L2Explorer: A Lifelong Reinforcement Learning Assessment Environment

Erik C. Johnson, Eric Q. Nguyen, Blake Schreurs, Chigozie S. Ewulum, Chace Ashcraft, Neil M. Fendley, Megan M. Baker, Alexander New, Gautam K. Vallabha
Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 21045

Abstract

Despite groundbreaking progress in reinforcement learning for robotics, gameplay, and other complex domains, major challenges remain in applying reinforcement learning to the evolving, open-world problems often found in critical application spaces. Reinforcement learning solutions tend to generalize poorly when exposed to new tasks outside of the data distribution they are trained on, prompting an interest in continual learning algorithms. In tandem with research on continual learning algorithms, there is a need for challenge environments, carefully designed experiments, and metrics to assess research progress. We address the latter need by introducing a framework for continual reinforcement-learning development and assessment using Lifelong Learning Explorer (L2Explorer), a new, Unity-based, first-person 3D exploration environment that can be continuously reconfigured to generate a range of tasks and task variants structured into complex and evolving evaluation curricula. In contrast to procedurally generated worlds with randomized components, we have developed a systematic approach to defining curricula in response to controlled changes with accompanying metrics to assess transfer, performance recovery, and data efficiency. Taken together, the L2Explorer environment and evaluation approach provides a framework for developing future evaluation methodologies in open-world settings and rigorously evaluating approaches to lifelong learning.

In recent years, Deep Reinforcement Learning (DRL) approaches have begun to deliver powerful results for a variety of compelling domains, including games such as Chess, Go, and Shogi (Silver et al. 2018); Atari video games (Mnih et al. 2013); more complex strategy video games (Berner et al. 2019; Vinyals et al. 2019); and dexterous robotic manipulation (Rajeswaran et al. 2017). Despite the groundbreaking success in training autonomous agents, resulting policies tend to be very brittle and generalize poorly (Chan et al. 2019). When presented with a new task or a task variant, DRL approaches are susceptible to a performance drop (Zhang et al. 2018; Kirk et al. 2021) due to the catastrophic forgetting problem (French 1999; McCloskey and Cohen 1989), which may not be overcome by domain randomization strategies alone. As the field moves from environments which are fixed to evolving, open-world scenarios, current DRL approaches will be insufficient.

This performance gap has led to an interest in *Continual Learning*, which seeks to design algorithms to learn over sequences of tasks. In the related, but broader, concept of *Lifelong Learning* (Chen and Liu 2018), an agent learns over a lifetime of experiences (see Fig. 1) in an evolving environment (for purposes of this paper, however, we treat *continual learning* as synonymous with *lifelong learning* as our approach is applicable to both concepts). Much recent work has been on supervised classification under distribution shifts (Song et al. 2020) and learning a sequence of tasks (Parisi et al. 2019; Hsu et al. 2018). Continual RL (Khetarpal et al. 2020) seeks to create agents which can maintain performance in the face of nonstationary distributions.

A key issue in developing Continual RL algorithms is how to assess performance in a rigorous and informative way. There are existing attempts to address this with procedurally generated open worlds (Risi and Togelius 2020), which generate tasks using randomly parameterized environments. A recent software framework has begun integrating metrics, baselines, and environments for Continual RL (Powers et al. 2021). We argue that successful Continual RL assessment requires both sufficiently complex environments and highly reconfigurable tasks, as well as carefully structured experiments and metrics. We must also consider a multi-dimensional approach to assessment (in terms of raw performance, generalization, task transfer, and data efficiency).

To help address this need, we have developed Lifelong Learning Explorer (L2Explorer), a first-person-view (FPV), highly configurable Unity^{TM} environment[1]. The environment allows for procedural generation through Python code for open-ended task and task variant definitions. Moreover, the environment is set up for testing lifelong learning curricula with a set of lifelong learning metrics. While the Unity environment itself is designed for Continual RL assessment, this work also outlines an approach for taking complex, reconfigurable open-world environments and creating rigorous evaluation for lifelong learning algorithms.

