# Retrospective Analysis of the 2019 MineRL Competition on Sample Efficient Reinforcement Learning

**Stephanie Milani**                                                          SMILANI@CS.CMU.EDU
*Machine Learning Department, Carnegie Mellon University*

**Nicholay Topin**                                                            NTOPIN@CS.CMU.EDU
*Machine Learning Department, Carnegie Mellon University*

**Brandon Houghton**[*]                                                       BHOUGHTON@OPENAI.COM
*OpenAI*

**William H. Guss**                                                           WGUSS@CS.CMU.EDU
*OpenAI and Machine Learning Department, Carnegie Mellon University*

**Sharada P. Mohanty**                                                        MOHANTY@AICROWD.COM
*AIcrowd*

**Keisuke Nakata**                                                            NAKATA@PREFERRED.JP
*Preferred Networks*

**Oriol Vinyals**                                                             VINYALS@GOOGLE.COM
*DeepMind*

**Noboru Sean Kuno**                                                          NKUNO@MICROSOFT.COM
*Microsoft Research*

**Editors:** Hugo Jair Escalante and Raia Hadsell

## Abstract

To facilitate research in the direction of sample efficient reinforcement learning, we held the MineRL Competition on Sample Efficient Reinforcement Learning Using Human Priors at the Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019). The primary goal of this competition was to promote the development of algorithms that use human demonstrations alongside reinforcement learning to reduce the number of samples needed to solve complex, hierarchical, and sparse environments. We describe the competition, outlining the primary challenge, the competition design, and the resources that we provided to the participants. We provide an overview of the top solutions, each of which use deep reinforcement learning and/or imitation learning. We also discuss the impact of our organizational decisions on the competition and future directions for improvement.

**Keywords:** reinforcement learning competition, reinforcement learning, imitation learning

---

## 1. Introduction

Many of the recent, celebrated successes of artificial intelligence (AI), such as AlphaStar (Vinyals et al., 2019), AlphaZero (Silver et al., 2018), and OpenAI Five[1] achieve human- or super-human-level performance using deep reinforcement learning (DRL). Thus far, improvements to the state of the art have used ever-increasing computational power[2]. In part, this computational increase is due to the computation required per environment sample; however, it is primarily due to the increasing number of environment samples required for training these learning algorithms. These growing computational requirements mean that a shrinking portion of the AI community can reproduce these results, let alone improve upon these systems. One well-known way to reduce environment-sample complexity, and, by extension, make methods more accessible, is to leverage human demonstrations and priors over desired behavior (Pfeiffer et al., 2018; Dubey et al., 2018; Zhang et al., 2019).

To further the development of novel, sample efficient methods that leverage human priors for sequential decision-making problems, we held the first-ever MineRL Competition on Sample Efficient Reinforcement Learning Using Human Priors[3] at the Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019) (Guss et al., 2019). Teams developed procedures for training an agent to obtain a diamond in Minecraft. To limit computational requirements, the learning procedures were evaluated by retraining them from random weights under a strict computational and environment-sample budget. To assist participants with developing their algorithms, teams were provided with the largest-ever data set of human trajectories in Minecraft (Guss* et al., 2019). Our challenge attracted over 1000 registered participants with a total of 662 submissions[4].

In this work, we describe the competition paradigm and procedure, and provide a summary of the top nine solutions. We identify common high-level approaches — such as leveraging human demonstrations and using hierarchical reinforcement learning — used in many of the top solutions. We conclude by discussing the outcomes of our choices as organizers of the competition, including how we evaluated the submissions and specified the rules.

## 2. Background

**Minecraft.** Minecraft is a 3D, first-person, open-world game revolving around gathering resources and creating structures and items. Because it is a sandbox environment, designers can devise a variety of goals and challenges for intelligent agents. Additionally, because it is an embodied domain and the agent's surroundings are varied and dynamic, it presents many of the same challenges found in real-world control tasks, such as planning over long time horizons and determining a good representation of the environment (Alterovitz et al., 2016). As a result, it is a promising and popular testbed for both single- and multi-agent reinforcement learning (RL) and planning algorithms (Abel et al., 2015; Aluru et al., 2015; Abel et al., 2016; Oh et al., 2016; Tessler et al., 2017; Shu et al., 2018; Frazier and Riedl, 2019; Master et al., 2019; Arumugam et al., 2019; Trott et al., 2019).

