NiceWebRL: a Python library for human subject experiments with reinforcement learning environments

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present NiceWebRL, a research tool that enables researchers to use machine reinforcement learning (RL) environments for online human subject experiments. NiceWebRL is a Python library that allows any Jax-based environment to be transformed into an online interface, supporting both single-agent and multi-agent environments. As such, NiceWebRL enables AI researchers to compare their algorithms to human performance, cognitive scientists to test ML algorithms as theories for human cognition, and multi-agent researchers to develop algorithms for human-AI collaboration. We showcase NiceWebRL with 3 case studies that demonstrate its potential to help develop Human-like AI, Human-compatible AI, and Human-assistive AI. In the first case study (Human-like AI), NiceWebRL enables the development of a novel RL model of cognition. Here, NiceWebRL facilitates testing this model against human participants in both a grid world and Craftax, a 2D Minecraft domain. In our second case study (Human-compatible AI), NiceWebRL enables the development of a novel multi-agent RL algorithm that can generalize to human partners in the Overcooked domain. Finally, in our third case study (Human-assistive AI), we show how NiceWebRL can allow researchers to study how an LLM can assist humans on complex tasks in XLand-Minigrid, an environment with millions of hierarchical tasks. The library is available at https://anonymous.4open.science/r/nicewebrl-28BF.

1 Introduction

Recent progress in Artificial Intelligence (AI) increasingly motivates researchers to study AI in the context of human behavior. Some ML researchers aim to improve AI systems by comparing them to humans, since humans can still provide an upper bound on the performance our systems can hope to achieve [26, 31]. For example, Minecraft [14, 15] remains a challenging exploration problem for machines but a fun exploration adventure for people [18, 10]. In cognitive science, there is increased interest in asking whether these machines are human-like [23, 36, 6]. Even if they are not, cognitive scientists are interested in using them as the basis for building human-like machines [23, 8]. In multi-agent RL, many researchers are interested in whether these agents can act as adaptive partners to humans across the wide range of social settings they might be deployed in [5, 29]. This is increasingly relevant with LLMs, as they possess superhuman knowledge and are improving in their "reasoning" abilities [12]. A natural question is how well they can combine their prior knowledge with environment perception to assist us in completing complex tasks. Collectively, these advances could have a wide impact—ranging from robotics [33] to education [19] to healthcare [30]. Clearly, many are interested in building human-like, human-compatible, and human-assistive AI.

Despite this interest in human-centered AI development, pursuing research that integrates human subject experiments with modern ML libraries is currently a cumbersome process. To run experiments with many participants, researchers leverage the internet to get large sample sizes [13]. Thus, most infrastructure is written in the web's programming language: JavaScript [11, 17, 13, 9]. Machine learning code, on the other hand, relies heavily on Python for model development [1, 28, 4] and

