Godot Reinforcement Learning Agents

Edward Beeching,[1] **Jilles Debangoye,** [1] **Olivier Simonin** [1] **Christian Wolf**[2]

[1]INRIA Chroma team, CITI Laboratory. INSA-Lyon, France.
[2]Université de Lyon, INSA-Lyon, LIRIS, CNRS, France.
Project page https://edbeeching.github.io/papers/gdrl

Abstract

We present Godot Reinforcement Learning (RL) Agents, an open-source interface for developing environments and agents in the Godot Game Engine. The Godot RL Agents interface allows the design, creation and learning of agent behaviors in challenging 2D and 3D environments with various on-policy and off-policy Deep RL algorithms. We provide a standard Gym interface, with wrappers for learning in the Ray RLlib and Stable Baselines RL frameworks. This allows users access to over 20 state of the art on-policy, off-policy and multi-agent RL algorithms. The framework is a versatile tool that allows researchers and game designers the ability to create environments with discrete, continuous and mixed action spaces. The interface is relatively performant, with 12k interactions per second on a high end laptop computer, when parallized on 4 CPU cores. An overview video is available here: https://youtu.be/g1MlZSFqIj4.

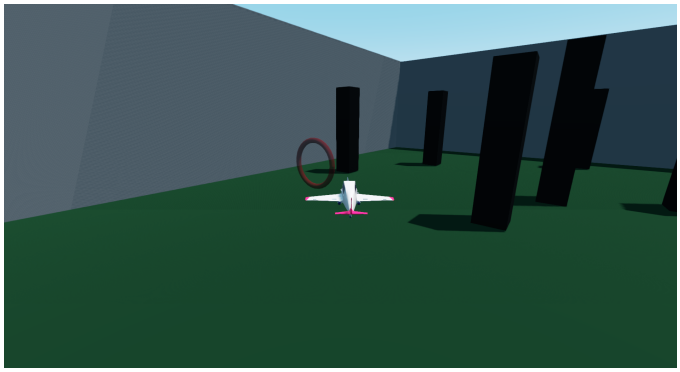


Figure 1: An example Godot RL Agents environment *Fly By*, where an agent learns to perform way-point based aerial navigation in a 3D space.

Introduction

Over the next decade advances in AI algorithms, notably in the fields of Deep Learning (Lecun, Bengio, and Hinton 2015) and Deep Reinforcement Learning, are primed to revolutionize the Video Games industry. Customizable enemies, worlds and story telling will lead to diverse gameplay experiences and new genres of games. Currently the field is dominated by large organizations and pay to use engines that have the resources to create such AI enhanced agents. The objective of the Godot RL Agents package is to lower the bar of accessibility; so that game developers, researchers and hobbyists can take their idea from creation to publication end-to-end with an open-source and free package. The Godot RL Agent provides wrappers for two well known open-source Deep Reinforcement Learning libraries: Ray Rllib (Liang et al. 2018) and Stable Baselines (Hill et al. 2018). In addition the interface is compatible with any RL framework that can interact with a gym wrapper. Although the Godot RL Agents interface was predominantly designed for the creation and prototyping of RL approaches in video games, there is nothing prohibiting users from the implementation of tasks involving mobile robotics, grasping or designing complex memory-based scenarios to test the limitations of current Deep RL architectures and algorithms. The contributions the Godot RL Agents packages are as follows:

- A free and open source tool for Deep RL research and game development.
- By providing an interface between the Godot Game Engine and Deep RL algorithms, we enable game creators to imbue their non-player characters with unique behaviors, learned through interaction.
- The framework enables automated game-play testing with an RL agent.
- The Godot RL Agents framework enables researchers and game designers the flexibility to design, create and perform rapid iteration on new ideas and scenarios.

The library is open-source with an MIT licence, code is available on our github page:
https://github.com/edbeeching/godot_rl_agents

Related work

The last decade has seen rapid advancement in the field of artificial intelligence, this has been driven by the application of Deep Learning approaches and in particular Convolutional Neural Networks (Fukushima and Miyake 1982; Lecun et al. 1998) . Supervised Deep Learning in its current state often considered to be narrow AI, that performs particularly well at performing predictions on static images but does not **learn from interaction**.