---

1. https://openai.com/five

2. https://blog.openai.com/ai-and-compute

3. http://minerl.io

4. https://www.aicrowd.com/challenges/neurips-2019-minerl-competition

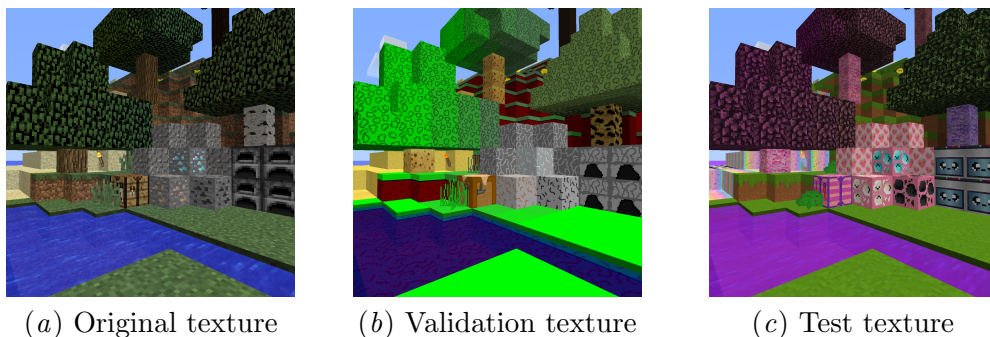(*a*) Original texture      (*b*) Validation texture      (*c*) Test texture

Figure 1: An example game state rendered with three different textures.

**Reinforcement Learning Competitions,** To our knowledge, no previous competitions explicitly focused on using imitation learning (and, more generally, learning from demonstrations) alongside RL. Most previous RL competitions (Kidziński et al., 2018; Nichol et al., 2018; Perez-Liebana et al., 2018; Juliani et al., 2019) focused on performing well on a given domain, and not on developing robust algorithms that can perform well on a broad set of domains. Consequently, winning submissions often required hand-engineered features and stemmed from using large amounts of computational resources for training and development.

## 3. Competition Overview

We provide an overview of our competition. We describe the primary task and environment in Section 3.1. In Section 3.2, we describe the structure of our competition. In Section 3.3, we describe the resources that we provided participants.

### 3.1. Task

Competitors were tasked with solving the `ObtainDiamond` environment. Solving the environment consists of controlling an embodied agent to obtain a diamond by navigating the complex item hierarchy of Minecraft. In solving this task, a learning algorithm has direct access to a 64x64 pixel observation from the perspective of the embodied Minecraft agent, and a set of discrete observations consisting of each item required for obtaining a diamond that the agent has in its possession. The action space is the Cartesian product of continuous view adjustment (turning and pitching), binary movement commands (left/right, forward/backward), and discrete actions for placing blocks, crafting items, smelting items, and mining/hitting enemies.

An agent receives reward for completing the full task of obtaining a diamond. The full task of obtaining a diamond can be decomposed into a sequence of prerequisite subtasks of increasing difficulty. An agent also receives reward for the first time it accomplishes each subtask in the sequence. An agent receives twice the reward as received for accomplishing the previous subtask (starting from a reward of 1). The exception to this rule is achieving the full `ObtainDiamond` task by obtaining a diamond: accomplishing the final task is worth four times as much as completing the previous subtask.

### 3.2. Competition Design

The competition was split into two rounds. In Round 1, teams of up to six total participants trained their agents and submitted trained models for evaluation to determine their leaderboard ranks. In this round, we limited each team to 25 total submissions. At the end of Round 1, the teams submitted their source code. Each submission received a score consisting of the average reward achieved by that submission over 100 episodes. The top submission for each team was then reviewed by the rules committee to verify that they complied with the rules. The 13 teams with the highest-scoring, rule-compliant submissions advanced to Round 2. We invited these teams to present their work at the NeurIPS 2019 competition workshop. With the support of Microsoft Research, we provided each team with a $1200 USD travel grant to attend the conference.

In Round 2, each team could submit up to five learning procedures to be trained from random weights on a held-out test environment. We independently trained each submission for six days using Azure NC6 and NC12 instances with a strict environment step budget of 8 million samples. To encourage generalizability and compliance with rules, we retrained the solutions submitted in the final round using: 1) a perturbed action space, and 2) an entirely new, previously-unseen texture pack[5] (illustrated in Figure 1(c)). We provided teams with a validation texture pack[6] (illustrated in Figure 1(b)) to use during training so that they could check whether their appeoaches could generalize to previously-unseen environments before submitting their learning procedures. The final ranking was determined based on each team's highest-scoring submission.

