes	184				0.736	
VizDoom	MazeExplorer	81	Image	Discrete	Extensive	0.553	
	LevDoom	72	Image	Discrete	None	1.059	

It follows that AMaze fills a very specific niche in the benchmarking landscape by providing a computationally inexpensive framework to design challenging environments. Control over the various characteristics of said environments is left in the hands of the experimenter through a number of high- and low-level parameters that will be described in the following section.

3 Generating mazes

Learning to navigate mazes represents a flexible, diverse, yet challenging tested for training agents. Here, we propose a *generator* (AMaze) for this task with the following primary characteristics:

Loose embodiment

The agent has access only to local spatial information (its current cell) and limited temporal information (previous cell). Arbitrarily complex visual-like information is provided to the agent in either discrete (preprocessed) or continuous (image) form as detailed in Section 3.2.

Computational lightweightness

No physics engine or off-screen renderings are required for such 2D mazes. Thus, challenging environments can be generated that are both observably complex (Beattie et al., 2020) and relatively fast (as seen in Table 1).
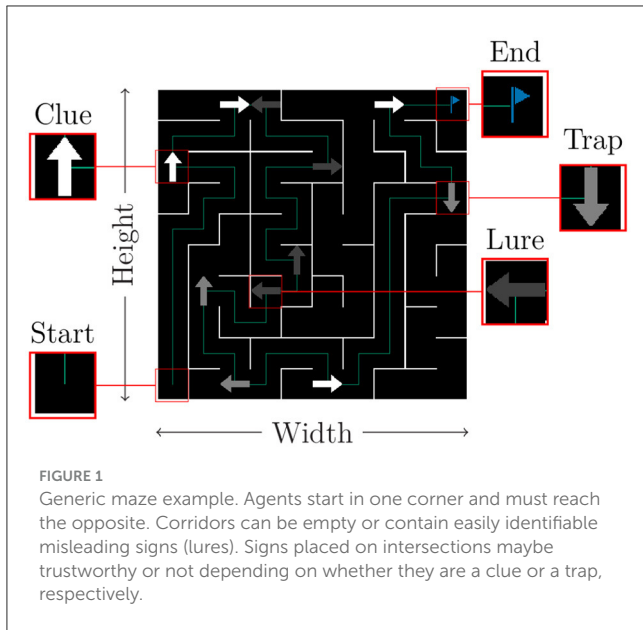
Open-endedness

As illustrated in Figure 1, a given maze results from the interaction of numerous variables controlled by the experimenter, such as its dimensions or the frequency and type of visual cues. In practice, an experimenter can inject any level of complexity into the maze by selecting images of appropriate deceptiveness as cues.

These features make it possible to generate a wide range of mazes for a variety of purposes, from a fast prototyping RL platform to a testbed for embodied computer vision in indirectly encoded NeuroEvolution. In the remainder of the section, we detail the major components of this generator namely the environment's parameters, the agents' capabilities, the reward function and, finally, unifying metrics for comparing widely different mazes.

3.1 Maze generation

A maze is defined, at its core, by its size (width, height) and the seed of a random number generator. A depth-first search algorithm is then used to create the various paths and intersections with the arbitrary constraint that the final cell is always diagonally opposed to the starting point (itself a parameter). Additionally, mazes can be made unicursive by blocking every intersection that does not lead directly to the goal. Such mazes are called *trivial*, as

FIGURE 1
Generic maze example. Agents start in one corner and must reach the opposite. Corridors can be empty or contain easily identifiable misleading signs (lures). Signs placed on intersections maybe trustworthy or not depending on whether they are a clue or a trap, respectively.

an optimal strategy simply requires going forward without hitting any wall. In contrast, general-purpose mazes do have intersections, the correct direction being indicated by a sign, hereafter called a *clue*. This corresponds to the class of *simple* mazes, since making the appropriate move in such cases is entirely context dependent. Each intersection *on the path to the goal* is labeled with such a clue.

However, to provide a sufficient level of difficulty, additional types of sign can also be added to a given maze with a user-defined probability. *Lures*, occurring with probability p_l, are easily identifiable erroneous signs that request an immediately unfavorable move (going backward or into a wall). They can be placed on any non-intersectional cell along the path to the solution. *Traps*, replace an existing clue (with probability p_t) and instead point to a dead end. These types of sign are much harder to detect as they do not violate local assumptions and can result in large, delayed negative rewards. Mazes containing either of these misleading signs are named accordingly, while mazes containing *both* are called *complex*.

3.2 Agents and state spaces

To successfully navigate a maze, the learning agent must only rely on the visual contents of its current cell to choose its next action. The framework accounts for three combinations of input/output types: fully discrete, fully continuous, and hybrid (continuous observations with discrete actions). Observations in continuous space imply that cells are perceived directly as images albeit with a lower resolution than that presented to humans. Thus, wall detection may not be initially trivial (even for unicursive mazes), and sign recognition comes into play with the possibility of using different symbols for different sign types. In the discrete space, the agent is fed a sequence of eight floats, corresponding to preprocessed information in direct order $(W_e, W_n, W_w, W_s, S_e, S_n, S_w, S_s)$, as illustrated in Figure 2.

In this case, the observations take the form of a monodimensional array containing all eight fields, in direct order. Signs can be differentiated through their associated decimal value, which is fully configurable by the experimenter. In the subsequent experiment, we used a single sign of each type with values of 1.0, 0.5, and 0.25 for clues, traps, and lures, respectively. The walls and the originating direction (limited temporal information) are assigned fixed values of 1.0 and 0.5, respectively.

With respect to actions, a discrete space implies that the agent moves directly from one cell to another by choosing one of the four cardinal directions. In contrast, in a continuous action space, the agent controls only its acceleration.