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
DISTRIBUTION STATEMENT A. Approved for public release. Distribution Unlimited.

[1]`https://unity.com/`

Existing Environments for Continual RL

Many environments have been used to test DRL approaches, including game environments. Extensive work, including transfer learning studies, have been conducted with Atari games (Mnih et al. 2013). This has been generalized into a meta-learning framework that allows sampling of Atari-like games (Staley et al. 2021). Complex strategy games have also been leveraged in this context, including Starcraft 2[2], team-based play in Dota 2 (Berner et al. 2019), and rogue-likes such as Dungeon Crawl Stone Soup (Dannenhauer et al. 2019). While impressive, many games can be limited for Continual RL testing due to the lack of full configurablility.

Simulators of real-world systems, such as autonomous vehicles, have also been extensively used. Two notable examples are the CARLA simulator (Dosovitskiy et al. 2017) for autonomous driving and the AirSim package for unmanned aerial vehicles (Shah et al. 2018). While these include photorealistic images and realistic physics, the computational complexity of the environments coupled with the design time required to produce tasks and task variants can limit their applicability to Continual RL testing.

There are also many FPV environments, which combine some of the desirable features of game environments and physical simulators. These are valuable for continual and lifelong learning testing because they allow natural inclusion of partial observations, multiple observation modes, and proxies of real world tasks. These strengths come at the downside of reduced task complexity and reduced fidelity. Examples include the use of the Unity Environment through the ML-Agents package (Juliani et al. 2018) and the procedurally generated Obstacle Tower (Juliani et al. 2019) environment. The FPV game DOOM forms the basis of the Vizdoom FPV environment (Kempka et al. 2016). Also related are the real-world home and robot environment simulators such as AI2Thor (Kolve et al. 2017), which also introduce semantic object relationships into FPV environments.

In order to introduce the flexibility required for open-world learning scenarios, increasing emphasis has been placed on procedurally generated environments. These include the open-world game environments of XLand (Team et al. 2021) and MiniHack (Samvelyan et al. 2021). These represent major steps forward, but the lack of controllable parameters limits the precision of testing that can be done in these environments (Kirk et al. 2021).

Our environment and assessment framework, L2Explorer, seeks to combine the strengths of different types of environments. The use of Unity allows for a visually complex world, and ML-Agents exposes hooks that allow for principled procedural generation using flexible Python code. The environment has lower computational complexity than simulators of autonomous cars or aerial vehicles. Finally, the critical component is linking this procedurally generated environment to careful experimental design and metrics, similar to CoRA (Powers et al. 2021). We aim to create an approach that will also generalize and improve Continual RL assessment utilizing other open-world environments.

[2]`deepmind.com/research/open-source/pysc2-starcraft-ii-learning-environment`

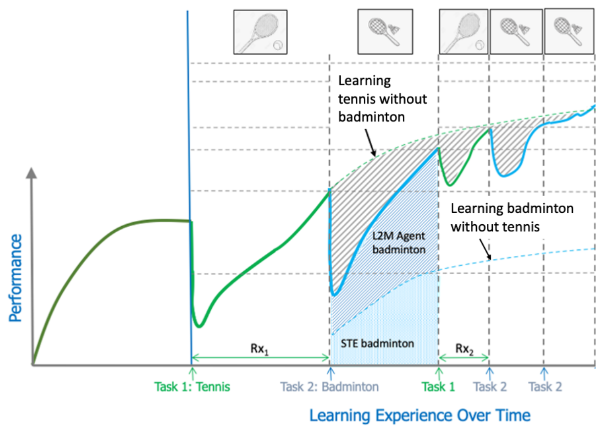

Figure 1: Key considerations when assessing lifelong learning agents (a Lifelong Learning Machine, L2M) with open-world environments. An agent's lifetime is broken into blocks of tasks (e.g., playing an individual sport) for training and evaluation. Blocks generate metrics for assessing performance. Our approach utilizes a highly flexible environment to generate curricula (lifetimes), tasks, and task variants. A multi-dimensional approach to metrics, going beyond raw performance, is required to capture the nuances such as transfer and performance recovery. Controlled sequences of tasks can elucidate particular strengths and weaknesses in learners. Figure adapted from (New et al. 2022).