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

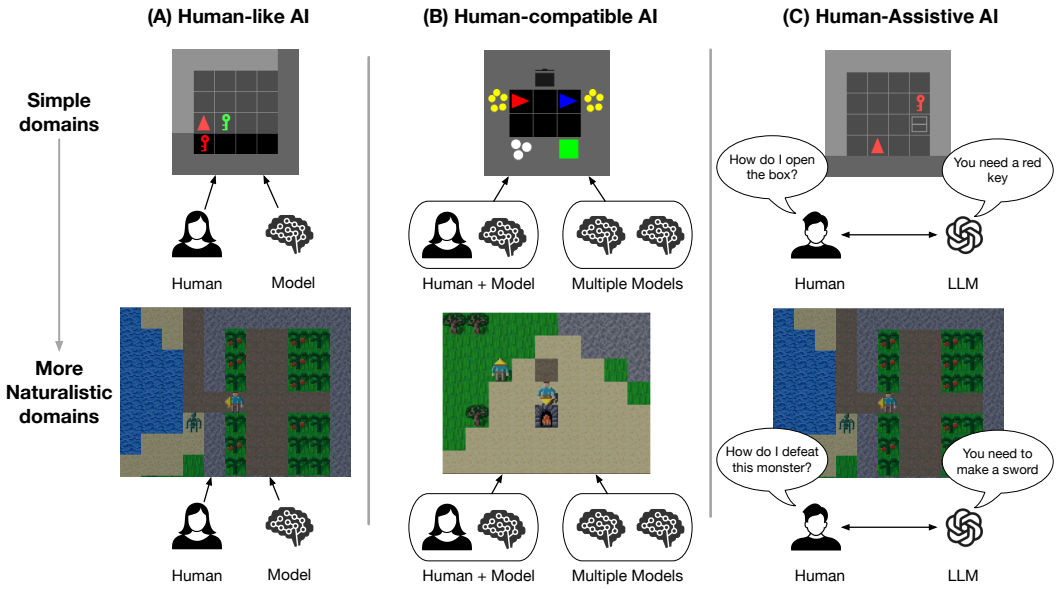

Figure 1: **NiceWebRL is a meta-environment that enables the use of Jax-based environments to develop Human-like, Human-compatible, and Human-assistive AI.** (A) Researchers can compare how humans and AI complete tasks to evaluate if AI behaves in human-like ways. (B) They can study how AI coordinates with humans during task completion to assess if the AI has learned human-compatible social behaviors. (C) They can also integrate Large Language Models into their experiments to evaluate how effectively they combine prior knowledge with environmental perception to assist participants. Importantly, NiceWebRL enables findings to generalize across potentially more complex domains.

variants of C for developing fast environments for simulation [22, 34]. Leveraging ML models or environments for human subject experiments currently requires setting up domain-specific server-client configurations that integrate Javascript, Python, and sometimes C. Doing this for each domain makes the process even more cumbersome.

To address this challenge, we present NiceWebRL: a research tool that lets researchers leverage ML environments for human subject experiments (see Figure 1). Integrating Python with JavaScript requires maintaining a connection between a remote Python-based server and a local Javascript-based client. This distance can cause latency issues when running online experiments. To circumvent this challenge, NiceWebRL exploits Jax [4]—a high-performance numerical computing library—to precompute and cache environment dynamics for arbitrary Jax-based environments. NiceWebRL then acts as a meta-environment for researchers to use arbitrary Jax-based environments in their human subject experiments. Critically, NiceWebRL allows researchers to program experiments entirely in Python by integrating with NiceGUI[1]—a library that enables web developers to specify advanced Graphical User Interface (GUI) components entirely in Python.

1.1 Case study 1: Developing Human-like AI with NiceWebRL

Developing a novel Deep RL cognitive science model with NiceWebRL [8]. A central question in cognitive science is how people represent the environment to enable generalization to new tasks. Successor features (SFs) are a mechanism for how an agent can cache expectations of what it will see when pursuing a policy [2]. Recent ML research has shown that SFs enable agents to repurpose policies for new tasks [3, 7]. Later, cognitive scientists showed that SFs also explained how people reuse prior policies for new tasks [32]. However, the behavioral work was done in a small grid-world with 13 states. Carvalho et al. [8] studied whether SFs could explain how humans reuse behaviors for new tasks in 2 more complex domains: a maze gridworld (Figure 2 A) and Craftax [25], a 2D minecraft domain (Figure 2 D). Across both domains, they set up training tasks where a test object was visible from along the optimal training paths (e.g. top-right corner of Figure 2 A). SFs could not generalize here. People could; however, when people reused a training path to a novel goal, their response times suggested that they were using a caching-based solution rather than something more

