Towards robust and domain agnostic
reinforcement learning competitions: MineRL 2020

William Hebgen Guss  
Stephanie Milani  
Nicholay Topin  
Brandon Houghton  
Sharada Mohanty

Andrew Melnik  
Augustin Harter  
Benoit Buschmaas  
Bjarne Jaster  
Christoph Berganski  
Dennis Heitkamp  
Marko Henning  
Helge Ritter  
Chengjie Wu  
Xiaotian Hao  
Yiming Lu  
Hangyu Mao  
Yihuan Mao  
Chao Wang  
Michal Opanowicz  
Anssi Kanervisto  
Yanick Schraner  
Christian Scheller  
Xiren Zhou  
Lu Liu  
Daichi Nishio  
Toi Tsuneda  
Karolis Ramanauskas  
Gabija Juceviciute

Editors: Hugo Jair Escalante and Katja Hofmann

Abstract

Reinforcement learning competitions have formed the basis for standard research benchmarks, galvanized advances in the state-of-the-art, and shaped the direction of the field. Despite this, a majority of challenges suffer from the same fundamental problems: participant solutions to the posed challenge are usually domain-specific, biased to maximally exploit compute resources, and not guaranteed to be reproducible. In this paper, we present a new...
framework of competition design that promotes the development of algorithms that overcome these barriers. We propose four central mechanisms for achieving this end: submission retraining, domain randomization, desemantization through domain obfuscation, and the limitation of competition compute and environment-sample budget. To demonstrate the efficacy of this design, we proposed, organized, and ran the MineRL 2020 Competition on Sample-Efficient Reinforcement Learning. In this work, we describe the organizational outcomes of the competition and show that the resulting participant submissions are reproducible, non-specific to the competition environment, and sample/resource efficient, despite the difficult competition task.

**Keywords:** Reinforcement learning competitions, Minecraft, Sample Efficiency, Imitation Learning

1. Introduction

Deep reinforcement learning has emerged as a compelling solution to a wide range of problems in machine learning. Techniques from this field have been successfully applied to a number of difficult domains such as real-time video games (Berner et al., 2019; Vinyals et al., 2019b), complicated control and scheduling problems, real-world robotic manipulation tasks, and self-driving. The success of deep reinforcement learning (RL) has been accompanied by an increase in RL competitions spanning a number of domains and difficult open problems (Guss et al., 2019; Perez-Liebana et al., 2019; Koppejan and Whiteson, 2009). As competitions mature, they form the basis for benchmarks used throughout the community (Machado et al., 2018; Bellemare et al., 2013). Typically, competitions focus on a core problem with current RL algorithms or domain(s) not yet readily solved by current methods, and challenge competitors to train agents (using competitor resources) to solve domain(s). These agents are then submitted to an evaluation platform and ranked based on final performance.

Despite this common format for research benchmarks, a number of issues have become apparent. In competitions where only final trained agents are submitted, the algorithmic underpinnings of submissions become difficult to reproduce: winning solutions are often trained with a disproportionately larger compute resource budget to that of other competitors (Guss et al., 2019); competitors often choose to train on a specific set of environment seeds and benefit greatly from large-scale hyperparameter searches, making reimplementations and broader use more difficult (Khetarpal et al., 2018); and training code is sometimes not shared when only inference code is submitted, preventing validation of the algorithmic claims of the submission and allowing the submission to be trained using hard-coded, engineered features or action and reward shaping (Houghton et al., 2020). Furthermore, competitions tied to a specific unrandomized domain can fail to yield direct algorithmic advancements, as the most successful methods commonly overfit to and exploit the specific structure of the problem. Similarly, multi-year competitions reward increases in domain knowledge exploitation, reducing the role of algorithmic novelty. For domains of specific real-world importance, such as robotics or self-driving, the task solution has greater utility than any resulting secondary algorithmic advancements. However, there is a large gap between domain-specific submissions to RL competitions on video-game or artificial domains and their downstream utility in the research community.
To address these problems, we proposed, organized, and ran the MineRL 2020 Competition on Sample Efficient Reinforcement Learning using Human Priors (Guss et al., 2021). Our competition utilized several novel mechanisms for yielding robust and domain agnostic submissions, including observation and action space obfuscation, submission retraining, domain randomization, and environment interaction limits. In this paper, we present the general methodologies and design principles that comprise the competition structure and describe the resulting top-performing submissions. Section 2 provides a general background for the problem settings that motivate the competition. In Section 3, we give an overview of the competition, including its design, central task, rules, and resources provided. In Section 4, the top teams describe the approaches used in their submissions. Thereafter, in Section 5 we discuss the organizational outcomes of our competition with respect to our goal of robust, reproducible, and high quality solutions. Finally, we position the MineRL competition in the context of other concurrent and past RL competitions in Section 6 and discuss challenges and opportunities for future work in Section 7.