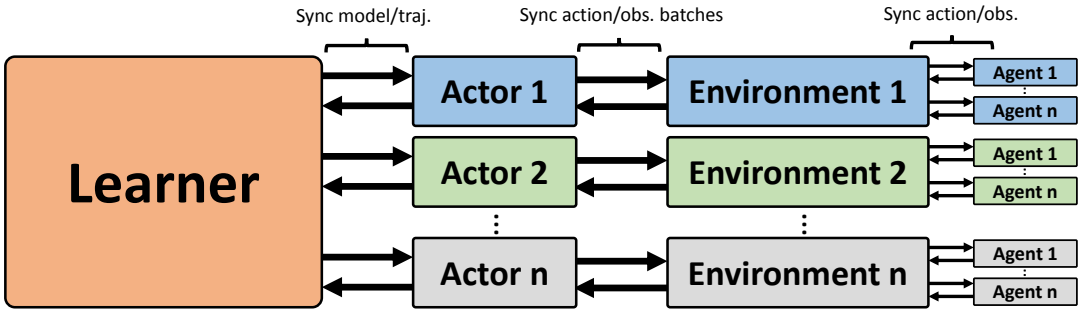


Figure 2: The training architecture used for the Godot RL Agents interface. Parallel actor processes collect trajectories from several environment executables. Each environment contains multiple parallel agents, each with their own instantiation of the environment.

Deep Reinforcement Learning

Reinforcement Learning approaches provide the ability to learn in sequential decision making problems, where the objective is to maximize accumulated reward. As we encounter large state spaces, continuous actions, partial observability (Åström 1965) and the desire to generalize to unseen environment configurations; Deep Reinforcement Learning (RL) provides a general framework for solving these problems. In recent years the field of Deep RL has gained attention with successes on board games (Silver et al. 2016) and Atari Games (Mnih et al. 2015). One key component was the application of deep neural networks to frames from the environment or game board states. Recent works that have applied Deep RL for the control of an agent in 3D environments such as maze navigation are (Mirowski et al. 2017) and (Jaderberg et al. 2017) which explored the use of auxiliary tasks such as depth prediction, loop detection and reward prediction to accelerate learning. Meta RL approaches for 3D navigation have been applied by (Wang et al. 2016) and (Lample and Chaplot 2017) also accelerated the learning process in 3D environments by prediction of tailored game features. There has also been recent work in the use of street-view scenes in order train an agent to navigate in city environments (Kayalibay et al. 2018). In order to infer long term dependencies and store pertinent information about the environment; network architectures typically incorporate recurrent memory such as Gated Recurrent Units (Chung et al. 2015) or Long Short-Term Memory (Hochreiter and Schmidhuber 1997).

The scaling of Deep RL has produced some impressive results, such as in the IMPALA (Espeholt et al. 2018) architecture which successfully trained an agent that can achieve human level performance in all 30 of the 3D, partially observable, DeepMind Lab (Beattie et al. 2016) tasks; accelerated methods such as in (Stooke and Abbeel 2018) which solve many Atari environments in tens of minutes. The achievements in long term planing and strategy in DOTA (OpenAI 2018) and StartCraft 2 (Vinyals et al. 2019), has demonstrated the Deep RL can learn policies with complex reasoning and long term decision making. There has been a recent push towards photo-realistic simulation, with the release of the Gibson (Xia et al. 2018) dataset and the Habitat simulator (Manolis Savva et al. 2019; Szot et al. 2021). Many research teams focus on Structured Deep Reinforcement Learning, where priors are incorporated in the agent architecture, leading to advancements in GPS guided point-to-point navigation (Chaplot et al. 2020a,d), memory-based tasks (Wayne et al. 2018; Beeching et al. 2020a,b), semantic prediction (Chaplot et al. 2020b,c). Other approaches aim to solve these challenging problems with scale, using distributed computing to training larger networks for billions of environment interactions (Wijmans et al. 2019; Espeholt et al. 2018). The video games industry has identified the advancements in Deep Reinforcement Learning as an opportunity to enrich video game experiences. With short term objectives being that of automated testing (Gordillo et al. 2021) and content generation (Gisslén et al. 2021), and medium to longer term being player facing bots (Alonso et al. 2021), controlled by policies learned through interaction with the game environment.