[1] https://github.com/zauberzeug/nicegui/

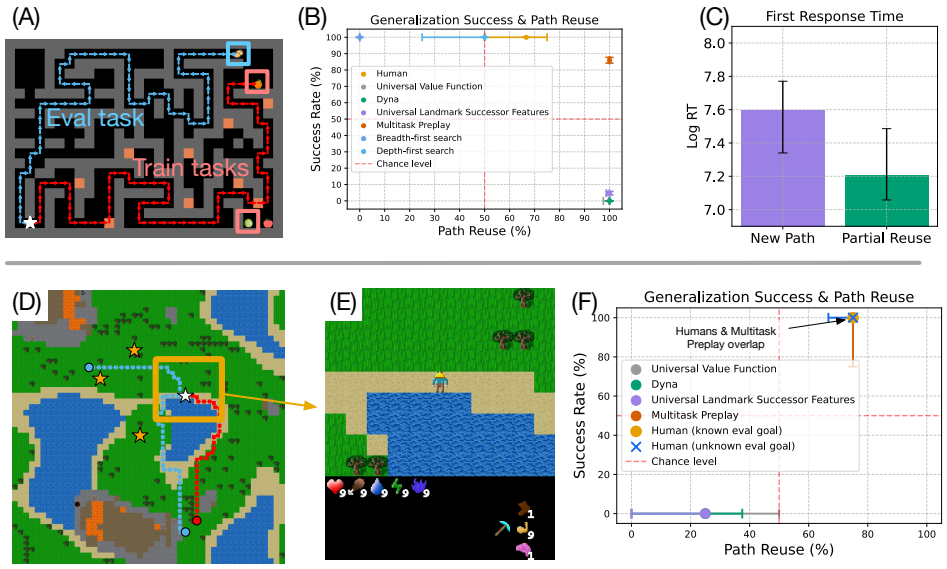

Figure 2: **Case study 1: NiceWebRL enabled the development of a novel Deep RL cognitive model that generalizes to new tasks with the same qualitative behaviors as Humans across multiple domains**. (A, D): a gridworld and 2D minecraft environment that both Human participants and Deep RL models learned in. (B, F): behavioral results studying the same phenomena across the two domains. NiceWebRL enabled developing a Deep RL cognitive model that could both (1) generalize to novel test goals in ways not permitted by previous methods, while (2) doing so with a similar suboptimal path reuse strategy that humans tend to exhibit. Figures reproduced from [8].

flexible—but expensive—like planning at decision-time (Figure 2 B-C). They developed an algorithm termed Multitask Preplay which *preemptively* learns solutions for unpursued tasks nearby training tasks by augmenting experience replay with small amounts of counterfactual simulation. They found this algorithm both better accounted for the response times people exhibited and better predicted how they would reuse prior behaviors. These results generalized to Craftax, where participants and models had to navigate from partial observations of a large world with many objects (Figure 2 E-F).

Role of NiceWebRL. First, NiceWebRL enabled comparing human behavior to advanced Deep RL algorithms including Successor Features. Second, NiceWebRL enabled using the same infrastructure to study both Deep RL algorithms and human behavior in two domains of increasing complexity: a gridworld and Craftax. This helped to ensure that findings were generalizable. Finally, NiceWebRL enabled the measurement of response times. This helped to adjudicate between theories that predict more or less computation at decision-time.

1.2 Case study 2: Developing Human-compatible AI with NiceWebRL

Developing a novel Multi-agent reinforcement learning (MARL) algorithm for coordinating with humans using NiceWebRL [20]. One central question in MARL is how we can develop MARL agents that can generalize to human partners without human training data. One current benchmark for human-compatible AI is the Overcooked domain [5] where agents must coordinate on basic cooking tasks. The state-of-the art algorithm is "Efficient End-to-End Training" [E3T; 35], a "Self Play" algorithm that plays with—and tries to predict the actions of—a noisy variant of itself. Jha et al. [20] developed a novel algorithm, Cross-Environment Cooperation (CEC), where an agent plays only against itself but across millions of different procedurally-generated environments (Figure 3 A). They found that while E3T [35] was able to "succeed" on more episodes when collaborating with humans (Figure 3 B), humans gave that agent a lower rating than CEC (Figure 3 C). The authors asked participants questions about their subjective experience using a Likert scale [24] and found that CEC was rated as more "adaptive" and "human-like"—despite succeeding *less* than E3T. When the authors analyzed game trajectories, they found CEC would collide less with humans across environments.

Role of NiceWebRL. First, NiceWebRL enabled comparing multiple MARL algorithms in their ability to generalize to human partners. Second, NiceWebRL enabled the researchers to collect feedback from participants after every environment interaction stage using the "Feedback" stage