2. Background

2.1. The sample inefficiency problem

Many of the most celebrated successes of machine learning, such as AlphaStar (Vinyals et al., 2019a), AlphaGo (Silver et al., 2017), OpenAI Five (Berner et al., 2019), and their derivative systems (Silver et al., 2018), utilize deep RL to achieve human or super-human level performance in sequential decision-making tasks. These improvements to the state-of-the-art have thus far required exponentially increasing computational power (Amodei and Hernandez, 2018), which is largely due to the number of environment-samples required for training. These growing computational requirements prohibit many in the AI community from improving these systems and reproducing state-of-the-art results. Additionally, the application of many reinforcement learning techniques to real-world challenges, such as self-driving vehicles, is hindered by the raw number of required samples.

A variety of approaches have been proposed towards the goal of sample efficiency, including learning a model of the environment (Buckman et al., 2018), leveraging AutoRL to perform efficient hyperparameter optimization (Franke et al., 2020), and incorporating domain information in the form of human priors and demonstrations (Dubey et al., 2018; Pfeiffer et al., 2018). In our competition we encourage the use of any techniques that improve the sample efficiency of RL algorithms without using domain-specific hard-coding.

2.2. Generalization

The development of RL algorithms that can generalize is of great importance to the research community (Zhang et al., 2018; Cobbe et al., 2019; Malik et al., 2021). There are various notions of generalization, but it is broadly defined as the ability of an agent to learn desired behavior and perform well in similar environments. Through our competition, we want to promote the development of algorithms that generalize across different domains and tasks, such as those with different state and action spaces.
2.3. Minecraft

The central competition task, ObtainDiamond, is set in the Minecraft domain. Minecraft is a popular and compelling environment for the development of reinforcement learning (Oh et al., 2016; Shu et al., 2017; Tessler et al., 2017) and imitation learning methods because of the unique challenges it presents. Notably, the procedurally-generated world is composed of discrete blocks that allow modification. Over the course of gameplay, players change their surroundings by gathering resources and constructing structures. Since Minecraft is a 3D, first-person, embodied domain and the agent’s surroundings are varied and dynamic, it presents many of the same challenges as real-world robotics domains, like determining a good representation of the environment and planning over long time horizons.

3. Competition Overview

In line with its previous iteration (Guss et al., 2019; Milani et al., 2020), the MineRL 2020 Competition challenges teams to submit reproducible (Houghton et al., 2020) training code for an agent that can solve a complex, long time-horizon task with robustness to environment domain-shift under a strict sample and computational budget. We describe the competition design in Section 3.1, including the mechanisms we implemented to ensure the development of robust and sample efficient learning algorithms. The primary competition task is the MineRL ObtainDiamond environment, which we detail in Section 3.2. We summarize the rules of the competition in Section 3.3. Our methodology for evaluating submitted algorithms is explained in Section 3.4. To assist participants with developing their learning algorithms, we provide them with a number of important resources, including a set of reinforcement learning and imitation learning baselines, which we describe in Section 3.5.

3.1. Competition Design

The MineRL Competition is designed to promote the development of robust, domain agnostic, and sample efficient algorithms for solving complex, long time-horizon tasks with sparse rewards using human priors.

Reproducibility and Sample Efficiency. To yield sample efficient algorithms, we provide participants with the 60 million frame MineRL-v0 human demonstration dataset (Guss et al., 2019) of the competition task. These samples allow the use of imitation learning techniques, which can drastically reduce the number of resources and samples required to solve complex tasks. Gathering expert demonstrations is practical for many RL environments, so this approach can be applied in many competitions.