Given the unique paradigm of training learning procedures, we set strict rules to encourage sample efficiency and generalizability, and prevent participants from loading pretrained models. Sample efficiency constraints were enforced by setting a strict environment sample limit of 8 million frames (or roughly 110 hours of in-game interactions), as well as limiting computation to 144 hours of wall clock time. We permitted frame skipping; however, each frame skipped counted against the sample budget. Participants were also free to shape the observations of the agent, but they could not directly encode them in the policy. For example, switching sub-policies based on the number of logs collected was not permitted. To avoid participants including pretrained models in their submissions, in Round 2, any files in the submitted code repositories that exceeded 4MB were programmatically deleted. A more detailed discussion of the competition paradigm can be found in previous work (Houghton et al., 2019).

In both rounds, participants submitted their code as standalone repositories compatible with AIcrowd-repo2docker[7]. These repositories included the code, trained models, and runtime specifications, which enabled us to build consistent Docker images out of the submissions. These images were orchestrated on a custom Kubernetes cluster while respecting any round-specific policies. We evaluated the submissions in complete network isolation to avoid any leak of information due to malicious code submitted by participants.

---

5. The test texture pack we used can be found here.

6. We used a modified version of this texture pack as our validation texture pack.

7. https://github.com/AIcrowd/repo2docker

| Team Name | Round 1 Score | Team Name | Round 2 Score |
|-----------|---------------|-----------|---------------|
| MeatMountain | 48.42 | CDS | 61.61 |
| xt | 46.75 | mc_rl | 42.41 |
| shadowyzy | 37.82 | i4DS | 40.80 |
| CraftRL | 33.15 | CraftRL | 23.81 |
| CDS | 29.62 | UEFDRL | 17.90 |
| mc_rl | 27.43 | TD240 | 15.19 |
| I4DS | 23.96 | LAIR | 14.73 |
| UEFDRL | 21.70 | Elytra | 8.25 |
| TD240 | 17.12 | karolisram | 7.87 |

Table 1: Scores of best-performing submissions of top teams in Round 1 and Round 2.

### 3.3. Resources

In addition to the `ObtainDiamond` task, we created a number of auxiliary tasks, which we believed would be useful for solving `ObtainDiamond`. These tasks were provided to participants as Gym environments (Brockman et al., 2016). Additionally, we provided the participants with a comprehensive starter pack[8], consisting of extensive documentation, starter code, a description of the submission procedure, a Microsoft Azure quick-start template, and the Docker images that we used to validate the training procedures during retraining. We gave the participants a large dataset of human demonstrations (Guss* et al., 2019), which participants could use to train their agents. Preferred Networks[9] provided extensive baselines[10] implemented in ChainerRL (Fujita et al., 2019). Participants were able to incorporate these baselines into their submissions. These baselines included behavioral cloning, deep Q-learning from demonstrations (DQfD) (Hester et al., 2018), Rainbow (Hessel et al., 2018), generative adversarial inverse RL (GAIL) (Ho and Ermon, 2016), and proximal policy optimization (PPO) (Schulman et al., 2017).

AIcrowd[11] provided a unified interface[12] for participants to sign-up for the competition, ask the organizers and other participants questions, and monitor their progress. We also created a public Discord server for more informal communication with participants.

Through our generous sponsor, Microsoft Azure, we provided compute grants of $500USD each in Micrsoft Azure[13] credits for 50 individuals that self identified as lacking access to the necessary compute power to participate in the competition. Additionally, thanks to Microsoft Azure, we provided each of the top teams from Round 1 with $1500USD of Azure credits for their experiments in Round 2.

---

8. https://github.com/minerllabs/competition_submission_starter_template

9. https://preferred.jp/en

10. https://github.com/minerllabs/baselines

11. https://www.aicrowd.com

12. https://discourse.aicrowd.com/c/neurips-2019-minerl-competition

13. https://azure.microsoft.com/en-us

Figure 2: Maximum item score for each team over the 100 evaluation episodes in Round 2.

## 4. Outcomes for Participants

We quantitatively and qualitatively analyze the submissions made by participants of our competition. In Section 4.1, we provide quantitative information about the submissions. In Section 4.2, we summarize the approaches used by the top nine teams in the final round. In Section 4.3, we describe the special awards awarded as part of the competition.