L2Explorer Design Approach

Similar to previous efforts to define key requirements for lifelong learning evaluations (Farquhar and Gal 2018), we developed key criteria for our framework which allow for rigorous assessment of lifelong learning with an open-ended, procedurally generated environment. We believe the key criteria of our framework to be:

- Flexible description of multiple tasks: A programmatic API to specify new tasks with a common specification
- Flexible control over task variants: Selection of key variables that can be modified within a task definition
- Notions of task relationships/similarity: While there is no agreed upon approach to measure generic task similarity for RL tasks, some notion of task similarity is required to appropriately structure evaluation
- Control over degree of similarity: The environment needs to be able to control the degree of similarity between configurations to allow for abrupt and gradual transitions
- Parametric and random variation: The tasks require the notion of parameters used to deterministically create variants, as well as intrinsic parameters that can be randomly sampled, similar to the previous work (Kirk et al. 2021)
- Targeted testing curricula: A set of designed curricula and tests, along with integrated metrics and baselines

Taken together, these factors address some of the limitations that come with testing with open-world, procedurally generated environments.

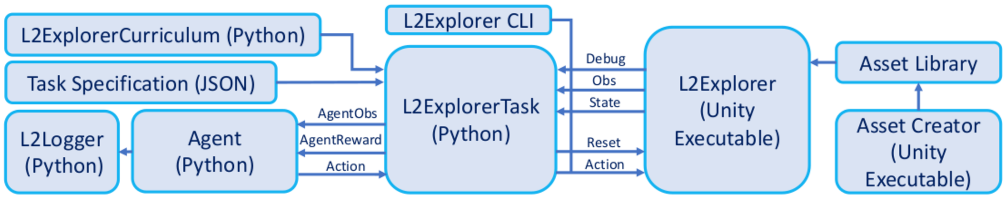

Figure 2: Overview of L2Explorer Software system, consisting of a suite of Python tools which interact with a custom Unity environment. The Python suite allows for execution of testing curricula, integration of standard agent code, and rapid reconfiguration of the unity environment through the reset channel. Logging and metrics computations are integrated. A custom asset creator allows new Unity models to be incorporated into the framework to maximize extensibility.

Continual RL Assessment

In L2Explorer, we build on the standard formalism for RL agents using Partially Observable Markov Decision Processes (POMDPs). These consist of the tuple $M = (S, A, O, R, T, \phi, p)$, where S is the state space, A is the space of actions available to an agent, $R : S \times A \times S \to \mathbb{R}$ is the reward function, $T(s'|s, a)$ is the state transition function, and p is the distribution of initial states. The system is partially observable as the state is not directly available to the agent, which instead gets an observation from the observation space O generated by the emission function $\phi : S \to O$. The goal of the RL agent is to learn a policy $\pi(a|s)$.

$$\pi^* = \text{argmax}_{\pi \in \Pi} \mathbb{E}_{s \sim p(s_0)}[R(s)]$$

where $R(s)$, which is a real value, is the total expected reward gained by the state.

We consider nonstationarities in the underlying POMDP for both tasks and tasks variants. Of critical consideration are S, A, $T(s'|s, a)$, R, O, and ϕ. In continual reinforcement learning, these are allowed to vary as a function of the number of episodes to create a nonstationary POMDP.

To expose agents to nonstationarity in a meaningful way, controlled variation in environment is required. It is rarely practical to fully quantify distributions in high dimensional POMDPs, so instead we seek useful and practical surrogates for controlled manipulation of nonstationary POMDPs. We propose that each curricula should have some desired testing hypothesis relating to changing distributions in the underlying POMDP. Different tasks and task variants can represent particular distributions in the POMDP tuple, and altering parameters results in a shift in these distributions.