3.3 Reward function

An optimal strategy, in the fully discrete case, is one where the agent makes no error: no wall collision, no backward steps, and naturally, correct choices at all intersections. Although identical in the hybrid case, as the increase in observation complexity does not change the fact that there exists only one optimal trajectory, this statement no longer holds for the fully continuous case, at least not in the trivial sense. In fact, by controlling its acceleration, an agent can take shorter paths along corners or even take risks based on assumed corridor lengths.

However, in all cases, the same reward function is used to improve strategies as defined by:

$$
\begin{aligned}
r(s, a, s') = \ &\rho_e, && \text{if } s' \text{ is the goal} \\
&- \rho_w, && \text{if } a \text{ caused a collision} \\
&- \rho_b, && \text{if } a \text{ caused a backward step} \\
&- \rho_t, && \text{constant time penalty}
\end{aligned}
\tag{1}
$$

Given l, the length of the optimal trajectory, we define two versions of the reward function: r and its normalized version $\bar{r}$. The first is used during the training process to provide large incentive toward reaching the goal, while the second's purpose is to compare performance on mazes with different sizes. Furthermore, we refer to the cumulative (episodic) reward as R and $\bar{R}$, respectively. Table 2 details the specific values used in this experiment.

3.4 Evaluating maze complexity

Due to the randomness of the generation process, two mazes with different seeds can have very different characteristics. Thus, to provide a common ground from which mazes can be compared, we define two metrics based on Shannon's entropy (Shannon, 1948). First the *Surprisingness* $S(M)$ of a maze M:

$$
S(M) = - \sum_{i \in I_M} p(i) * log_2(p(i))
\tag{2}
$$

where $p(i)$ is the observed frequency of input i and I_M is the set of inputs encountered when performing an optimal trajectory in M. Second, the *Deceptiveness* $D(M)$ defined as:

$$
\begin{aligned}
\text{cells}(M) &= \{c[0:3], \forall c \in M\} \\
\text{traps}(M) &= \{c, \forall c \in M/\text{cost}(c) > 0\} \\
D(M) &= \sum_{c \in \text{cells}(M)} \sum_{\substack{s \in \text{traps}(M) \\ s[0:3]=c}} -p(s|c)log_2(p(s|c))
\end{aligned}
\tag{3}
$$

FIGURE 2
Discrete observation space. **(a)** Visual inputs: W_* denotes whether there is a wall in the corresponding direction, as well as the direction of the previous cell; S_* is non-zero if a sign points toward the corresponding direction. **(b)** Examples: Sample inputs from cells highlighted in Figure 1, as would be perceived by agents (without geometric relationship).
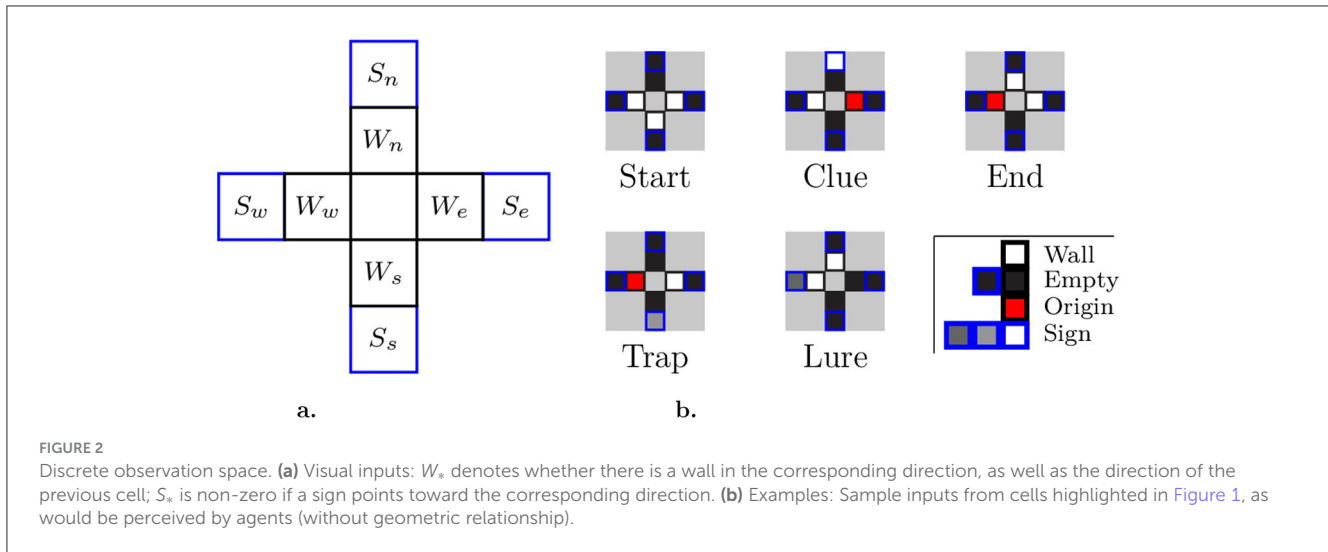
TABLE 2 Elementary rewards for both versions of the reward function (Equation 1): r promotes reaching the goal with a large associated reward, while $\bar{r}$ always indicates an optimal strategy with a cumulative reward $\bar{R} = 1$. l is the number of cells on the optimal path between start and finish.

	ρ_e	ρ_w	ρ_b	ρ_t	Cumulative
r	$2l - 1$	-0.1	-0.2	-1	$R = l$
$\bar{r}$	2	-0.01	-0.02	$-1/(l-1)$	$\bar{R} = 1$

where the cost of c is above zero for cells containing traps and lures.

As illustrated in Figure 3, both metrics cover different regions of the maze space. Surprisingness describes the likelihood of encountering numerous infrequent states while traversing the maze. Conversely, the Deceptiveness focuses on the frequency with which deceptive states may be encountered, that is, it captures how "dangerous" the maze is. With respect to other types of tasks, be it in simulation or the physical world, $S(M)$ accounts for the intrinsic variability of the environment while $D(M)$ would correspond to the ambiguity of said environment e.g., how frequently similar observations can give divergent results. One can see that, by taking advantage of both types of deceptive signs, Complex mazes exhibit the highest combined difficulty and frequency. Furthermore, even with the limitations of discrete inputs, we can here see how it is theoretically possible to generate mazes of arbitrarily high Surprisingness and Deceptiveness. Additional information and the data set on which these analyses are based can be found in the associated Zenodo record (Godin-Dubois, 2024).