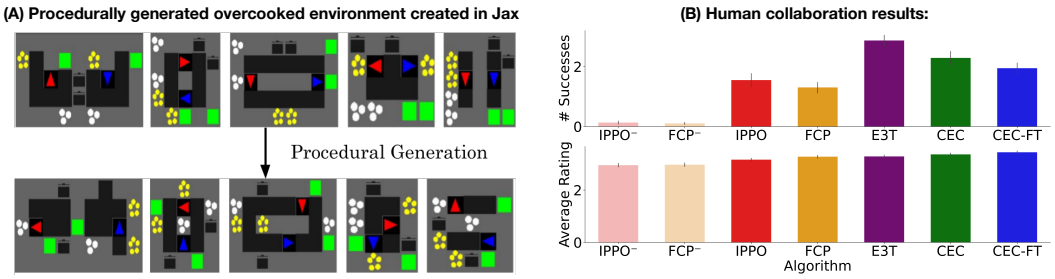

Figure 3: **Case study 2: NiceWebRL enabled the development of a novel MARL algorithm that is more compatible with novel human partners.** (A) A procedurally-generated environment used to design a novel MARL algorithm: Cross-Environment Cooperation (CEC). Prior work had agents learn with diverse sets of agents. CEC has a *single agent* play itself across millions of procedurally-generated environments. (B) While CEC *succeeded less* than other methods when collaborating with humans (top), it succeeded in ways that were *most favorable* to humans (bottom). Analysis suggested showed prior agents succeeded in less collaborative ways. Figures reproduced from [20].

object available in NiceWebRL coupled with NiceGUI's data collection GUI elements. This provided an easy way to get feedback from participants while agent-interaction data was fresh in their memories. Third, NiceWebRL stores all environment interactions so participant episodes could be analyzed post-hoc. This enabled the researchers to analyze trajectories by participants and agents to determine what qualitative behaviors (such as colliding) were different between the different MARL algorithms.

1.3 Case study 3: Developing Human-assistive AI with NiceWebRL

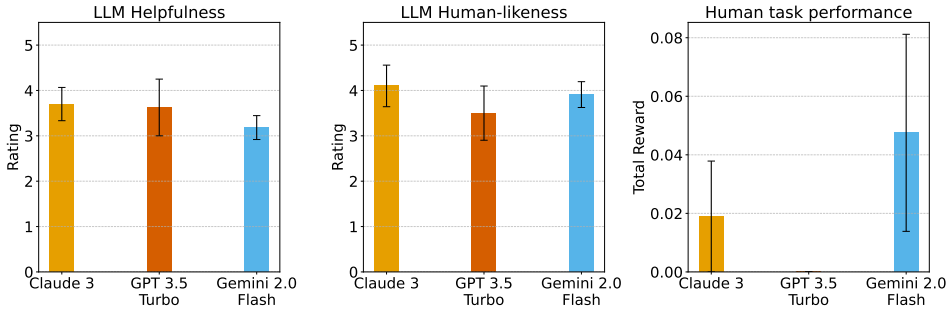

Figure 4: **Case study 3: Proof-of-concept experiment showing that NiceWebRL enables comparing how different LLMs can assist people in completing tasks.** We had Claude 3 Opus, GPT 3.5 Turbo, or Gemini 2.0 Flash act as assistants for people completing tasks in the XLand-Minigrid domain [27]. Each plot is showing the mean and standard error for 10 subjects per model.

Developing an LLM-assistant for sequential-decision making tasks in a virtual environment. We created a simple proof-of-concept experiment where people had to interact with tasks from the Xland-Minigrid domain [27]. To require assistance, they were given no task information but could ask an (anonymous) LLM assistant for help. We present more experiment details in§D.

Role of NiceWebRL. NiceWebRL enabled using an existing ML domain to develop an experiment that studied how an LLM could assist people on long-horizon tasks. Additionally, the Feedback Stage object in NiceWebRL enabled collecting feedback from participants about the LLMs at the end of the experiment. This could also be done after every episode or during episodes.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[2] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[3] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510. PMLR, 2018.

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

[5] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

[6] Wilka Carvalho and Andrew Lampinen. Naturalistic computational cognitive science: Towards generalizable models and theories that capture the full range of natural behavior. *arXiv preprint arXiv:2502.20349*, 2025.

[7] Wilka Carvalho, Andre Saraiva, Angelos Filos, Andrew Lampinen, Loic Matthey, Richard L Lewis, Honglak Lee, Satinder Singh, Danilo Jimenez Rezende, and Daniel Zoran. Combining behaviors with the successor features keyboard. *Advances in Neural Information Processing Systems*, 36, 2024.

[8] Wilka Carvalho, Sam Hall-McMaster, Honglak Lee, and Sam Gershman. Preemptive solving of future problems: Multitask preplay in humans and machines. *arXiv preprint*, 2025.

[9] Joshua R De Leeuw. jspsych: A javascript library for creating behavioral experiments in a web browser. *Behavior research methods*, 47:1–12, 2015.