### 4.1. Submission Performance Overview

In Round 1, there were over 540 submissions, and 25 teams drastically outperformed the provided baselines. Table 1 shows that the top 3 teams from Round 2 on average outperformed the top 3 teams from Round 1 (means: 48.27 and 44.33, respectively), despite the Round 2 agents being trained on a hold-out test environment. Perhaps unsurprisingly, the Round 1 scores had a higher mean (31.77) and lower standard deviation (10.22) than the Round 2 scores (25.84 and 17.37, respectively). Additionally, the Round 1 scores had a smaller range (31.30) than the Round 2 scores (53.74). Figure 2 shows the maximum item score over the evaluation episodes of Round 2. Although no team obtained a diamond, the top team obtained the penultimate prerequisite item to obtaining a diamond.

### 4.2. Summary of Top Nine Solutions

The top four teams used some form of hierarchical RL, taking advantage of the hierarchicality of Minecraft. All teams used some form of action reduction (e.g., removing actions that were almost never performed by human experts) to manage the complexity of the environment. Most teams leveraged the human data to improve the sample efficiency of their algorithms.

The top team, CDS, used a hierarchical deep Q-network with forgetting that uses an adaptive ratio for sampling expert demonstrations from a separate demonstration replay buffer[14] (Skrynnik et al., 2019). The second-place team, mc_rl, trained their hierarchi-

---

14. A video example of CDS's trained agent can be found here.

|  | Use hierarchical RL | Use human data | Use action reduction |
|---|---|---|---|
| Number of teams | 6 | 7 | 9 |
| Percent of teams | 67% | 78% | 100% |

Table 2: Overview of general approaches taken by the top 9 teams.

cal policies entirely from human demonstrations with *no* environment interactions[15]. The third-place team, i4DS, used demonstrations to bootstrap their hierarchical RL algorithm, effectively using the human data to pretrain the models to predict recorded human actions from observations, then used RL to refine the resulting agents (Scheller et al., 2020).

The fourth-place team, CraftRL, aggregated grounded actions into options (Sutton et al., 1999) and then used a meta-controller to select between options. They pretrained the meta-controller on demonstration data using behavioral cloning. The fifth-place team, UEFDRL, used a single deep residual neural network trained to mimic human actions from the MineRL dataset. This team used action discretization to deal with the complexity of the environment. The sixth-place team, TD240, used a discriminator soft actor critic, which uses a discriminator based on adversarial inverse RL as a reward function.

The seventh-place team, LAIR, used meta-learning shared hierarchies (MLSH) (Frans et al., 2018), a hierarchical RL algorithm, and pretrained the master and subpolicies with human data before training MLSH with environment interactions. The eighth-place team, Elytra submitted a Rainbow baseline with modifications, including adding a small amount of stochasticity to the environment and restricting the camera movement to 1 degree of freedom. However, they explored the idea of training an ensemble of value functions and believe this approach to be promising. The ninth-place team, karolisram, used the PPO baseline with modifications, including removing frame-skip and reducing the action space.

### 4.3. Special Awards

In addition to evaluating teams on the performance of their learning algorithms, we awarded teams prizes for research contributions. Since this was the first year in which the competition was held, we awarded two prizes to extreme and opposite research paradigms: one based purely on imitation learning and the other one based purely on RL, with no use of human priors. We awarded the main research prize to mc_rl for their imitation-learning-based approach, and the runner-up research prize to karolisram for their purely-RL-based approach. We hope that these special awards will encourage future participants to maximally explore the space of solutions, even if this comes at the expense of performance.

## 5. Organizational Outcomes

We describe important organizational outcomes of the competition. We begin by presenting the impact of our chosen rule set in Section 5.1. Then, in Section 5.2, we discuss the effectiveness of our distribution of competition materials. Finally, we summarize the effects of our chosen evaluation methodology in Section 5.3.

---

15. A video example of mc_rl's trained agent can be found here.

### 5.1. Repercussions of the Rules

Unlike many other DRL competitions, we sought to limit the use of domain knowledge through feature engineering and hard-coded policies. Though this ruleset has led to the use of more general methods, it also led to a lot of clarifying questions from participants. Since these were asked across various communication channels, there were duplicate questions and no single, comprehensive answer. In future renditions of this competition, we will have a clear place for rule clarifications and a more robust restriction on use of domain knowledge.