Deep RL environments

There has been a large expansion to the number of Deep RL environments available in the last five years, the most well known being the gym framework (Brockman et al. 2016) which provides an interface to fully observable games such as the Atari-57 benchmark. Mujuco (Todorov, Erez, and Tassa 2012) has remained the de facto standard for continuous control tasks, its recent open-source release can only benefit the larger RL community. OpenSpiel (Lanctot et al. 2019) provides a framework for RL in games such as Chess, Poker, Go, TicTacToe and Habani, to name a few. There are a variety of partially observable 3D simulators available, with more game-like environments such as the ViZDoom simulator (Wydmuch, Kempka, and Jaśkowski 2018), DeepMind-Lab (Beattie et al. 2016) and Malmo (Johnson et al. 2016). A number of photo-realistic simulators have also been released such as Gibson (Xia et al. 2018), AI2Thor (Kolve et al. 2017), and the highly efficient Habitat-Lab simulator (Manolis Savva et al. 2019; Szot et al. 2021). The aforementioned environments were designed to train and evaluate RL agents in relatively small environments and typically are somewhat inflexible to addition of new tasks, abilities and action spaces. The Unity ML agents framework (Juliani et al. 2018), provides flexibility in terms of design of envi-

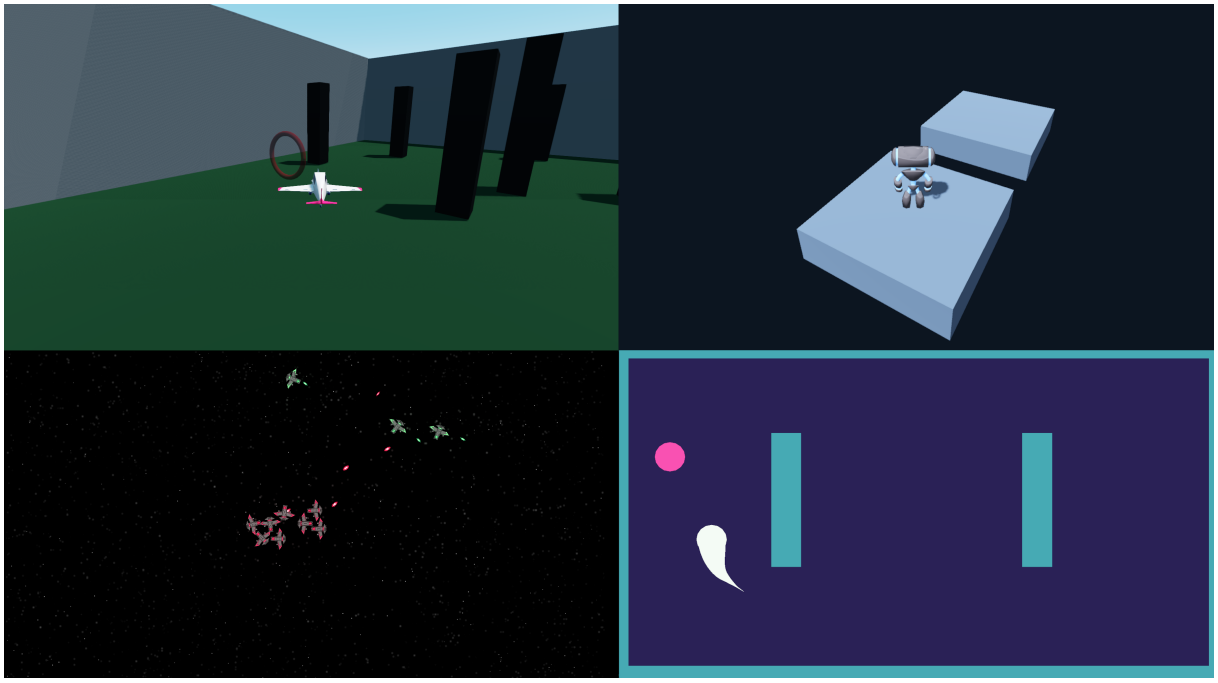


Figure 3: 4 of the example environments available in the Godot RL Agents framework. Top left: *Fly By*, where the agent must learn to fly a place between way-points in 3D space. Top right: *Jumper*, where the objective is the navigated a bipedal robot between platforms, jumping where necessary. Bottom left: *Space Shooter*, a 2D multi agent scenario, where one team must shoot and and destroy the agents from the other team. Bottom-right: *Ball Chase*, where the agent must learn to collect pink fruits, while avoiding walls and obstacles.

ronments and tasks, but with the downside of being a closed-source and pay to use (if your organization has a revenue of more than $100k per year). In addition its closed-source implementation is inflexible if you wish to, for example, make changes to the protocol used to exchange observations between the environment and the RL algorithm. For these reasons the platform is platform somewhat prohibitive to the wider RL community.