[10] Yuqing Du, Eliza Kosoy, Alyssa Dayan, Maria Rufova, Pieter Abbeel, and Alison Gopnik. What can ai learn from human exploration? intrinsically-motivated humans and agents in open-world exploration. In *Neurips 2023 workshop: Information-theoretic principles in cognitive systems*, 2023.

[11] Holger Finger, Caspar Goeke, Dorena Diekamp, Kai Standvoß, and Peter König. Labvanced: a unified javascript framework for online studies. In *International conference on computational social science (cologne)*, pages 1–3. University of Osnabrück Cologne, 2017.

[12] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[13] Todd M Gureckis, Jay Martin, John McDonnell, Alexander S Rich, Doug Markant, Anna Coenen, David Halpern, Jessica B Hamrick, and Patricia Chan. psiturk: An open-source framework for conducting replicable behavioral experiments online. *Behavior research methods*, 48:829–842, 2016.

[14] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

[15] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2023. *URL https://arxiv. org/abs/2301.04104*, 2023.

[16] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[17] Felix Henninger, Yury Shevchenko, Ulf K Mertens, Pascal J Kieslich, and Benjamin E Hilbig. lab. js: A free, open, online study builder. *Behavior Research Methods*, pages 1–18, 2021.

[18] Larissa Hjorth, Ingrid Richardson, Hugh Davies, and William Balmford. *Exploring Minecraft: Ethnographies of play and creativity*. Springer Nature, 2021.

[19] Wayne Holmes and Ilkka Tuomi. State of the art and practice in ai in education. *European journal of education*, 57(4):542–570, 2022.

[20] Kunal Jha, Wilka Carvalho, Yancheng Liang, Simon S Du, Max Kleiman-Weiner, and Natasha Jaques. Cross-environment cooperation enables zero-shot multi-agent coordination. In *International Conference on Machine Learning (ICML)*, 2025.

[21] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[22] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[23] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

[24] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[25] Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2024.

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

[27] Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Artem Agarkov, Viacheslav Sinii, and Sergey Kolesnikov. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Nicolas Gribonval, Zachary Devito, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8024–8035, 2019.

[29] Stuart Russell. Human-compatible artificial intelligence., 2022.

[30] Mohammed Yousef Shaheen. Applications of artificial intelligence (ai) in healthcare: A review. *ScienceOpen Preprints*, 2021.

[31] Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

[32] Momchil S Tomov, Eric Schulz, and Samuel J Gershman. Multi-task reinforcement learning in humans. *Nature Human Behaviour*, 5(6):764–773, 2021.

[33] Sai H Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities. *Ieee Access*, 2024.

[34] Tom Ward, Andrew Bolt, Nik Hemmings, Simon Carter, Manuel Sanchez, Ricardo Barreira, Seb Noury, Keith Anderson, Jay Lemmon, Jonathan Coe, et al. Using unity to help solve intelligence. *arXiv preprint arXiv:2011.09294*, 2020.

[35] Xue Yan, Jiaxian Guo, Xingzhou Lou, Jun Wang, Haifeng Zhang, and Yali Du. An efficient end-to-end training approach for zero-shot human-ai coordination. *Advances in Neural Information Processing Systems*, 36:2636–2658, 2023.

[36] Lance Ying, Katherine M Collins, Lionel Wong, Ilia Sucholutsky, Ryan Liu, Adrian Weller, Tianmin Shu, Thomas L Griffiths, and Joshua B Tenenbaum. On benchmarking human-like intelligence in machines. *arXiv preprint arXiv:2502.20502*, 2025.

A Formal domain description

Jax functions automatically compile to a fixed behavior when they receive their first input data. As such, if one wants different domain functionality across different *contexts* (e.g. training vs. testing), the domain's functions typically need a "`env_parameter`" argument. Thus, Jax-based domains are naturally formulated as Partially Observable Contextual Markov Decision Processes (POCMDPs) $\mathcal{M}_c = \langle S, \mathcal{A}, \mathcal{X}, \mathcal{C}, \rho, P, R, O \rangle$ [16, 21]. Here, S denotes the environment state space, $\mathcal{A}$ denotes its action space, $\mathcal{X}$ denotes (potentially partial) observations of the environment, and $\mathcal{C}$ denotes a space of contexts that an MDP can be in. `env_parameter` then corresponds to an MDP's context $c \in \mathcal{C}$. It can be used to augment the initial state distribution $\rho_c(s_0)$ (e.g. having an agent start in different states in different contexts), the transition probabilities, $P_c(s'|s,a)$ (e.g. an agent's speed or strength can be changed in different contexts), the reward function $R_c(s)$ (e.g. different objects can be rewarded in different contexts), or the observation function $O_c(s)$ (e.g. objects can take on different colors in different contexts).