To further ensure reproducibility and sample efficiency, we retrain participant submissions during Round 2. In addition, to directly address the problem of disproportionate and limited access to computing resources across the AI research community, we deliberately limit the hardware available for training; this further enables the democratization of AI research and the development of novel AI techniques with a low-barrier to reproduction. In Round 1, participants develop and train a learning algorithm on the environment using a fixed hardware and compute budget (1 NVIDIA P100 GPU for 4 days) and a fixed number of environment samples (8,000,000 frames or approximately 114 hours of game time). This compute budget was chosen as it represents an upper bound on consumer hardware;
Figure 1: The action and observation space obfuscation mechanism; a randomized autoencoder trained to encode the entire mixed discrete-continuous observation and action space into a compact ball with which participants’ models interact. This obfuscation prevents action shaping and feature engineering and enables domain randomization during Round 2.

Robustness and Domain Agnosticism. Preventing domain specific solutions is a difficult task. As in the previous iteration of the MineRL Competition, during Round 2, the environment and dataset is randomized (textures are remapped, action effects are randomized, and game dynamics are changed), thus penalizing submissions which rely on domain specific strategies and feature engineering. Despite this mechanism in last year’s competition, participants leveraged small, shaped subpolicies. These subpolicies were not robust to domain shifts because they depended on the semantics of the environment.

In this year’s competition, we introduce a novel obfuscation scheme that prevents domain specific hard-coding. Specifically, we learn a random, volume-preserving embedding that takes semantically-labeled actions and observations (e.g., crafting or inventory items) and obfuscates them into feature vectors. This scheme is agnostic to the environment, as algorithms trained to solve environments with this vector observation and action space can be immediately retrained against a different environment given a corresponding embedding of the new environment’s action and state spaces.

Shown in Figure 1, we obtain this embedding with careful considerations of injectivity and surjectivity. Let $X \subset A$ be some bounded action/observation space (discrete or continuous), $P_X$ be the default sampling distribution for that space (often uniform). Let $Z$ be some bounded subset of $\mathbb{R}^n$ into which we wish to obfuscate $X$. In the MineRL Compe-
tion, \( Z = [-1, 1]^n \) where \( n \) is the length of the obfuscated feature vector. Let \( d_X(u, v) \)
be a natural reconstruction metric for \( X \) (normed difference squared for continuous \( X \),
and cross entropy for discrete \( X \)). Let \( g_\theta : A \rightarrow \mathbb{R}^n \) and \( f_\theta : \mathbb{R}^n \rightarrow A \) be encoder
and decoder networks. We train these maps to encode the original observation/action space \( X \) respecting the bounds of the space by minimizing reconstruction loss while maintaining
that \( g_\theta(X) \subset Z \) and \( f_\theta(Z) \subset X \); that is, we define our reconstruction loss as
\[
L_X(\theta) = \mathbb{E}_{y \sim P_X}[d_X(f_\theta(g_\theta(y)), y) + \text{Hinge}(g_\theta(y), Z)] \\
+ \mathbb{E}_{z \sim \text{Unif}(Z)}[\text{Hinge}(f_\theta(z), X)]
\]
where \( \text{Hinge}(u, V) \) is a hinge loss which is zero when \( u \) is in \( V \) and piecewise linear, increasing
otherwise. For example, for \( Z \), a simple box space, \( \text{Hinge}(z, [-1, 1]^n) = \sum_{i=1}^{n} \text{ReLU}(|z_i| - 1) \).
This loss accomplishes two goals: when the competitors use \( Z \) as an action space, they
are approximately guaranteed to have a valid action \( f_\theta(z) \) in \( X \) when sampling within
the bounds of \( Z \). Furthermore, the reconstruction loss ensures that all possible actions
in \( X \) can be taken by finding a point in \( Z \). Likewise, when using \( Z \) as an observation
space, all observations in the unobfuscated action space \( X \) are approximately guaranteed
to be contained inside of \( Z \). Therefore, competitors can appropriately normalize their
observations in \( Z \). It is important that \( Z \) is of the same or higher dimensionality than \( X \)
so there is certain to be a minimizer \( \theta^* \). In the MineRL Competition, both action and
observation space embeddings were trained with \( \text{dim}(Z) = 64 \) to an error of at most \( 1e^{-12} \);
we chose this space as it was of high enough dimension to embed the observation and action
space but of low enough dimension that reinforcement learning algorithms converge within
the compute budget.