The Godot Game Engine

The Godot Game Engine (Linietsky, Manzur, and Verschelde 2014) is an open-source tool for developing 2D and 3D games. Currently Godot is the most popular open-source Game Engine available and can export to MacOS, Linux, Windows, Android, iOS, HTML and web assembly. Godot supports multiple programming languages including Python, C#, C++ and GDScript (a Python-like scripting language). The Game Engine provides an interactive editor for the design and creation of video game environments, characters, animations and menus, which enables fast iteration for prototyping new ideas. Godot has all the features of a modern Game Engine such physics simulation, custom animations, plugins and physically-based rendering.

Deep RL Frameworks

When it comes to applying a state of the art Deep RL algorithm, there are many quality open-source implementations available such as RLlib (Liang et al. 2018), ACME (Hoffman et al. 2020), SampleFactory (Petrenko et al. 2020), Seed RL (Espeholt et al. 2019) and Stable Baselines (Hill et al. 2018) to name a few. Due to the availability of such high-quality, featured and well tested Deep RL frameworks; in this work we chose to design and implement an interface between the Godot Game Engine and two well known RL frameworks: RLlib (Liang et al. 2018) and Stable Baselines (Hill et al. 2018). Our objective is to focus on the interface and quality examples of 2D and 3D environments. This is contrary to the Unity ML Agents framework, which also implemented a limited set of RL algorithms.

Godot RL Agents

The Godot RL Agents framework is an interface between the Godot Game Engine and Deep RL frameworks, allowing the creation of 2D, 3D and text-based environments, custom observations and reward functions. It is a flexible, fast and robust interface between RL training code running in Python and the Godot environment running interactively or as a compiled executable. We implement two forms of parallelization in the Godot RL Agents interface: in-game environment duplication and multiprocessing with several parallel executables. The interface has been designed to support continuous, discrete and mixed action spaces. Action repetition is implemented on the environment side to avoid unnecessary inter-process communication. Training can be performed interactively, while the environment is running