4 Training protocol on AMaze

To teach agents generalizable navigation skills, we define a *training* maze, presented in Figure 4a. Although, for simplicity, we only depict one variation of this maze, in practice, the agent is trained on all four rotations (Figure 4c). Thus, the agent will not overfit to a particular upper-diagonal type
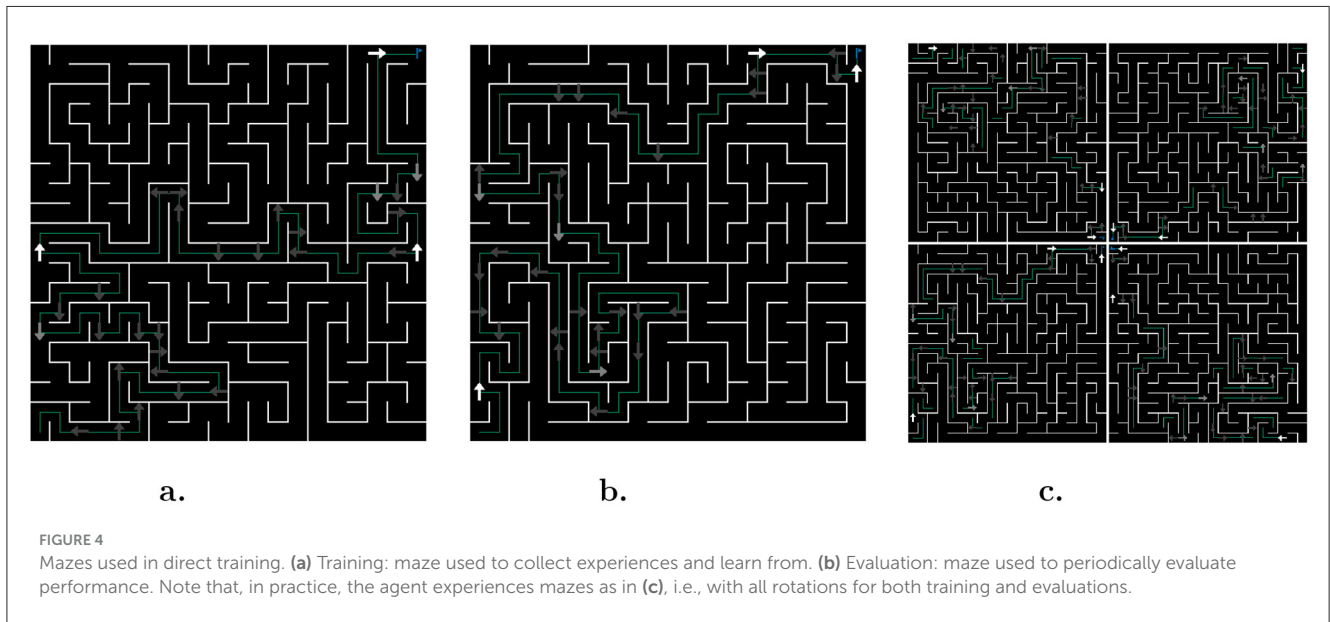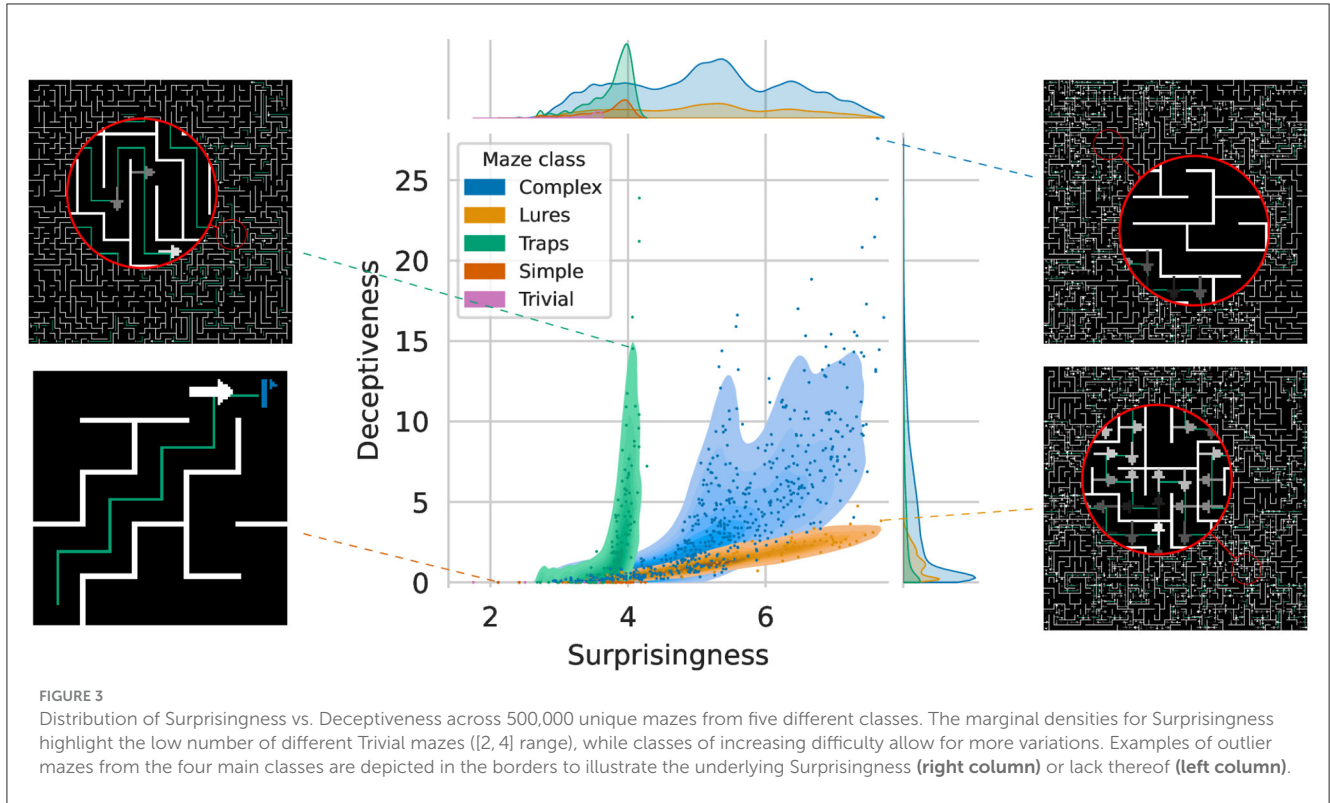
of behavior, but instead will have to develop a context-dependent strategy. Furthermore, intermediate evaluations of the agent's performance are performed in a similar maze (in terms of complexity) with a different seed, as shown in Figure 4b.