3.2. Task

The primary task of the competition is ObtainDiamond. Agents begin at a random position
on a randomly-generated Minecraft map with no items in their inventory. Completing
the task consists of controlling an embodied agent to obtain a single diamond, which can
only be accomplished by navigating the complex item hierarchy of Minecraft. The learning
algorithm has direct access to a 64x64 pixel point-of-view observation from the perspective
of the embodied agent, as well as a set of discrete observations of the agent’s inventory for
every item required for obtaining a diamond. 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 once per episode for reaching a set of
milestones of increasing difficulty that form a set of prerequisites for the full task. Table 1
depicts the full reward structure.

Progress towards solving the ObtainDiamond environment under strict sample complexity constraints lends itself to the development of sample-efficient—and therefore more computationally accessible—sequential decision-making algorithms. In particular, because we maintain multiple versions of the dataset and environment for development, validation, and evaluation, it is difficult to engineer domain-specific solutions to the competition challenge. The best performing techniques must explicitly implement strategies that efficiently leverage human priors across general domains.
3.3. Rules

Due to the unique competition paradigm, we provide a strict set of rules to ensure high-quality submissions. We prohibit teams from manually engineering the reward function, action space, and observation space. For example, we permit curiosity rewards but not bonus rewards for encountering specific objects; we allow a learned hierarchical controller but not one that switches between policies based on manually-specified conditions; we allow agents to act every even-numbered timestep based on the previous two observations but prohibit the application of manually specified edge detectors to the observation. Furthermore, we require that competitors’ code make no semantic reference to the environment. To encourage reproducible submissions, we require entries to the competition to be open: teams must reveal most details of their method, including the source code. Moreover, the competition features two sets of tracks, each of which have distinct rules. The RL + IL Track features methods that leverage both samples from the environment and human demonstrations, while algorithms in the IL-only Track must only use imitation learning with no access to the environment — except for during evaluation.

3.4. Evaluation

Submission Platform. The submissions for the competition were evaluated using AIcrowd, which has been used in numerous RL benchmarks (Juliani et al., 2019; Mohanty et al., 2020), and allows for the much needed flexibility when designing complex benchmarks. Participating teams independently develop their solutions using Git repositories provided by AIcrowd. The repositories include simple configurations specifying the expected software runtime. They also have a prescribed structure to enable clear specifications of code entry points for different phases of evaluation. Participants submit to the benchmark by releasing Git tags, which trigger the evaluation workflow. The workflow then builds a Docker image with the submitted code repository, which is automatically orchestrated on a scalable Kubernetes cluster. During evaluation, the evaluators provide real-time feedback on the progress of the evaluation and submission-specific metrics to the participants. If a submission fails, participants can debug their submission using the readily-available logs. To avoid data leak, care is taken to ensure that logs are not made available for sensitive phases of the evaluation. On successful completion of the evaluation workflow, the evaluators update the scores, and any generated assets on the competition leaderboard.

Metrics. When models are submitted in Round 1 and after they are trained by organizers in Round 2, participants are evaluated on the average score of their model over 200 episodes. Scores are computed as the sum of the milestone rewards (shown in Table 1) achieved by the agent in a given episode. Ties are broken by the number of episodes required to achieve the last milestone.