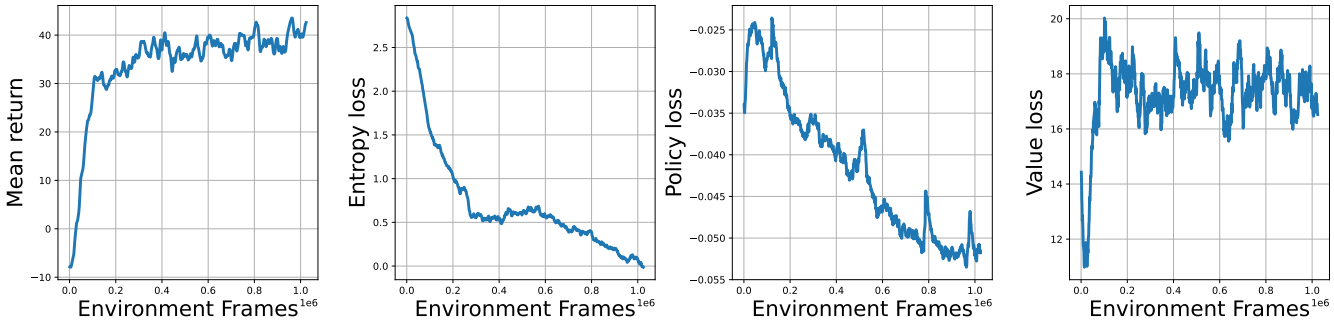


Figure 4: Training curves available in Godot RL Agents for the Ball Chase environment. Shown are a subset of available statistics output during training, Mean return, Entropy loss, Policy loss and Value loss

in the Godot game editor: enabling mid-training pausing, analysis and debugging of variables. Once the user is confident with their environment configuration though interactive testing, the environment can be exported and run as an executable. Environment export enables accelerated, faster than real-time physics and headless training. The connection between the Game Engine and the Python training code is implemented with a TCP socket client-server relationship; this allows the environment to be run locally or distributed across many machines.

We provide a standard Gym wrapper and wrappers to Ray RLlib and Stable Baselines, which support the following algorithms.

On-policy algorithms:

- A2C/A3C (Mnih et al. 2016)
- PPO/APPO (Schulman et al. 2017)
- IMPALA (Espeholt et al. 2018)
- DD-PPO (Wijmans et al. 2019)

Off-policy algorithms:

- DQN (Mnih et al. 2013)
- Rainbow (Hessel et al. 2018)
- CQL (Kumar et al. 2020)
- DDPG / TD3 (Lillicrap et al. 2015)
- APEX (Horgan et al. 2018)
- R2D2 (Kapturowski et al. 2018)
- SAC continuous (Haarnoja et al. 2018) and discrete (Christodoulou 2019).

Other algorithms are also available such as Behavior Cloning (Wang et al. 2018), Intrinsic curiosity (Pathak et al. 2017) and Evolutionary Strategies (Salimans et al. 2017). These pre-existing implementations enable researchers and game developers to rapidly evaluate different baselines architectures and algorithms, with robust implementations of state of the art RL algorithms. The detail the distributed architecture used in Godot RL Agents in Figure 2.

Sensors

Godot RL Agents allows users to add custom sensors to their agents to observe the environment. While simulated monocular cameras are possible, rendering is an expensive operation and rendered images do no always provide the more pertinent information. We provide implementations of circular and spherical "Raycast" sensors in both 2D and 3D that perform depth measurements, essentially a virtual LIDAR. Raycasts are a popular technique in video games and Game Engines such as Godot offer a highly optimized raycast implementation, making ray-casts a cheap option to augment an agent's observation. Other custom information, such as position, can be included in the agents observation, if the environment designer believes it is applicable to the task.

Example environments

In version 0.1.0 of Godot RL Agents, we have implemented 4 example environments as a reference for researchers and game creators who wish to use our tool to build new worlds. We have created two 3D environments and two 2D environments. The source-code and further details (such as reward functions, reset conditions, action spaces) of these reference implementations, is available in our open-source repository https://github.com/edbeeching/godot_rl_agents.

Jumper

Jumper is a 3D single agent environment where the agent must learn to jump from one randomly placed platform to the next. The action space is continuous for turning and movement, and discrete for the jump action. The episode is terminate if the agent falls off the platform. Observations are a cone of 3D ray-casts in front of the agent and vector pointing towards the next platform.

Ball Chase

Ball Chase is a 2D single agent environment where the agent must navigate around room and collect pink balls. The episode is terminated if the agent hits a wall. Observations are a circle of 2D ray-casts around the agent and vector pointing towards the next fruit. The action space is a continuous 2D vector indicating the move direction in x and y.

Fly By

Fly By is a 3D single environment where the agent must learn to fly from one way-point to the next. The episode is terminated if the agent hits a leave the game are. Observations are two vectors to the next two subsequent way-points.