An episode proceeds as follows. An initial state $s_0 \in S$ is sampled from the initial state distribution $\rho_c(s_0)$. When an agent takes an action $a \in \mathcal{A}$ in state $s \in S$, the next state s' is sampled according to a next state distribution $s' \sim P_c(\cdot|s,a)$. The agent then receives an observation $x' = O_c(s')$ and reward $r' = R_c(o)$. Note that c is typically fixed within an episode.

	Server-side Operations		**Client-side Operations**
	Input: env context parameters c		
	At time $t = 0$:		
1:	$s_0, o_0 =$ env.reset(c)	1:	
2:	$\{(s_1, o_1) =$ env.step$(s_0, a, c)\}_{a \in \mathcal{A}}$	2:	
3:	cache $s_{\texttt{next}} = \{s_1\}$	3:	
4:	send o_0 and $o_{\texttt{next}} = \{o_1\}$ to the client	4:	display o_0 and record time t_1
5:		5:	cache $o_{\texttt{next}}$
6:		6:	participant selects action a
7:		7:	record time t_2
8:		8:	send a_0, t_1, t_2 to the server
9:		9:	select $o_1 \in o_{\texttt{next}}$ corresponding to a_0
10:		10:	display o_1 and record t_1
	At time $t = 1, 2, \ldots$:		
11:	receive and store (a_{t-1}, t_1, t_2)	11:	
12:	select $s_t \in s_{\texttt{next}}$ corresponding to a_t	12:	
13:	$\{s_{t+1}, o_{t+1} =$ env.step$(s_t, a, c)\}_{a \in \mathcal{A}}$	13:	
14:	send $o_{\texttt{next}} = \{o_{t+1}\}$ to the client	14:	cache $o_{\texttt{next}}$
15:	update $s_{\texttt{next}} = \{s_{t+1}\}$	15:	participant selects a_t
		16:	record time t_2
		17:	send a_t, t_1, t_2 to the server
		18:	select $o_{t+1} \in o_{\texttt{next}}$ corresponding to a_t
		19:	display o_{t+1} and record t_1

Figure 5: **Server-client Human-Environment Interaction Protocol.** Note that we omit displaying reward r due to space constraints.

Let `env` be the programmatic object representing a domain. In our library (and many RL libraries), $s_0, o_0 =$ `env.step`(c) essentially plays the role of sampling from the initial state distribution and computing the corresponding observation for the agent. The standard practice is to have $s_{t+1}, o_{t+1}, r_{t+1} =$ `env.step`(s_t, a_t, c) implement (a) sampling a new state (b) computing the corresponding reward, (c) computing the observation that an agent will get.

B Descriptions of stage types

Currently, there are three basic stage classes, though more can easily be added.

1. `Stage`: used to display instructions or information to a participant.
2. `FeedbackStage`: used to collect information from participants. Typically involves an interactive screen that does *not* interact with the environment.
3. `EnvStage`: used to interact with an environment. It takes as input an environment and environment parameters. We describe how NiceWebRL uses this abstraction to have a remote server-side program display images to one's local web-browser client in Figure 5.

We present examples of each in Figure 6.