To showcase this benchmark's integration with current Reinforcement Learning frameworks, we used Stable Baselines 3 (Raffin et al., 2021) and more specifically their off-the-shelf Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO) algorithms with all hyperparameters kept to their default values (Mnih et al., 2016; Schulman et al., 2017). The total training budget is of 3,000,000 timesteps, divided over the four rotational variations of the training maze with possible early stopping if the optimal trajectory is observed on all evaluation mazes.

4.1 Scaffolding

In addition to this *direct* training in a hard maze, we also followed two incremental protocols: an *interpolation* training, which "smoothly" transitions from simple to more complex mazes, and the *EDHuCAT* training, which leverages human creativity and reactivity (Eiben and Smith, 2015). In the former case, agents start from trivial environments and gradually move onto harder challenges. However, the final mazes on which agents are trained and evaluated are identical to those of the direct case.

Succinctly, every atomic parameter is interpolated between the initial and final mazes' values according to specific per-field rules, e.g. for the apparition of intersections or traps. In this work, the initial maze is unicursive (no intersections) of size 5×5 and eight intermediates are inferred through interpolation. As such a total of ten training stages are performed in this protocol, that is 300,000 timesteps each. In case of early convergence, the remainder of the budget is transferred equally to future stages. For more details, the full spectrum of mazes, in image and textual forms, can be found in Supplementary material 1.

FIGURE 3
Distribution of Surprisingness vs. Deceptiveness across 500,000 unique mazes from five different classes. The marginal densities for Surprisingness highlight the low number of different Trivial mazes ([2, 4] range), while classes of increasing difficulty allow for more variations. Examples of outlier mazes from the four main classes are depicted in the borders to illustrate the underlying Surprisingness **(right column)** or lack thereof **(left column)**.

FIGURE 4
Mazes used in direct training. **(a)** Training: maze used to collect experiences and learn from. **(b)** Evaluation: maze used to periodically evaluate performance. Note that, in practice, the agent experiences mazes as in **(c)**, i.e., with all rotations for both training and evaluations.

4.2 Interactive training

In the interactive setup, we use the Environment-Driven Human-Controlled Automated Training (EDHuCAT) algorithm, loosely inspired by the EDEnS[2] algorithm. As summarized in Algorithm 1, EDHuCAT operates under the joint principles of concurrency (multiple agents evaluated in parallel) and diversity (multiple mazes are generated by the user/experimenter). The advantage of this method over the simple interpolation between initial and final mazes is that it can take advantage of unforeseen developments that occur in the middle of the training. For instance, if the human agent detects that the learning agent has too much difficulty with some newly presented features, they can decide to decrease the difficulty, select from a wider diversity of mazes, or even increase the difficulty. At the same time, the human component makes it harder for the training algorithm itself (A2C or PPO) due to the potential introduction of so-called moving

2 Environment-Driven Evolutionary Selection (Godin-Dubois et al., 2020), used for automated open-ended evolution.

```
input  : K, Number of concurrent environments
         S, number of intermediate steps
         m, an initial maze
         a human
output : EDHuCATed agent
a ← Agent()                    // Create one agent
train(a, m)     // Let it learn from a simple maze
agents ← [a]
for i ← 1 to S − 1 do
    a ← select(agents)       // Select "best" agent
    mazes ← generate(K)       // Create K new mazes
    agents ← copy(a, K)       // Make K copies of A
    for k ← 1 to K do
        // Independent training on different mazes
        train(agents[k], mazes[k])
    end
end
return select(agents)  // Return "best" final agent
```

Algorithm 1. EDHuCAT Algorithm. A human agent is used to perform the *select* and *generate* operations. In this work $K = 3$ and $S = 10$.

targets (see Section 5.4). That is, a Human may not follow a strict policy for choosing mazes or agents, whether between replicates or even during a given run. The total budget is the same as for the other protocols; however, as three concurrent evaluations are performed for each stage, an agent in a given stage is only trained for a maximum of 100,000 time steps.

5 Evaluation of generalized performance

Following the training protocols defined in the previous section, we evaluated the final agents on two complementary tasks to determine whether they had acquired generalized behavior in the target maze class. The first is straightforward: can the agent solve any maze of a given complexity or lower? To answer this, we generated 18 mazes, as shown in Figure 5, based on varying amounts of features (clues, lures, and traps) and Surprisingness. The agents are then evaluated with respect to two goals: their success (do they reach the goal) and their reward (cumulative normalized reward $\bar{R}$, as in Equation 1). Although this allows for comparison between agents based on performance under "normal conditions," this method suffers from cumulative failure: an error at a given time point may preclude any further success. In fact, agents who take a wrong turn somewhere have little information on how to get back on track. Thus, sub-optimal strategies may end up indistinguishable from trivially bad ones.