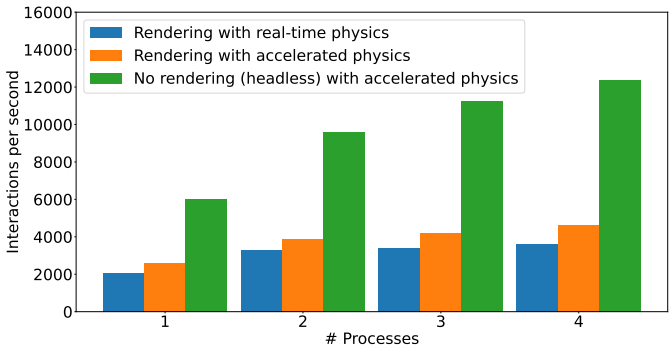


Figure 5: Benchmark of interactions per second when performing multi-process training in the Jumper environment with an action repeat of 4. We achieve 12k interactions per second with 4 parallel processes. Ideally scaling would be linear, future versions of Godot RL Agents will aim to profile and optimize the scaling of environment interactions.



Figure 6: Future work, an example of a virtual monocular camera, rendering part of a scene from the Gibson dataset, loaded as an environment in the Godot RL Agents framework.

Space Shooter

Space Shooter is a 2D multi-agent environment with two teams of 8 agents, where the objective is to shoot and destroy the enemy team. Currently observations are an array of relative positions of the enemy and friend teams. Results, observations and training methodology in the environment are still work in progress and will change in the next version of Godot RL Agents.

Experiments and Benchmarks

As part of our reference implementation, we provide a PPO example with reasonable default parameters in order to provide a good starting point for training RL agents. We believe this algorithm is a suitable starting point, as it allows for both continuous and discrete action spaces, is robust to hyper-parameter configurations, and is relatively sample efficient. In Figure 4 we show example training curves for the Ball Chase environment. We provide pre-trained models for all example environments in our repository, refer to our overview video for examples of these behaviors: https://youtu.be/g1MlZSFqIj4.

We benchmark the interaction rate of the Jumper environment and achieve 12k environment interactions per second with up to 4 parallel processes, results shown in Figure 5. Benchmarking was performed on a relatively high-end laptop computer [1], future work on Godot RL Agents will benchmark on dedicated hardware.

Conclusions and future work

We have introduced the Godot RL Agents framework, a tool for building 2D and 3D environments in the Godot Game Engine and learning agent behaviors with Deep Reinforcement Learning. We have created 4 example environments to demonstrate the capabilities of this versatile tool. By interfacing with well known open-source Deep RL implementations, we provide access to over 20 RL algorithms out of the box. We have implemented sensors, tools for parallel interaction with environments, and support for continuous, discrete and mixed action spaces.

Future work on the framework aims to include: variable size observations coupled with attention mechanisms, text-based environments, environments with discrete state spaces, virtual monocular cameras for vision-based agent control, multi-modal observations (1D, 2D, text & audio) and model export for in-engine inference of trained behaviors. We plan to extend our focus to multi-agent environments, preliminary work in the *Space Shooter* environment shows this area is a challenging area of research, with applications in both video games and collaborative robotics.

External datasets can also be loaded in to the Godot Game Engine. In Figure 6 we load photo-realistic scans from the Gibson (Xia et al. 2018) dataset, to demonstrate that is tool can be used to build realistic 3D scenes for learning robotic control and sim2real transfer. We believe that there are near endless possibilities for this tool and are interesting to discuss with the research community the future direction of this work.

[1] Dell XPS - i7-10750H (6 cores), GTX 1650 Ti, 32 GB RAM

References

Alonso, E.; Peter, M.; Goumard, D.; and Romoff, J. 2021. Deep Reinforcement Learning for Navigation in AAA Video Games. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2133–2139. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Åström, K. J. 1965. Optimal control of Markov processes with incomplete state information I. *Journal of Mathematical Analysis and Applications*, 10: 174–205.

Beattie, C.; Leibo, J. Z.; Teplyashin, D.; Ward, T.; Wainwright, M.; Küttler, H.; Lefrancq, A.; Green, S.; Valdés, V.; Sadik, A.; Schrittwieser, J.; Anderson, K.; York, S.; Cant, M.; Cain, A.; Bolton, A.; Gaffney, S.; King, H.; Hassabis, D.; Legg, S.; and Petersen, S. 2016. DeepMind Lab. *arxiv pre-print 1612.03801*.