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Reward</th>
<th>Milestone</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>log</td>
<td>1</td>
<td>furnace</td>
<td>32</td>
</tr>
<tr>
<td>planks</td>
<td>2</td>
<td>stone_pickaxe</td>
<td>32</td>
</tr>
<tr>
<td>stick</td>
<td>4</td>
<td>iron_ore</td>
<td>64</td>
</tr>
<tr>
<td>crafting_table</td>
<td>4</td>
<td>iron_ingot</td>
<td>128</td>
</tr>
<tr>
<td>wooden_pickaxe</td>
<td>8</td>
<td>iron_ingot</td>
<td>256</td>
</tr>
<tr>
<td>stone</td>
<td>16</td>
<td>diamond</td>
<td>1024</td>
</tr>
</tbody>
</table>

Table 1: Rewards for sub-goals and main goal (diamond) for Obtain Diamond.
MineRL 2020: robust and domain agnostic reinforcement learning

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

<table>
<thead>
<tr>
<th>Baselines</th>
<th>Round 1</th>
<th>Round 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Score</td>
<td>Team Name</td>
</tr>
<tr>
<td>SQIL</td>
<td>2.94</td>
<td>HelloWorld</td>
</tr>
<tr>
<td>DQFD</td>
<td>2.39</td>
<td>NoActionWasted</td>
</tr>
<tr>
<td>Rainbow</td>
<td>0.42</td>
<td>michal_opanowicz</td>
</tr>
<tr>
<td>PDDDQN</td>
<td>0.11</td>
<td>CU-SF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cit-ec.de</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NuclearWeapon</td>
</tr>
<tr>
<td></td>
<td></td>
<td>murarinCraft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RL4LYFE</td>
</tr>
</tbody>
</table>

Table 2: Scores of the baselines (left) and the best-performing submissions from Round 1 (middle) and Round 2 (right).

3.5. Resources

In addition to providing the MineRL-v0 dataset (Guss* et al., 2019), we give participants an open-source Github repository with starting code, including an OpenAI Gym template interface, a data-loader, a Docker container, and the code for the solutions created by last year’s top participants. We also provide participants with a set of four state-of-the-art baselines that they could readily submit. Implemented by the organizers from Preferred Networks, these baselines consist of Soft-Q Imitation Learning, (SQIL) (Reddy et al., 2020), Deep-Q From Demonstrations (DQFD) (Hester et al., 2018), Rainbow Deep-Q Networks (Rainbow) (Hessel et al., 2018), and Prioritized Dueling Double Deep Q-Networks (PDDDQN) (Schaul et al., 2015; Van Hasselt et al., 2016; Wang et al., 2016). The baselines all used K-means clustering (MacQueen et al., 1967; Lloyd, 1982) to discretize the action space.

4. Solutions

We provide an overview of the submissions made by the participants of our competition. In Section 4.1, we describe the performance of the submissions and how they compare to the performance of submissions to the 2019 competition. The remaining sections summarize the techniques used by the competitors in their submissions.
4.1. Submission Performance Overview

The conditions surrounding the competition and the changes to the competition itself proved to make the competition more challenging for the competitors compared to last year. Although this year’s competition enjoyed participation from more teams than last year’s (95 vs. 47), there were fewer submissions overall (513 vs. 662). We believe that this decrease in submissions is in part due to the global pandemic. Teams still performed well: in Round 1, 36 teams achieved a non-zero score, and 17 of these teams outperformed the best-performing baseline, SQIL. In Round 2, seven teams achieved a non-zero score and some teams performed even better than they did in Round 1.

Table 2 shows the scores of the best-performing submissions from both rounds of the 2020 competition. The average scores of the top nine competitors in the 2020 competition were 8.13 for Round 1 and 16.42 for Round 2. In contrast, the average scores of the top nine competitors in the 2019 competition were 31.77 for Round 1 and 25.84 for Round 2. Surprisingly, the standard deviation of the top nine scores from Round 1 of 2020 was smaller than the standard deviation of the top nine scores from Round 1 of 2019 (5.72 and 10.22, respectively). This finding may be due to the overlap of techniques used by the competitors: at least four of the top nine competitors in the 2020 competition leveraged a similar K-means clustering of the action space. Figure 2 depicts the maximum item score for each team over the evaluation episodes in Round 2. Although no team obtained a diamond, many of the top teams progressed quite far along the item hierarchy.

4.2. Team 1: cit-ec.de

Overall, team cit-ec.de placed fifth in Round 1 (score of 6.40) and first in Round 2 (score of 72.510). They also placed first in the IL-only track, but they switched to the RL + IL Track in Round 2. Inspired by the recent success of cognitive science research (Melnik et al., 2018b; König et al., 2018) and its applications in artificial intelligence systems (Melnik et al., 2019; Konen et al., 2019; Bach et al., 2020; Melnik et al., 2018a; Harter et al., 2020; Schilling and Melnik, 2018), their approach aims to learn to detect object-centric representations from pixels (Simonyan et al., 2013) using rewarding signals of interaction with the environment.