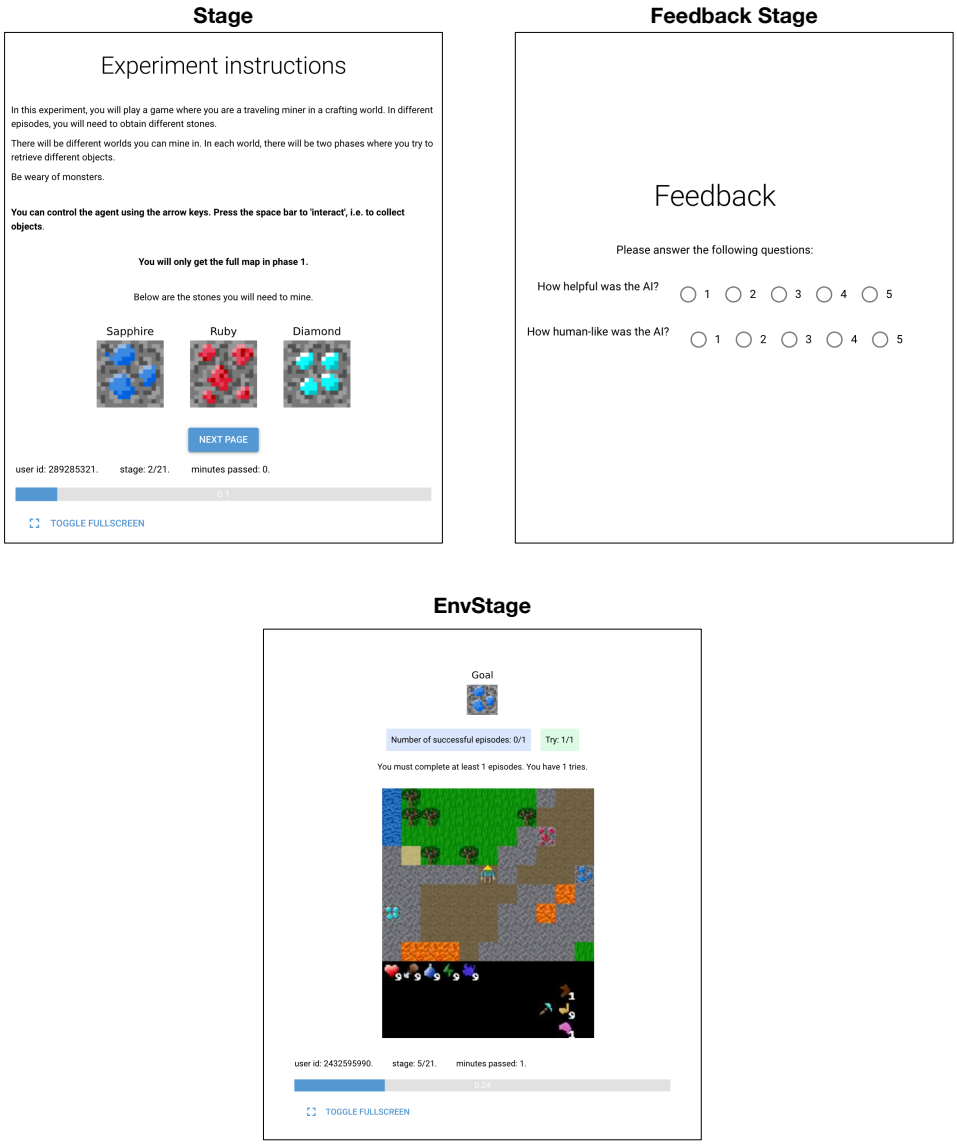

Figure 6: **Examples of different kinds of stages.**

C Computing resources

For details on case study 1 or 2, please see [8] or [20], respectively. For case study 3, experiments were conducted using computing infrastructure from the `fly.io` platform with the "performance-2x" configuration. This is a machine with 4GB of RAM. The machine had no GPU. Even in this setting, Jax's compilation features provide a significant speed up to environment computation.

D Human subject experiment details

Our study is approved by the University IRB. All subjects were recruited with https://www.cloudresearch.com/ and provided informed consent. We provide the consent form in the GitHub example. Participants were compensated $4 for completing the task. The average task completion time was 23.33 minutes. At the beginning of each experiment, the participants provided demographic information (age and gender, coded as male or female).

At the beginning of the experiment, users were randomly assigned either Claude 3 Opus, GPT 3.5 Turbo, or Gemini 2.0 Flash. We set up our server so that it would interact with the LLMs via API calls[2]. Importantly, the LLM assistants were given text descriptions of the ground truth environment state including information on: (1) the goal of the episode (2) the locations and identities of all objects in the environment (3) the rules of the environment (e.g. how objects interact when combined). In principle, this can enable them to help users figure out the goal to maximize task reward. In this proof-of-concept, each participant completed 3 episodes, where a new task was sampled per episode. After all 3 episodes, participants answered two questions on a 5-point scale, "How helpful was the AI?" and "How human-like was the AI?". We collect data from 30 participants via CloudResearch. For this proof-of-concept, we used older models with cheap API calls. We don't expect that these results are representative of what is possible with frontier models.

[2]we provide an example of how to set up a web experiment with a local LLM in our examples folder