Achieving this requires some notion of task and variant similarity. We propose exploiting the parameters of procedural generation to create heuristics of similarity. For example, if two task differ only by the probability distribution used to place objects, a distance metric applied to the tasks' distributions could be a proxy measure for the similarity. A key caveat, however, is that heuristic notions of similarity may not be inherently related to agent transfer. What seems intuitively similar to a human may not correspond to positive transfer between tasks for an RL algorithm, and tasks that may be similar for one algorithm may not be for another (Carroll 2005). Research into measures of similarity, transfer, and generalization is ongoing (Barreto et al. 2016; Ma et al. 2020) and beyond the scope of this work.

L2Explorer variants are designed by selecting a subset of procedural generation parameters that must be fixed, and a subset that may be fixed or allowed to be sampled from a distribution. Differences in the parameter values may be passed into the appropriate similarity heuristics. Tasks are assembled into curricula with particular experimental goals, such as determining sensitivity to changes in reward space, the observation emission function, and so forth. This overall approach will allow for meaningful experimental design and overcome limitations of completely randomly generated environments (Song et al. 2020; Kirk et al. 2021).

Curriculum Design Considerations

We utilize existing a lifelong learning evaluation framework for curriculum design, as specified in (New et al. 2022) and released open-source[1]. This can be seen in Fig. 1, where an agent is pretrained and then deployed on a sequence of tasks. A lifelong learner should be able to take such a fixed curriculum and demonstrate learning over deployment. This framework allows for pre-deployment training or parameter selection before learning. The agent is then subjected to a series of units of experience (e.g., rounds in a game) that can be grouped into longer blocks. We assume blocks can incorporate drift in the underlying distributions of the POMDP. Periodically, the agent can be frozen and have its capabilities tested in a separate evaluation block. Individual environment metrics can be specified, then aggregated into environment-agnostic lifelong learning metrics (detailed below). In L2Explorer, similarity heuristics can be utilized to order tasks and task variants into sequences of learning and evaluation blocks.

Metrics

L2Explorer utilizes an existing set of lifelong learning metrics that characterizes the nuances of the performance curves such as the one in Fig. 1. Full definitions can be found in (New et al. 2022), with open-source implementations available. Task-specific performance measures (e.g., total reward, time to completion) generated during learning and evaluation blocks enable computation of:

[1]`https://github.com/darpa-l2m/l2metrics`, `https://github.com/darpa-l2m/l2logger`

```
"environment_params": {
   "bounding_wall_color": [0, 0.5, 1.0],
   "light_intensity": 3.0,
   "predefined_map": 0,
   "map_size": [100, 100]
},
"agent_params": {
   "action_model": "byvelocity",
   "agent_height": 1.0,
   "agent_width": 1.0,
   "max_linear_speed": 10.0,
   "max_angular_speed": 90.0,
   "observation_size": 84,
   "coordinates": [10.0, 10.0],
   "heading": 0.0,
   "pickup_range": 5.0
},
"objects": [{
   "model": "1",
   "class": "geometric",
   "coordinates": [10.0, 8.0],
   "color": [0.0, 0.0, 1.0],
   "motion_model": "stationary",
   "reward_stimulus": ["AgentCollide", "AgentInteract"],
   "destroy_stimulus": ["AgentInteract"],
   "reward": 100.0
}]
```

Figure 3: Example subset of a reset JSON object. This shows the available variables to procedurally generate variants of the environment, agent, and objects. Each object can have independently specified coordinates and interaction models, and objects can be created dynamically within an episode. On each reset call, a new JSON can be loaded into the environment.

- Performance Maintenance: A measure of catastrophic forgetting for a given task
- Forward Transfer: A measure of learning improvement for a task, given learning experiences on a different task
- Backward Transfer: A measure of improvement on a previously learned task given learning experiences on a new task
- Performance Relative to Single-Task Expert: Comparison of lifelong learning agent to an expert agent trained only on the task in question (e.g., the dashed lines in Fig. 1).
- Sample Efficiency: A measure of the experience required for a lifelong learning agent to reach maximal performance relative to a single task expert

Together, the metrics, curriculum design, and design philosophy inform the development of the L2Explorer environment. Lessons from this approach can be generalized to assessment with future open world exploration environments.