Depicted in Figure 3 (top), they train a U-Net model to generate masks over reward-related objects in images. This approach enables the training of the U-Net model without explicit label information. Instead, they perform this training in a contrastive fashion with image pairs using an adversarial scheme employing the critic score gradient with respect to the mask operation. The pair consists of two images, where the first has a high and the second a low critic value. Training with such pairs enables the U-Net to produce masks that decrease the critic value in the first image and increase the critic value in the second image when transferring pixels in the masked segment from the first to the second image. The critic learns to estimate the expected-reward value of an image observation using experience replay buffer collected from human player demonstrations. As shown in Figure 3 (bottom), this approach of training the U-Net model showed encouraging results of segmentation of rewarding objects in the competition (Melnik et al., 2021).

4.3. Team 2: HelloWorld
Figure 3: cit-ec.de’s method. Top: First phase (highlighted in red): Image A (high critic value) passes through the U-Net, forming a mask M. Second phase: the mask M is used to merge image A (high critic value) with image B (low critic value) resulting in image X (masked parts of A replaced with B) and image Y (masked parts of A injected in B). Images A, B, X, and Y are then passed through the encoder and critic. The losses penalize differences in critic values for image pairs A : Y, and B : X. Mask-size loss prevents a trivial solution when M takes the full image. Bottom: Segmentation results: The U-Net model learns to segment tree trunks without any label information but only from reward signals. It generalizes well between different positive and negative reward scenarios. The first row shows the input images, the second row shows the masked segments of the input images, and the third row shows the U-Net generated masks.

Figure 4: HelloWorld’s method. Team HelloWorld placed first in Round 1 (score of 19.840) and second in Round 2 (score of 39.55). They competed in the RL + IL track. Shown in Figure 4, their approach splits each episode into several stages based on accumulated rewards. They train one policy for each stage of the episode to enable each subpolicy to capture different information towards different goals. They train a meta-policy to select a subpolicy for execution at each timestep. For each subpolicy, they select the best algorithm from all of the tested ones, including behavioral cloning, BCQ (Fujimoto et al., 2019), SQIL.
DQfD, POoD (Kang et al., 2018), and Rainbow. They find that SQIL incorporated with self-imitation avoids performance drop during training. Finally, their solution consists of other key components to make their approach more robust and generalizable, including pre-training agents on simpler environments, leveraging auxiliary tasks (e.g., predicting future state), and applying state augmentation (Yarats et al., 2021).

4.4. Team 3: michal_opanowicz

Team michal_opanowicz achieved third place in Round 1 (score of 9.290) and Round 2 (score of 13.290). However, this team achieved first place overall in the IL-only track. They use imitation learning, in which the problem is framed as a classification task where the agent predicts the human player’s action at each environment step. To produce discrete labels for this task, they quantize the actions using K-means with 120 clusters. During evaluation, they randomly sample the actions from the cluster means with the network’s prediction used as a probability distribution.

For visual processing, they use the ResNet (He et al., 2015) architecture with FixUp initialization (Zhang et al., 2019), that was proposed for this task in a MineRL 2019 submission (Amiranashvili et al., 2020). They process non-visual observations with a fully connected layer with ReLU activation and concatenate it with the ResNet outputs. They are then processed by a LSTM (Hochreiter and Schmidhuber, 1997) and two fully connected layers to produce the final prediction. Notably, they add the LSTM inputs to its outputs to form a residual connection similar to ResNets. They train the network on 100-step sequences of observation-action pairs, with the final LSTM state in a sequence used as the initial state for the next sequence. Figure 5 shows an analysis of their algorithm’s performance. In most runs, their agent obtains either a crafting table or a stick.