To counteract this trend, we also perform a complementary evaluation in a more abstract context. Because the input is discrete and thus enumerable, we can generate the complete set of possible input arrays. As we know which is the correct decision, we can assess which inputs are correctly processed by the agents among the four classes: *empty* corridor, corridor with *lure*, intersection with *clue*, and intersection with *trap*. Although less "natural," this

method ensures complete coverage of all the possible situations that an agent may encounter on an infinite number of mazes. Conversely, it also implies that we may be testing an agent on input configurations that it has never seen during training.

The interested reader might also refer to Supplementary material 2 for details on the dynamics of each group's error with the different sign types. The full data for every training dynamics is available in Godin-Dubois (2024).

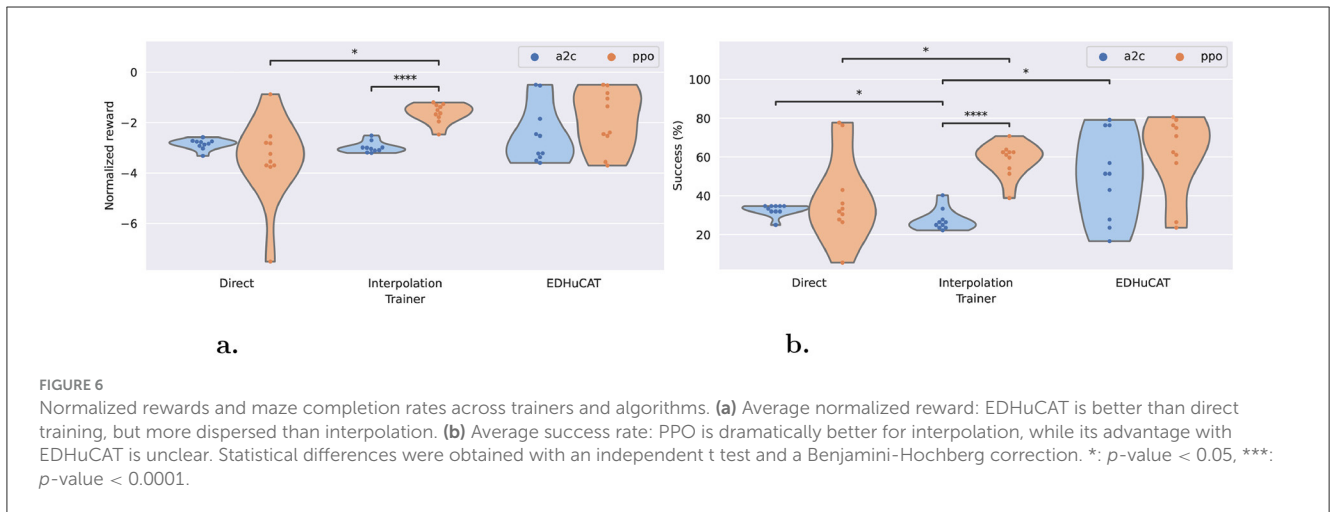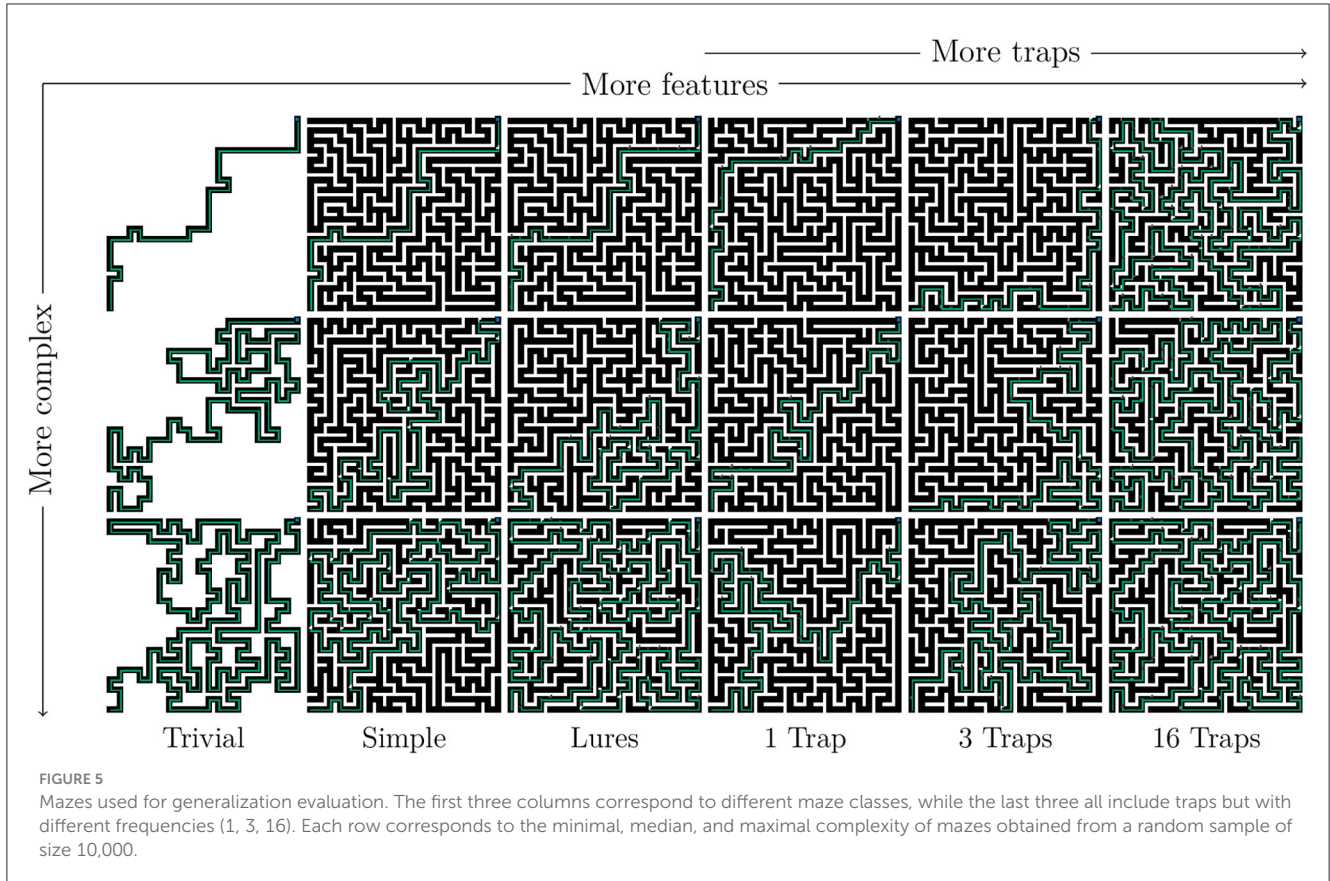## 5.1 Generalized maze-navigation