L2Explorer

L2Explorer is a software framework and testing environment which implements this overall design vision, including integration with metrics and testing curricula. The project builds on Unity ML-Agents (Juliani et al. 2018), which exposes key parameters to create a reconfigurable, 3D, FPV reinforcement-learning environment. Our custom Unity environment enables exploration and interaction with a world specified through a JSON object format. Additional side channels allow communication of observations, actions, and debugging information. The Python code specifies the world JSON objects, allows for creating testing curricula, integrates metrics, and provides a Gym-compatible interface (Brockman et al. 2016). An overview of the software can be seen in Fig. 2, and is intended to be released open-source. The package will provide testing curricula and baselines to assist development, and will allow users to fully customize tasks, task variants, curricula, and even import custom 3D models into the environment.

Unity Executable

The core environment of L2Explorer is a custom Unity app which instantiates a simulated world in response to a JSON specification communicated through a Unity ML-Agents side channel. This is activated when the reset function is called, allowing each learning experience to take place in a custom specified world. A partial example of a reset JSON can be seen in Fig. 3. The format allows for specification of environment specific parameters, including lighting conditions, backgrounds, agent specific parameters, and object specific parameters. Pre-caching 3D models enables rapid world construction on reset.

The Unity environment exposes three custom communication channels seen in Fig. 2. These allow the reset functionality, querying the current state of the world (returned in the specified world JSON format), and sending debugging information. Standard channels allow for the communication of actions (continuous or discrete) and observations (state vectors and visual observations). Visual observations include RGB images, grayscale depth images, and semantic segmentation maps (segmented by the object "class").

The agent model is, by default, a kinematic model controlled by a linear and angular velocity term. Several agent-object interaction models are specified. These can require an agent to be in a nearby zone, require contact with the object, or require the agent to take an "interact" action. Interaction can result in a positive or negative reward, and can result in the destruction of the object. There is dynamic respawning through the side channels, allowing new objects to be specified during runtime. For certain task designs, this can be key for learning with fixed episode step length.

A separate unity executable is provided to allow users to prepare and save custom 3D models in an L2Explorer compatible format. These assets, once prepared, can be loaded in during a reset action. This can allow for continued extensibility of the environment.

Python Wrappers

In L2Explorer, Python wrappers allow for task specification and agent development. The primary class, L2ExplorerTask, is a wrapper for communicating with the Unity executable. This class processes the raw observations to provide a reward and observation to the agent. This allows for definitions of alternative rewards functions in Python, such as soft rewards, as well as any data filtering required for a task.

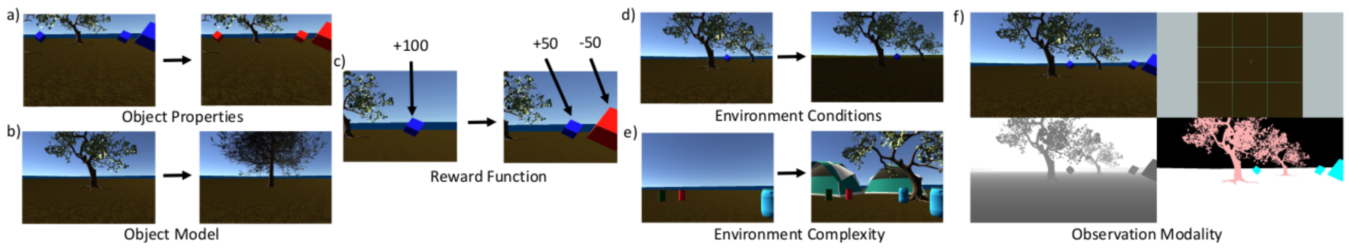

Figure 4: Variants which can be produced using the standardized JSON format through the reset function. Panel a) shows how object properties can be manipulated, in this case turning blue targets into red ones. Panel b) shows the replacement of an object of the same class with a different model, in this case swapping two trees. Panel c) shows how the reward structure can be modulated with one reward object of value 100 becoming two reward objects with values 50 and -50. Panel d) shows the manipulation of environmental conditions through changes in lighting. Panel e) shows how scene complexity can be varied through placement of additional objects from different classes. Finally, Panel f) demonstrates how different observation modalities including RGB, depth, and a semantic segmentation can be supplied to an agent.