4.5. Team 4: NoActionWasted

Team NoActionWasted competed in the IL-only track. Overall, they achieved second place in Round 1 and fourth in Round 2 (scores of 16.48 and 12.79, respectively). Depicted in Figure 6, their system consists of a ResNet-LSTM network (Espeholt et al., 2018) trained to predict human actions. They found that directly training agents on the obfuscated actions was not successful, so they discretized the action space into 150 clusters with K-means. This step was crucial to obtain good initial behavior. When using a lower amount of clusters, some rare but important actions are not represented. In-
stead of fixing the frame-skip parameter, they train the network to predict it as an action parameter, which provides significant benefit for agents in the Minecraft domain, as many tasks require repeating the same action multiple times, and it reduces the perceived episode length.

They filter the dataset to only include successful games, which consistently improved the performance. They fine-tune the system with IMPALA (Espeholt et al., 2018). Following their submission last year (Scheller et al., 2020), they pair this algorithm with extensive experience replay (Lin, 1992), clipping advantages to promote exploitation of good behavior, and CLEAR (Rohnick et al., 2018) to combat catastrophic forgetting. Crucially, they find that large batch sizes were crucial for stability during the RL fine tuning.

4.6. Team 5: Rabbits

Team Rabbits achieved fifth place in Round 2 (score of 5.16). Although they competed in both tracks, their highest-performing submission was from their RL + IL track submission. For their approach in both tracks, they split the overall task into 10 subtasks based on reward, transforming it into a hierarchical learning problem. They use a task-classification network to determine which of the 10 reward stages the agent is currently in. The task-classification network only takes the state vector as input. To learn the subtasks, they use 10 separate Q networks. For the IL-only track submission(s), they employ Regularized Behavior Cloning (RBC) (Piot et al., 2014) on the demonstration data. They train each Q network separately, stopping when the TD error no longer decreases. For their RL + IL submission(s), they apply RBC as the starting point for RL. The RL algorithm they use is a variant of SQL: they set the reward in expert replay buffer to be half of the reward of the corresponding task, except for the steps that obtain the sparse reward.

4.7. Team 6: MajiManji

Team MajiManji achieved sixth place in Round 2 (score: 5.16). They participated in the RL + IL track. Their approach uses hierarchical offline reinforcement learning. Outlined in Figure 7, they decompose the high-level task of mining a diamond into a set of subtasks, and train a separate CQL agent (Kumar et al., 2020) for each subtask. To determine these subtasks, they use a label encoder that maps cumulative reward to a label. The high-level policy leverages the subtask label to select the low-level policies. In the environments ending with “VectorObf”, which have both POV images and vectors as states, they use ATC (Stooke et al., 2020) to extract features only from the POV images.

They apply a K-means algorithm to the action space for each subtask. In addition, they construct a “necessary action space”. An action is added to this action space when the total number of times it is performed is few, but it is performed in most episodes. Since the low-level policies may not learn to do these necessary actions, the high-level policy chooses...
a necessary action randomly with low probability $\epsilon$. In the future, they plan to find how to weight important demonstrations with no reward and few samples.

### 4.8. Team 7: BeepBoop

Team **BeepBoop** achieved twelfth place in Round 1 (score: 3.110) and seventh in Round 2 (score: 1.970). In Round 1, they participated in the RL + IL track, where they simply used the SQIL baseline. For Round 2, they participated in both tracks. Their RL + IL track submission was designed with the goal of improving the DQfD baseline. First, they improved data preparation: since their agent never reached the later stages of the challenge, they only use the data from the earlier stages of the **ObtainDiamond** and **ObtainIronPickaxe** environments, as measured by cumulative rewards. They also included the full TreeChop dataset. Second, they focused on improving the action space representation. They use K-means clustering with a large number of clusters to cluster the action space to ensure that the agent could access all necessary actions. For their IL-only track submission, they used behavioral cloning with the ResNet-50 network architecture (He et al., 2015). For the labels, they used the centroids of K-means clustered obfuscated actions. They apply the same data preparation and action space modification techniques as in the RL+IL track submission.

### 5. Discussion

Promoting the development of sample-efficient, domain agnostic, and reproducible RL algorithms is crucial in translating the research advances of RL into complex, real-world settings. In this section, we discuss the extent to which the design principles behind the MineRL competition have accomplished this goal.