Beeching, E.; Dibangoye, J.; Simonin, O.; and Wolf, C. 2020a. EgoMap: Projective mapping and structured egocentric memory for Deep RL. In *ECMLPKDD*.

Beeching, E.; Dibangoye, J.; Simonin, O.; and Wolf, C. 2020b. Learning to plan with uncertain topological maps. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 473–490. Springer.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *arxiv pre-print 1606.01540*.

Chaplot, D. S.; Gandhi, D.; Gupta, S.; Gupta, A.; and Salakhutdinov, R. 2020a. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*.

Chaplot, D. S.; Gandhi, D. P.; Gupta, A.; and Salakhutdinov, R. R. 2020b. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33.

Chaplot, D. S.; Jiang, H.; Gupta, S.; and Gupta, A. 2020c. Semantic curiosity for active visual learning. In *European Conference on Computer Vision*, 309–326. Springer.

Chaplot, D. S.; Salakhutdinov, R.; Gupta, A.; and Gupta, S. 2020d. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12875–12884.

Christodoulou, P. 2019. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2015. Gated Feedback Recurrent Neural Networks. In *ICML*.

Espeholt, L.; Marinier, R.; Stanczyk, P.; Wang, K.; and Michalski, M. 2019. SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference.

Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; et al. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Fukushima, K.; and Miyake, S. 1982. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, 267–285. Springer.

Gisslén, L.; Eakins, A.; Gordillo, C.; Bergdahl, J.; and Tollmar, K. 2021. Adversarial reinforcement learning for procedural content generation. *arXiv preprint arXiv:2103.04847*.

Gordillo, C.; Bergdahl, J.; Tollmar, K.; and Gisslén, L. 2021. Improving Playtesting Coverage via Curiosity Driven Reinforcement Learning Agents. arXiv:2103.13798.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

Hessel, M.; Modayil, J.; van Hasselt, H. P.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2018. Stable Baselines. https://github.com/hill-a/stable-baselines.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Hoffman, M.; Shahriari, B.; Aslanides, J.; Barth-Maron, G.; Behbahani, F.; Norman, T.; Abdolmaleki, A.; Cassirer, A.; Yang, F.; Baumli, K.; Henderson, S.; Novikov, A.; Colmenarejo, S. G.; Cabi, S.; Gulcehre, C.; Paine, T. L.; Cowie, A.; Wang, Z.; Piot, B.; and de Freitas, N. 2020. Acme: A Research Framework for Distributed Reinforcement Learning. *arXiv preprint arXiv:2006.00979*.

Horgan, D.; Quan, J.; Budden, D.; Barth-Maron, G.; Hessel, M.; Van Hasselt, H.; and Silver, D. 2018. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2017. Reinforcement Learning with Unsupervised Auxiliary Tasks. In *ICLR*.

Johnson, M.; Hofmann, K.; Hutton, T.; and Microsoft, D. B. 2016. The Malmo Platform for Artificial Intelligence Experimentation. In *IJCAI*.

Juliani, A.; Berges, V.-P.; Vckay, E.; Gao, Y.; Henry, H.; Mattar, M.; and Lange, D. 2018. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.

Kapturowski, S.; Ostrovski, G.; Quan, J.; Munos, R.; and Dabney, W. 2018. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*.

Kayalibay, B.; Mirchev, A.; Soelch, M.; Van Der Smagt, P.; and Bayer, J. 2018. Navigation and planning in latent maps. In *FAIM workshop "Prediction and Generative Modeling in Reinforcement Learning"*.

Kolve, E.; Mottaghi, R.; Gordon, D.; Zhu, Y.; Gupta, A.; and Farhadi, A. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. *arxiv pre-print 1712.05474v1*.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*.

Lample, G.; and Chaplot, D. S. 2017. Playing FPS Games with Deep Reinforcement Learning. In *AAAI*.