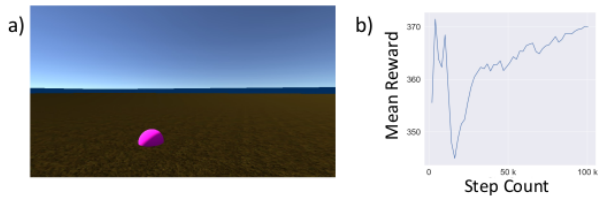

Figure 5: Example task demonstrating integration of a Stable Baselines PPO agent for learning a simple object finding task. Panel a) shows the visual input to the agent and Panel b) an example learning curve. Standard Python agents using the established Gym API can be rapidly tested in this framework.

The Unity environment is configured during a reset function, utilizing the JSON specification (Fig. 3). This enables the specification of curricula in Python, as well as the specification of procedural generation. Curricula are specified through task-specific classes and a task-specific initial JSON file. Learning and Evaluation Blocks can then be described as above, along with variants of the task. The base JSON file for the task is passed to the task class, and key-value pairs are modified, if required, to create a variant. Modifications are drawn from discrete or continuous probability density functions to create appropriate randomization. The Gym-compatible environment class allows for integration of standard baselines and custom algorithms. The agent observation space, O, is a tensor of number of pixels by number of pixels by number of channels in the selected modalities, and optionally state vector of size N. The action space a is a 2-dimensional continuous valued vector (linear and angular velocity), which can optionally be discretized. A third dimension can be added for the binary “interact” action.

In addition to the core functionality, integration with a Python logging package and Python metrics package for lifelong learning is provided to enable the multi-dimensional metrics evaluation. A command line interface (CLI) is also provided for directly interfacing with and debugging the environment.

Example Use Cases

L2Explorer is a highly configurable and extensible environment that can provide insight into the performance of Continual RL algorithms. Towards this end, we present some example tasks and curricula that will be refined into standardized benchmarking experiments.

Configuration of L2Explorer

Fig. 4 shows some aspects of the world that can be reconfigured. These relate (indirectly) to the underlying POMDP, including O, ϕ, and S, by altering the object properties, object models, environmental complexity, environmental conditions, and observation modality. The reward R and state transition function T are more directly manipulated through modifying the object interaction model, object reward, and agent models. The extensible L2ExplorerTask class allows for further manipulations such as soft reward functions and observation state vectors.

Tasks

For each task a JSON is specified from which variants can be created and randomization can be done. Each task has a set of parameters which define the core task, parameters to change to produce static variants, and parameters that can be randomized within a variant. Fig. 5 shows an example of performance curves for a Stable Baselines Proximal Policy Optimization[1]. Developers of novel algorithms will be able to exploit this interface to baseline directly against single task experts. While L2Explorer is designed to be extensible, several initial tasks have been developed:

- Find Objects: The agent interacts with objects to receive a positive reward. Environmental conditions, object properties, and agent properties can be randomized. Variant tasks include observation changes of the target model, state space changes of the map layout, and agent-interaction model changes

[1]`https://github.com/DLR-RM/stable-baselines3`

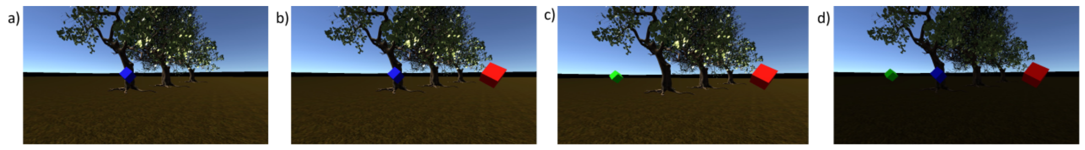

Figure 6: Example curricula showing variations in observation and reward models to realize the evaluation framework (Fig. 1). Between each variant, the reward structure and observation structure change, enabling analysis of performance and manipulating the underlying POMDP. In Panel a) there is a positive reward associated with capturing the blue object. In the variant in Panel b) a red object object is introduced which returns a negative reward when captured. In the variant in Panel c) a green object returns a positive reward. Finally, in Panel d) the variants in Panel b) and Panel c) are mixed with different environmental conditions.