**Successes.** As the results of Section 4 illustrate, the submissions to the competition are general and successful in the face of the competition constraints. Competitors submitted sample-efficient algorithms that made substantial progress towards the difficult **ObtainDiamond** task with a mere 8,000,000 frames from the environment. Further, despite the domain randomization in Round 2, many of the top algorithms from Round 1 also achieved impressive scores in Round 2. The submissions that performed well in Round 2 were those which were robust to domain-shift, and because the organizers retrained submissions from scratch in Round 2, those successful submissions were also certifiably reproducible and resource-efficient. Unlike the previous iteration of the MineRL Competition, the addition of the action/observation obfuscation technique completely prevented teams from action shaping and feature engineering, leading to the emergence of a very common action-simplification technique: K-means clustering of the expert action space.

**Limitations.** In its current form, the competition is limited to the Minecraft domain. Although domain randomization and action space obfuscation prevent the introduction of some inductive biases into the participants’ algorithms, competitors still expect to be faced with the broader game mechanics of Minecraft. This domain knowledge ultimately manifests itself in the particular algorithmic structure of participants’ submissions (e.g., in the choice of hierarchical methods). A natural next step to promote more domain agnostic methods is to expand the scope of the competition to include a training step against a different
6. Related Work

Previous competitions (Juliani et al., 2019) focused on a variety of aspects of RL, such as the multi-agent setting (Gao et al., 2019; Perez-Liebana et al., 2019; Mohanty et al., 2020) or practical applications (Marot et al., 2020). In most of these competitions, the focus is not on generalization. Instead, the goal is to develop a trained agent that performs well on a given domain. Consequently, winning submissions often relied on hand-engineered features and stemmed from the use of large amounts of computational resources to optimize the submission for the specific task. Competitions that have promoted the development of general RL algorithms do not perform action or observation obfuscation and either focus on generalization across a variety of tasks with shared objects, textures, and actions (Nichol et al., 2018) or utilize the approach of having hold-out test tasks (Cartoni et al., 2020).

To our knowledge, no previous competition has explicitly encouraged the use of imitation learning alongside RL. Previous imitation learning competitions (Diodato et al., 2019) concentrate on the prediction setting (e.g., predicting a vehicle’s speed given sensor inputs), which is reflected in their evaluation metrics. However, our evaluation metrics are more reflective of the sequential decision-making setting in which our competition takes place. Although some top solutions to previous competitions leveraged imitation learning (Meisheri et al., 2019), the use of imitation learning alongside reinforcement learning was not explicitly promoted. In contrast, we encourage the use of imitation learning by providing participants with a large dataset of human demonstrations and introducing a second track to the competition where competitors must only use the dataset to train their algorithms.

7. Conclusion

We ran the 2020 MineRL Competition on Sample Efficient Reinforcement Learning Using Human Priors in order to promote the development of general, sample efficient reinforcement learning and imitation learning algorithms. We described the competition, highlighting changes to the rules and structure from the 2019 version of the competition. We summarized the performance of the submissions and contrasted this performance with the performance from last year. We described the approaches of the top seven teams from Round 2. We concluded by discussing the benefits and limitations of our approach.

Acknowledgments

We thank AIcrowd for hosting the competition evaluator and providing tireless hours of support in ensuring that competitors could submit their solutions. We especially thank Shivam Khandelwal for his help in developing the competition starter-kit and providing constant
assistance to the organizers and the participants during the competition. We would like to thank Ansii Kanervisto for his continual and detailed responses in the competition Discord. In addition, we would like to acknowledge our sponsor Preferred Networks for providing a rich set of baselines in their new framework PFRL. We thank Microsoft Research and the Artificial Intelligence Journal for their generous sponsorship of competition compute (for retraining and evaluation), of compute grants enabling the participation of underrepresented groups, and of NeurIPS registrations for competitors. Finally, we would like to acknowledge Microsoft Research and NVIDIA for providing prizes for the competition.

References


James MacQueen et al. Some methods for classification and analysis of multivariate observations. 1967.


Mark Pfeiffer, Samarth Shukla, Matteo Turchetta, Cesar Cadena, Andreas Krause, Roland Siegwart, and Juan Nieto. Reinforced imitation: Sample efficient deep reinforcement