As summarized in Figure 6, the average cumulative rewards ($\bar{R}$) and the success rates (fraction of mazes whose target was reached) are uniquely distributed according to the training regimen and algorithm. For rewards, both direct and interpolation training have similar trends when using the A2C algorithm (in $[-2, -4]$), while EDHuCAT stands out with a more dispersed distribution. When considering the PPO algorithm, there is a clear negative impact of direct training vs. both alternatives. Although EDHuCAT still presents a higher variance than interpolation training, both generated agents who obtained better rewards. Furthermore, in the latter case, PPO significantly outperforms A2C with a *p*-value < 0.0001 for an independent t-test with Benjamini-Hochberg correction (Benjamini and Hochberg, 1995).

This difference is more clearly visible with the maze completion rate (Figure 6b), especially for agents generated by interpolation training: the best A2C agent is comparable to the worst PPO agent. Again, this is strongly confirmed statistically using the same methodology and with a similar p-value. Additionally, it would seem, from this distribution, that A2C is a slightly better choice in static environments (direct/A2C is marginally better than interpolation/A2C) and conversely for dynamical environments (direct/PPO generally lower than interpolation/PPO). As previously, the human interventions promoted by EDHuCAT do not appear to be beneficial to the PPO algorithm.

5.2 Generalized input-processing

We can make similar observations for the direct input processing test, as illustrated in Figure 7. Selecting the correct action is almost perfectly done by all agents, across all treatments, for the simplest cases (empty corridors and corridors with lures). Surprisingly, the reaction to the presence of a nontrivial sign is handled differently depending on the algorithm. Although PPO seems to be more efficient in detecting clues, A2C shows a better response to traps (Figure 7a). Nonetheless, we can see that, on average, PPO shows clear benefits over A2C (Figure 7b). With this test, we can confirm the advantage of using the former over the latter when facing dynamic environments. The statistical significance is lower than 10^{-3} and 10^{-2} for the interpolation training and EDHuCAT, respectively. In contrast, there is a marginally significant negative trend between A2C use and environmental variability.
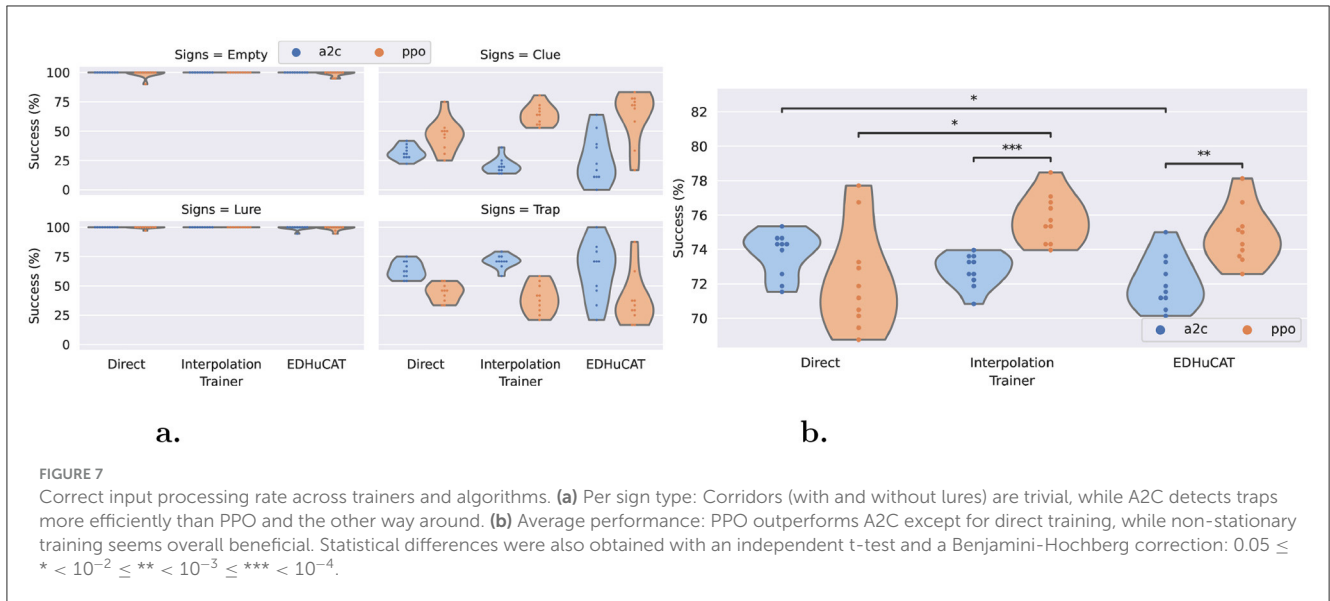
FIGURE 5
Mazes used for generalization evaluation. The first three columns correspond to different maze classes, while the last three all include traps but with different frequencies (1, 3, 16). Each row corresponds to the minimal, median, and maximal complexity of mazes obtained from a random sample of size 10,000.