- Get to Goal: The agent receives positive reward for navigating to a target position. Goal positions, map layouts, and agent parameters can be randomized. Variants include changes in visual landmarks, soft reward functions, and the addition of negative reward objects
- Select Object: The agent receives positive reward for selecting the correct object, negative reward for other objects. Reward probabilities can be manipulated as in a K-armed bandit problem. Variants include changes in the reward probability distribution and object models
- Moving Object: Similar to Find Objects, with a object motion model. Variants allow for manipulation of the action space and state transition space through changes in the agent and object motion models
- Scavenger Hunt: The agent can obtain a series of rewards by interacting with the right sequence of objects. Variants include changes in visual landmarks, changes in the reward distribution, and changes in the object models

Assessment Curricula

Tasks and task variants can be assembled into learning and evaluation blocks such as in Fig. 6. In this example changes in tasks can, through assumed heuristics such as a norm of the object RGB color difference, approximate changes in the observation space and reward space. The magnitude of parameter changes can also be used as a calculable surrogate for similarity, although we reemphasize that such heuristics do not necessarily indicate successful transfer learning.

Preliminary curricula are being developed for several key sources of variability impacting Continual RL agents. First is manipulation of the reward structure. Distributions of rewards and their associated objects can be shifted in gradual or continuous ways. Changes in the state space and state transition space can be probed through manipulations of the maps, agent and object motion models, and interaction models. Changes to the observation emission function can be probed through changes in environmental factors, observation modalities, and object models. Action space changes can relate to the agent interaction model. The metrics will allow the assessment of changes of different magnitudes.

Similarity measures are critical to the construction of these curricula. We propose utilizing changes in the parameters of the reset JSON to provide a usable, if limited, approximation of similarity. Changes to individual object parameters between tasks can be measured with chosen distance metrics to approximate differences in similarity between the tasks. Further, we may provide a mapping from non-numeric values, such as a color specification of “green”, to numeric ones, such as the array `(0, 128, 0)`, which can then be used with a distance measure to create another measure of similarity between tasks. Object coordinates can also be discretized into occupancy maps to allow for distance computations. While this approach is clearly limited, it is simple and useful for organizing curricula for algorithm assessment.

Conclusions

We have presented the design approach and implementation of L2Explorer, a highly configurable, Unity-based environment for testing continual and lifelong learning systems. In addition to the software system itself, we have argued for a framework to approach lifelong learning assessment in open-ended worlds. A key consideration is the integration of appropriate metrics, experiment design, and curriculum design into the framework.

To fully demonstrate the power of this approach, integration of single task baselines and Continual RL baselines (Khetarpal et al. 2020), including replay based approaches such as CLEAR (Rolnick et al. 2018), is needed. In combination with baseline development, the project aims to provide a set of canonical experiments for RL assessment in a FPV environment. Moreover, we anticipate the design framework can be reused. Without progress in assessment frameworks, it will be a challenge to push the frontier of reinforcement learning in open-world environments.

Code Availability

The code for the L2Explorer framework will be made open source at `https://github.com/lifelong-learning-systems/l2explorer`.

Acknowledgments

Development of L2Explorer was funded by the DARPA Lifelong Learning Machines (L2M) Program. The authors would like to thank Olivia Lyons, Ji Pak, Sasha Smith, Brianna Raphino, and Sydney Floryanzia for their help in the early development of L2Explorer, and Angel Yanguas-Gil for valuable feedback. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; Van Hasselt, H.; and Silver, D. 2016. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym.

Carroll, J. L. 2005. Task localization, similarity, and transfer; towards a reinforcement learning task library system.