A secondary effect of the limitation on domain knowledge use was the difficulty in checking Round 1 submissions for this rule violation. This rule was the only one which led to complications of this type; restrictions on training time and number of samples are quantitative restrictions and therefore easy to enforce. Since competitor's submissions were not trained on a "hold out" environment, a committee spent more work hours than expected manually reviewing the code to identify manually specified policies and other violations.

### 5.2. Effectiveness of Competition Materials

As mentioned in Section 3.3, we provided participants with the tasks as Gym environments, the dataset, and baseline method implementations. The environments and dataset were both complete before the competition began, but the baselines were developed after the competition was announced because of delays providing the environment to our collaborators, Preferred Networks. However, Preferred Networks worked quickly to provide the baselines in time for participants to use them in Round 1, and many participants found these baselines to be crucial in developing their own algorithms. In future iterations of this competition, we will be able to provide the baselines as soon as the competition begins now that the baselines are complete.

The data was readily available and participants were generally able to successfully use it. The provided Gym environments were a wrapper around the Malmo Minecraft environment (Johnson et al., 2016). In addition to wrapping Malmo in a standardized way, we fixed bugs and added functionality which allows faster and more efficient training. These modifications have increased interest in working on Minecraft, and we have been contacted by several academic groups about further extensions to the environment.

Competitor submissions had to adhere to a specific Docker format, as outlined in Section 3.2. Use of a standard container streamlined the evaluation process during both rounds, and it saved time from our end. Although we provided information about how to use the provided container, based on feedback from participants, we plan to extend the documentation to include a step-by-step demonstration of how to submit an agent using the provided structure.

### 5.3. Impact of Evaluation Methodology

We chose to have two rounds to balance the need for an explicit computational budget with a desire to not limit participation in the competition. We did not restrict the number of teams who could participate during the Round 1; however, we chose a pre-specified number of teams to move to Round 2. As a result, we could commit to retraining all submissions during Round 2. This commitment was possible due to the strict limits on training time as well as the selection of a specific system on which all training and evaluation would

be performed. Due to this structure, we did not use more computational resources than expected for evaluation.

However, these same strict limits on training time led to delays in obtaining final results. We delayed the submission deadlines for competitors; as a result, this reduced the time before we were scheduled to announce results. Since training had to happen for a fixed number of days on a specific system, we could not parallelize an individual competitor's submission or run it on faster hardware. Despite the delay, we will use a similar restriction in future competitions due to the reproducibility and sample efficiency benefits.

As mentioned in Section 3.3, we provided each team in Round 2 with a compute grant of $1500USD in Azure credits to enable training and validation of their methods. Because of the training time restrictions, participants were not incentivized to spend the remainder of their compute time on extending the training time of their final approach. We believe we allocated a sufficient amount to each team, since some teams had enough compute time remaining to continue to develop their methods even after the competition was over.

Finally, although we effectively utilized a novel visual domain randomization method to ensure generalization and rule compliance, competitors spent a great deal of effort shaping the action space using inductive bias. To yield fully generalizable competition results in future iterations of this competition, we plan to remove semantic labels of the environment's action space and apply a random bijective-mapping to the action space during evaluation.

## 6. Conclusion

We ran the MineRL Competition on Sample Efficient Reinforcement Learning Using Human Priors at NeurIPS2019 in order to promote the development of algorithms that: 1) use human demonstrations alongside reinforcement learning, 2) are sample efficient, and 3) are accessible to the general AI community. We summarized the submissions and looked at the performance of competitors in different rounds. We discussed the effects of the competition rules, the provided competition materials, and the evaluation methodology. Based on the largely positive outcome of the competition, we plan to hold the competition again. Because no team obtained a diamond, we plan to focus on the same task, `ObtainDiamond`. We also plan to keep the same paradigm promoting the development of generalizable and robust learning procedures. When we hold the competition again, we will clarify the rules and provide more demonstrations of how to participate in order to increase participation.

## Acknowledgments

## References

D. Abel, D. E. Hershkowitz, G. Barth-Maron, S. Brawner, K. O'Farrell, J. MacGlashan, and S. Tellex. Goal-based action priors. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling*, 2015.