FIGURE 6
Normalized rewards and maze completion rates across trainers and algorithms. **(a)** Average normalized reward: EDHuCAT is better than direct training, but more dispersed than interpolation. **(b)** Average success rate: PPO is dramatically better for interpolation, while its advantage with EDHuCAT is unclear. Statistical differences were obtained with an independent t test and a Benjamini-Hochberg correction. *: p-value < 0.05, ***: p-value < 0.0001.

5.3 Aggregated performance

To better compare the general performance of all training regimens and algorithms, we provide the maximum and median performance of the six combinations for the three metrics in Table 3. EDHuCAT succeeded in generating the most general maze navigator of all treatments with an average normalized reward of –0.498, compared to –0.873 and –1.2 of direct and interpolation trainings, respectively. Surprisingly, such rewards were obtained with both algorithms, while alternatives fared

much worse when using A2C. Furthermore, it reaches a maze completion rate of 80.6% with PPO and 79.2% with A2C, again taking the lead on direct training (77.8%). Interpolation showed more promise with the input recognition metric, although the low overall variations of this metric preclude additional inferences.

Complementarily, in the context of easily generating general maze-navigating agents, the median performance is useful to highlight which combination of training regime and algorithm was better across replicates. Although slightly less favorable

FIGURE 7
Correct input processing rate across trainers and algorithms. **(a)** Per sign type: Corridors (with and without lures) are trivial, while A2C detects traps more efficiently than PPO and the other way around. **(b)** Average performance: PPO outperforms A2C except for direct training, while non-stationary training seems overall beneficial. Statistical differences were also obtained with an independent t-test and a Benjamini-Hochberg correction: $0.05 \leq * < 10^{-2} \leq ** < 10^{-3} \leq *** < 10^{-4}$.

TABLE 3 Aggregated maximum and median performance by trainer and algorithm. The best values for a row are indicated in bold and second-best in italic. EDHuCAT produced the most general maze navigation agent with respect to normalized rewards and maze completion rates (top rows). Interpolation and EDHuCAT show complementary capabilities to produce better maze navigation *in general* (bottom rows).

	Trainer	Direct		Interpolation		EDHuCAT	
	Algorithm	A2C	PPO	A2C	PPO	A2C	PPO
Maximum	Normalized reward	−2.58	−0.873	−2.51	−1.2	**−0.498**	**−0.498**
	Success rate	0.347	0.778	0.403	0.708	*0.792*	**0.806**
	Input recognition	0.753	0.777	0.74	**0.785**	0.75	*0.781*
Median	Normalized reward	−2.8	−3.39	−3.02	**−1.57**	−2.87	*−1.87*
	Success rate	0.34	0.326	0.257	*0.618*	0.514	**0.667**
	Input recognition	0.743	0.715	0.729	**0.755**	0.717	*0.747*

for EDHuCAT, which is in the top position once and second position twice, the results still speak volumes in favor of nonstationary environments. However, this time around, PPO is clearly identifiable as the algorithm that performs the best, since EDHuCAT also shows a marked bias in its favor.

5.4 Human impact

The previous metrics showed how agents resulting from the EDHuCAT algorithm can have a wide range of performance. To provide a tentative investigation of the reasons for this variability, we classified the decisions made by the human agent into three categories: *Careful*, challenges are slowly integrated once previous ones are solved; *Risky*, the agent is exposed to unfair conditions to promote resilience; *Moderate*, new challenges can be presented even if the agent has not solved the previous ones. The results (in the associated record) show that the *Careful* strategy provides better performance. Agents resulting from both the PPO algorithm and this strategy often end at the top, while agents training with A2C followed an inverse trend.

6 Conclusion and discussion

In this work, we presented a benchmark generator that is geared toward the easy generation of feature-specific mazes and the intuitive understanding of the resulting agents' strategies. The visual cues (either pre-processed or raw) these agents must learn to use to successfully navigate mazes are designed in a CPU-friendly manner so as to drastically limit computational time. By grounding an embodied visual task in what is essentially a succession of lookup-table queries, we allow complex cognitive processes to take place while avoiding the cost of a full robotics simulator. As the agents have only access to local information, this generator is applicable across a broad range of research domains, e.g., from sequential decision making to embodied AI. To help future researchers in manipulating and comparing mazes with widely different characteristics, we introduced two partially orthogonal metrics that accurately capture two key features of such mazes: their Surprisingness and Deceptiveness.

Furthermore, to demonstrate the potential of this generator, we compared the training capabilities of the Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO) algorithms

in three different training regimens with varying levels of environmental diversity. Direct training was a brute-force approach with only a target maze, while the Interpolation case relied on a scaffolding approach presenting increasing challenges. Finally, an interactive methodology (EDHuCAT) was introduced to leverage human expertise as often as possible.

We evaluated the performance of both the maze navigation capabilities of trained agents and their ability to correctly process the entire observation space. Across all these metrics, it was shown that PPO significantly outperforms A2C in dynamic environments, demonstrating the relevance of the former in producing generalized agents. Furthermore, we found that EDHuCAT together with PPO was clearly one step above the alternatives when aiming for *a* general maze-navigating agent. At the same time, if one strives for more than a singular champion but, instead, for reproducible performance, then results point to both the Interpolation and interactive training setups as valid contenders when used in conjunction with PPO.

While demonstrating the potential of AMaze as a benchmark generator for AI agents, this work also raised a number of questions. First, we aim to confirm whether the observed higher performance of PPO is explained by its use of a trust region, which reduces learning speed and, in turn, overfitting. Furthermore, as we limited the study to two RL algorithms and a single neural architecture, many questions remain open with respect to the best choice of hyperparameters or even the applicability of other techniques, such as Evolutionary Algorithms. Second, we only briefly mentioned the impact of the human in the interactive case, and while preliminary data (Godin-Dubois, 2024) show tentative relationships between the human strategy, the training algorithm, and performance, dedicated studies are required to provide definitive answers. The strategy could be studied, as well as additional factors: Do youngsters train better than their elders? Does having a background in AI help? Or can laymen outperform experts?