Chan, S. C.; Fishman, S.; Canny, J.; Korattikara, A.; and Guadarrama, S. 2019. Measuring the reliability of reinforcement learning algorithms. *arXiv preprint arXiv:1912.05663*.

Chen, Z., and Liu, B. 2018. Lifelong machine learning, second edition. 12(3):1–207.

Dannenhauer, D.; Floyd, M. W.; Decker, J.; and Aha, D. W. 2019. Dungeon crawl stone soup as an evaluation domain for artificial intelligence. *arXiv preprint arXiv:1902.01769*.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 1–16.

Farquhar, S., and Gal, Y. 2018. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3(4):128–135.

Hsu, Y.-C.; Liu, Y.-C.; Ramasamy, A.; and Kira, Z. 2018. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*.

Juliani, A.; Berges, V.-P.; Teng, E.; Cohen, A.; Harper, J.; Elion, C.; Goy, C.; Gao, Y.; Henry, H.; Mattar, M.; et al. 2018. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.

Juliani, A.; Khalifa, A.; Berges, V.-P.; Harper, J.; Teng, E.; Henry, H.; Crespi, A.; Togelius, J.; and Lange, D. 2019. Obstacle tower: A generalization challenge in vision, control, and planning. *arXiv preprint arXiv:1902.01378*.

Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; and Jaśkowski, W. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. IEEE.

Khetarpal, K.; Riemer, M.; Rish, I.; and Precup, D. 2020. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*.

Kirk, R.; Zhang, A.; Grefenstette, E.; and Rocktäschel, T. 2021. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*.

Kolve, E.; Mottaghi, R.; Han, W.; VanderBilt, E.; Weihs, L.; Herrasti, A.; Gordon, D.; Zhu, Y.; Gupta, A.; and Farhadi, A. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Ma, C.; Ashley, D. R.; Wen, J.; and Bengio, Y. 2020. Universal successor features for transfer reinforcement learning. *arXiv preprint arXiv:2001.04025*.

McCloskey, M., and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*. Academic Press. 109–165.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

New, A.; Baker, M.; Nguyen, E.; and Vallabha, G. 2022. Lifelong learning metrics. *arXiv preprint arXiv:2201.08278*.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113:54–71.

Powers, S.; Xing, E.; Kolve, E.; Mottaghi, R.; and Gupta, A. 2021. Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents. *arXiv preprint arXiv:2110.10067*.

Rajeswaran, A.; Kumar, V.; Gupta, A.; Vezzani, G.; Schulman, J.; Todorov, E.; and Levine, S. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

Risi, S., and Togelius, J. 2020. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence* 2(8):428–436.

Rolnick, D.; Ahuja, A.; Schwarz, J.; Lillicrap, T. P.; and Wayne, G. 2018. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*.

Samvelyan, M.; Kirk, R.; Kurin, V.; Parker-Holder, J.; Jiang, M.; Hambro, E.; Petroni, F.; Kuttler, H.; Grefenstette, E.; and Rocktäschel, T. 2021. Minihack the planet: A sandbox for open-ended reinforcement learning research.

Shah, S.; Dey, D.; Lovett, C.; and Kapoor, A. 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, 621–635. Springer.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* 362(6419):1140–1144.

Song, L.; Sehwag, V.; Bhagoji, A. N.; and Mittal, P. 2020. A critical evaluation of open-world machine learning. *arXiv preprint arXiv:2007.04391*.

Staley, E. W.; Ashcraft, C.; Stoler, B.; Markowitz, J.; Vallabha, G.; Ratto, C.; and Katyal, K. 2021. Meta arcade: A configurable environment suite for deep reinforcement learning and meta-learning. In *Deep RL Workshop NeurIPS 2021*.

Team, O. E. L.; Stooke, A.; Mahajan, A.; Barros, C.; Deck, C.; Bauer, J.; Sygnowski, J.; Trebacz, M.; Jaderberg, M.; Mathieu, M.; et al. 2021. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* 575(7782):350–354.

Zhang, C.; Vinyals, O.; Munos, R.; and Bengio, S. 2018. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*.