D. Abel, A. Agarwal, F. Diaz, A. Krishnamurthy, and R. Schapire. Exploratory gradient boosting for reinforcement learning in complex domains. In *The 33rd International Conference on Machine Learning Workshop on Abstraction in Reinforcement Learning*, 2016.

R. Alterovitz, S. Koenig, and M. Likhachev. Robot planning in the real world: research challenges and opportunities. *AI Magazine*, 37(2):76–84, 2016.

K. C. Aluru, S. Tellex, J. Oberlin, and J. MacGlashan. Minecraft as an experimental world for AI in robotics. In *2015 AAAI Fall Symposium Series*, 2015.

D. Arumugam, J. K. Lee, S. Saskin, and M. L. Littman. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv: 1902.04257*, 2019.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

R. Dubey, P. Agrawal, D. Pathak, T. L. Griffins, and A. A. Efros. Investigating human priors for playing video games. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman. Meta learning shared hierarchies. *Proceedings of the 6th International Conference on Learning Representations*, 2018.

S. J. Frazier and M. O. Riedl. Improving deep reinforcement learning in Minecraft with action advice. In *Proceedings of the 15th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2019.

Y. Fujita, T. Kataoka, P. Nagarajan, and T. Ishikawa. ChainerRL: A deep reinforcement learning library. In *The 33rd Conference on Neural Information Processing Systems Workshop on Deep Reinforcement Learning*, 2019.

W. H. Guss, C. Codel*, K. Hofmann*, B. Houghton*, N. Kuno*, S. Milani*, S. Mohanty*, D. Perez Liebana*, R. Salakhutdinov*, N. Topin*, M. Veloso*, and P. Wang*. The MineRL competition on sample efficient reinforcement learning using human priors. In *The 33rd Annual Conference on Neural Information Processing Systems Competition Track*, 2019.

W. H. Guss*, B. Houghton*, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys. Learning from demonstrations for real world reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29*. 2016.

B. Houghton, S. Milani, N. Topin, W. H. Guss, K. Hofmann, D. Perez-Liebana, M. Veloso, and R. Salakhutdinov. Guaranteeing reproducibility in deep learning competitions. In *The 33rd Annual Conference on Neural Information Processing Systems Challenges in Machine Learning Workshop*, 2019.

M. Johnson, K. Hofmann, T. Hutton, and D. Bignell. The Malmo platform for artificial intelligence experimentation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016.

A. Juliani, A. Khalifa, V.-P. Berges, J. Harper, H. Henry, A. Crespi, J. Togelius, and D. Lange. Obstacle tower: A generalization challenge in vision, control, and planning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

L. Kidziński, S. P. Mohanty, C. Ong, J. L. Hicks, S. F. Carroll, S. Levine, M. Salathé, and S. L. Delp. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. *The NIPS '17 Competition: Building Intelligent Systems*, 2018.

J. Master, E. Patterson, S. Yousfi, and A. Canedo. String diagrams for assembly planning. *arXiv preprint arXiv: 1909.10475*, 2019.

A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman. Gotta learn fast: A new benchmark for generalization in RL. *arXiv preprint arXiv:1804.03720*, 2018.

J. Oh, V. Chockalingam, S. P. Singh, and H. Lee. Control of memory, active perception, and action in Minecraft. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

D. Perez-Liebana, K. Hofmann, S. P. Mohanty, N. Kuno, A. Kramer, S. Devlin, R. D. Gaina, and D. Ionita. The multi-agent reinforcement learning in Malmo (MARLO) competition. In *The 32nd Annual Conference on Neural Information Processing Systems Challenges in Machine Learning Workshop*, 2018.

M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3(4):4423–4430, 2018.

C. Scheller, Y. Schraner, and M. Vogel. Sample efficient reinforcement learning through learning from demonstrations in Minecraft. *arXiv preprint arXiv:2003.06066*, 2020.

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

T. Shu, C. Xiong, and R. Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

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

A. Skrynnik, A. Staroverov, E. Aitygulov, K. Aksenov, V. Davydov, and A. I. Panov. Hierarchical deep q-network with forgetting from imperfect demonstrations in Minecraft. *arXiv preprint arXiv: 1912.08664*, 2019.

R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112, 1999.

C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor. A deep hierarchical approach to lifelong learning in Minecraft. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

A. Trott, S. Zheng, C. Xiong, and R. Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. In *Advances in Neural Information Processes 32*. 2019.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 2019.

R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone. Leveraging human guidance for deep reinforcement learning tasks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.