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://zenodo.org/records/10622914.

Author contributions

KG-D: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology,

Project administration, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. KM: Conceptualization, Formal analysis, Funding acquisition, Methodology, Project administration, Supervision, Validation, Writing – review & editing. AK: Conceptualization, Investigation, Methodology, Supervision, Writing – review & editing.

Funding

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Gen AI was used in the creation of this manuscript.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frai.2025.1511712/full#supplementary-material

References

Alaguna, C., and Gomez, J. (2018). "Maze benchmark for testing evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA: ACM), 1321–1328. doi: 10.1145/3205651.3208285

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern*. 13, 834–846. doi: 10.1109/TSMC.1983.6313077

Beattie, C., Köppe, T., Dúe nez-Guzmán, E. A., and Leibo, J. Z. (2020). DeepMind Lab2D. *arXiv [Preprint]*. arXiv:2011.07027.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., et al. (2016). DeepMind Lab. *arXiv [Preprint]*. arXiv:1612.03801.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: an evaluation platform for general agents. *J. Artif. Intell. Res.* 47, 253–279. doi: 10.1613/jair.3912

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. Roy. Statist. Soc. Ser. B* 57, 289–300. doi: 10.1111/j.2517-6161.1995.tb02031.x

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2020). "Leveraging procedural generation to benchmark reinforcement learning," in *37th International Conference on Machine Learning, ICML 2020*, 2026–2034.

Eiben, A., and Smith, J. (2015). *Introduction to Evolutionary Computing, volume 28*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-662-44874-8

Fujimoto, S., Van Hoof, H., and Meger, D. (2018). "Addressing function approximation error in actor-critic methods," in *35th International Conference on Machine Learning, ICML 2018*, 2587–2601.

Godin-Dubois, K. (2024). AMaze: Fully discrete training with three regimes (direct, scaffolding, interactive) and two algorithms (A2C, PPO). *arXiv [Preprint]*. arXiv:2411.13072v1.

Godin-Dubois, K., Cussat-Blanc, S., and Duthen, Y. (2020). "Beneficial catastrophes: leveraging abiotic constraints through environment-driven evolutionary selection," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 94–101. doi: 10.1109/SSCI47803.2020.9308411

Harries, L., Lee, S., Rzepecki, J., Hofmann, K., and Devlin, S. (2019). "MazeExplorer: a customisable 3D benchmark for assessing generalisation in reinforcement learning," in *2019 IEEE Conference on Games (CoG)* (IEEE), 1–4. doi: 10.1109/CIG.2019.8848048

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). "Deep reinforcement learning that matters," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 3207–3214. doi: 10.1609/aaai.v32i1.11694

Juliani, A., Khalifa, A., Berges, V.-P., Harper, J., Teng, E., Henry, H., et al. (2019). "Obstacle tower: a generalization challenge in vision, control, and planning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, volume August* (California: International Joint Conferences on Artificial Intelligence Organization), 2684–2691. doi: 10.24963/ijcai.2019/373

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaskowski, W. (2016). "ViZDoom: a Doom-based AI research platform for visual reinforcement learning," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (IEEE), 1–8. doi: 10.1109/CIG.2016.7860433

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., et al. (2021). "URLB: unsupervised reinforcement learning benchmark," in *NeurIPS*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., et al. (2016). Asynchronous methods for deep reinforcement learning. *arXiv [Preprint]*. arXiv:1602.01783.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J. (2018). Gotta learn fast: a new benchmark for generalization in RL. *arXiv [Preprint]*. arXiv:1804.03720.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: reliable reinforcement learning implementations. *J. Mach. Learn. Res.* 22, 1–8.

Risi, S., and Togelius, J. (2020). Increasing generality in machine learning through procedural content generation. *Nat. Mach. Intell.* 2, 428–436. doi: 10.1038/s42256-020-0208-z

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

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Syst. Techn. J.* 27, 379–423. doi: 10.1002/j.1538-7305.1948.tb01338.x

Sutton, R., and Barto, A. (2018). *Reinforcement Learning: An Introduction*. Bradford: A Bradford Book.

Synnaeve, G., Nardelli, N., Auvolat, A., Chintala, S., Lacroix, T., Lin, Z., et al. (2016). TorchCraft: a library for machine learning research on real-time strategy games. *arXiv [Preprint]*. arXiv:1611.00625.

Tian, Y., Gong, Q., Shang, W., Wu, Y., and Zitnick, C. L. (2017). ELF: an extensive, lightweight and flexible research platform for real-time strategy games. *arXiv [Preprint]*. arXiv:1707.01067.

Todorov, E., Erez, T., and Tassa, Y. (2012). "MuJoCo: a physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE), 5026–5033. doi: 10.1109/IROS.2012.6386109

Tomilin, T., Dai, T., Fang, M., and Pechenizkiy, M. (2022). "LevDoom: a benchmark for generalization on level difficulty in reinforcement learning," in *2022 IEEE Conference on Games (CoG)* (IEEE), 72–79. doi: 10.1109/CoG51982.2022.9893707

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., et al. (2024). Gymnasium: a standard interface for reinforcement learning environments. *arXiv [Preprint]*. arXiv:2407.17032.

Wang, R., Lehman, J., Clune, J., and Stanley, K. O. (2019). Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv [Preprint]*. arXiv:1901.01753.

Wawrzyński, P. (2009). "A cat-like robot real-time learning to run," in *Adaptive and Natural Computing Algorithms. ICANNGA 2009*, 380–390. doi: 10.1007/978-3-642-04921-7_39

Yu, T., Quillen, D., He, Z., Julian, R., Narayan, A., Shively, H., et al. (2019). Meta-world: a benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv [Preprint]*. arXiv:1910.10897.