Lanctot, M.; Lockhart, E.; Lespiau, J.-B.; Zambaldi, V.; Upadhyay, S.; Pérolat, J.; Srinivasan, S.; Timbers, F.; Tuyls, K.; Omidshafiei, S.; Hennes, D.; Morrill, D.; Muller, P.; Ewalds, T.; Faulkner, R.; Kramár, J.; Vylder, B. D.; Saeta, B.; Bradbury, J.; Ding, D.; Borgeaud, S.; Lai, M.; Schrittwieser, J.; Anthony, T.; Hughes, E.; Danihelka, I.; and Ryan-Davis, J. 2019. OpenSpiel: A Framework for Reinforcement Learning in Games. *CoRR*, abs/1908.09453.

Lecun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature*, 521(7553): 436–444.

Lecun, Y.; Eon Bottou, L.; Bengio, Y.; and Haaner, P. 1998. Gradient-Based Learning Applied to Document Recognition. *IEEE*, 86(11): 2278–2324.

Liang, E.; Liaw, R.; Nishihara, R.; Moritz, P.; Fox, R.; Goldberg, K.; Gonzalez, J.; Jordan, M.; and Stoica, I. 2018. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, 3053–3062. PMLR.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Linietsky, J.; Manzur, A.; and Verschelde, R. 2014. Godot. https://github.com/godotengine/godot.

Manolis Savva; Abhishek Kadian; Oleksandr Maksymets; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; Parikh, D.; and Batra, D. 2019. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A. J.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; Kumaran, D.; and Hadsell, R. 2017. Learning to Navigate in Complex Environments. In *ICLR*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the Thirty-Third International Conference on Machine Learning*, 1928–1937. PMLR.

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*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*.

OpenAI. 2018. OpenAI Five. https://blog.openai.com/openai-five/.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, 2778–2787. PMLR.

Petrenko, A.; Huang, Z.; Kumar, T.; Sukhatme, G.; and Koltun, V. 2020. Sample Factory: Egocentric 3D Control from Pixels at 100000 FPS with Asynchronous Reinforcement Learning. In *ICML*.

Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; and Sutskever, I. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587): 484–489.

Stooke, A.; and Abbeel, P. 2018. Accelerated Methods for Deep Reinforcement Learning. *arxiv pre-print 1803.02811*.

Szot, A.; Clegg, A.; Undersander, E.; Wijmans, E.; Zhao, Y.; Turner, J.; Maestre, N.; Mukadam, M.; Chaplot, D.; Maksymets, O.; Gokaslan, A.; Vondrus, V.; Dharur, S.; Meier, F.; Galuba, W.; Chang, A.; Kira, Z.; Koltun, V.; Malik, J.; Savva, M.; and Batra, D. 2021. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. *arXiv preprint arXiv:2106.14405*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 5026–5033. IEEE.

Vinyals, O.; Babuschkin, I.; Czarnecki, W.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J.; Jaderberg, M.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, J. X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J. Z.; Munos, R.; Blundell, C.; Kumaran, D.; and Botvinick, M. 2016. Learning to reinforcement learn. *arxiv pre-print 1611.05763*.

Wang, Q.; Xiong, J.; Han, L.; sun, p.; Liu, H.; and Zhang, T. 2018. Exponentially Weighted Imitation Learning for Batched Historical Data. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Wayne, G.; Hung, C.-C.; Amos, D.; Mirza, M.; Ahuja, A.; Grabska-Barwinska, A.; Rae, J.; Mirowski, P.; Leibo, J. Z.; Santoro, A.; et al. 2018. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*.

Wijmans, E.; Kadian, A.; Morcos, A.; Lee, S.; Essa, I.; Parikh, D.; Savva, M.; and Batra, D. 2019. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*.

Wydmuch, M.; Kempka, M.; and Jaśkowski, W. 2018. ViZDoom Competitions: Playing Doom from Pixels. *arXiv preprint arXiv:1809.03470*.

Xia, F.; R. Zamir, A.; He, Z.-Y.; Sax, A.; Malik, J.; and Savarese, S. 2018. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE.