

NeurIPS 2020 Competition: The MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors

William H. Guss^{*†‡} Sam Devlin^{§¶} Brandon Houghton^{§‡}
Noboru Sean Kuno^{§¶} Stephanie Milani^{§†} Sharada Mohanty^{§||}
Ruslan Salakhutdinov^{§†} John Schulman^{§‡} Nicholay Topin^{§†}
Oriol Vinyals^{§**}

March 13, 2020

Overview of the competition

Though deep reinforcement learning has led to breakthroughs in many difficult domains, these successes have required an ever-increasing number of samples. As state-of-the-art reinforcement learning (RL) systems require an exponentially increasing number of samples, their development is restricted to a continually shrinking segment of the AI community. Likewise, many of these systems cannot be applied to real-world problems, where environment samples are expensive. Resolution of these limitations requires new, sample-efficient methods. To facilitate research in this direction, we propose the *MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors*.

The primary goal of the competition is to foster the development of algorithms which can efficiently leverage human demonstrations to drastically reduce the number of samples needed to solve complex, hierarchical, and sparse environments. To that end, participants will compete under a limited environment sample-complexity budget to develop systems which solve the MineRL ObtainDiamond task, a sequential decision making environment

^{*}Lead organizer: wguss@cs.cmu.edu

[†]Affiliation: Carnegie Mellon University

[‡]Affiliation: OpenAI Inc.

[§]**Equal contribution:** Organizer names are ordered alphabetically, with the exception of the lead organizer. Competitions are extremely complicated endeavors involving a huge amount of organizational overhead from the development of complicated software packages to event logistics and evaluation. It is impossible to estimate the total contributions of all involved at the onset.

[¶]Affiliation: Microsoft Research

^{||}Affiliation: AIcrowd SA

^{**}Affiliation: DeepMind Inc.

requiring long-term planning, hierarchical control, and efficient exploration methods. Participants will be provided the *MineRL-v0* dataset, a large-scale collection of over 60 million state-action pairs of human demonstrations that can be resimulated into embodied agent trajectories with arbitrary modifications to game state and visuals [14].

The competition is structured into two rounds in which competitors are provided several paired versions of the dataset and environment with different game textures and shaders. At the end of each round, competitors will submit containerized versions of their learning algorithms to the AIcrowd platform where they will then be trained from scratch on a hold-out dataset-environment pair for a total of 4-days on a pre-specified hardware platform. Each submission will then be automatically ranked according to the final performance of the trained agent.

This challenge is a follow-up to our NeurIPS 2019 MineRL competition [13], which yielded over 1000 registered participants and over 662 full submissions. The competition benchmark, RL environment, and dataset framework were downloaded over 52,000 times in 26+ countries [22]. In this second iteration, we intend on implementing new features to expand the scale and reach of the competition. In response to the feedback of the previous participants, we are introducing a second minor track focusing on solutions without access to environment interactions of any kind except during test-time. Both tracks will follow the same 2-round schedule. Additionally, in the previous year, a large portion of top submissions developed novel methods advancing inverse reinforcement learning and hierarchical imitation learning, and in the forthcoming competition we anticipate an even larger research impact. With the addition of action-space randomization and desemantization of observations and actions (see Section 1.2) in MineRL the most successful competition submissions will be highly task and domain agnostic.

Keywords

Reinforcement Learning, Imitation Learning, Sample Efficiency, Games, MineRL, Minecraft.

Competition type

Regular.

1 Competition description

1.1 Background and impact

Many of the recent, most celebrated successes of artificial intelligence (AI), such as AlphaStar [40], AlphaGo [34], OpenAI Five [3], and their derivative systems [35], utilize deep reinforcement learning 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 to achieve such performance [1]. In part, this is due to an increase in the computation required per environment-sample; however, the most significant change is the number of environment-samples required for training. For example, DQN [23], A3C [24], and Rainbow DQN [15] have been applied to ATARI 2600 games [2] and require from 44 to over 200 million frames (200 to over 900 hours) to achieve human-level performance. On more complex domains: OpenAI Five utilizes 11,000+ years of Dota 2 gameplay [27], AlphaGoZero uses 4.9 million games of self-play in Go [34], and AlphaStar uses 200 years of StarCraft II gameplay [7]. Due to the growing computational requirements, a shrinking portion of the AI community has the resources to improve these systems and reproduce 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. In these real-world domains, policy roll-outs can be costly and simulators are not yet accurate enough to yield policies robust to real-world conditions.

One well-known way to reduce the environment sample-complexity of the aforementioned methods is to leverage human priors and demonstrations of the desired behavior. Techniques utilizing trajectory examples, such as imitation learning and Bayesian reinforcement learning, have been successfully applied to older benchmarks and real-world problems where samples from the environment are costly. In many simple games with singular tasks, such as the Atari 2600 [2], OpenAI Gym [5], and TORCS environments¹, imitation learning can drastically reduce the number of environment samples needed through pretraining and hybrid RL techniques [17, 6, 28, 11]. Further, in some real-world tasks, such as robotic manipulation [9, 8] and self-driving [4], in which it is expensive to gather a large number of samples from the environment, imitation-based methods are often the only means of generating solutions using few samples. Despite their success, these techniques are still not sufficiently sample-efficient for application to many real-world domains.

Impact. To that end, the central aim of our proposed competition is the advancement and development of novel, sample-efficient methods which leverage human priors for sequential decision-making problems. Due to the competition’s design, organizational team, and support, we are confident that the competition will catalyze research towards the deployment of reinforcement learning in the real world, democratized access to AI/ML, and reproducibility. By enforcing constraints on the computation and sample budgets of the considered techniques, we believe that the methods developed during the competition will broaden participation in deep RL research by lowering the computational barrier to entry.

While computational resources inherently have a cost barrier, large-scale, open-access datasets can be widely used. To that end, we center our proposed competition around techniques which leverage the newly introduced MineRL dataset [14]. To maximize the development of domain-agnostic techniques that enable the application of deep reinforcement

¹https://github.com/ugo-nama-kun/gym_torcs



Figure 1: The top agent from the MineRL 2019 competition mining the first item in the hierarchy.

learning to sample-limited, real-world domains, such as robotics, we carefully developed a novel data-pipeline and hold-out environment evaluation scheme with AIcrowd to prevent the over-engineering of submissions to the competition task.

Crucially, the competition will stimulate a broad set of new techniques in reinforcement and imitation learning. In the previous NeurIPS 2019 iteration of the competition, competitors developed several new algorithms and approaches to tackle the challenge in spite of the difficult sample-complexity limitations [12]. Ranging from hierarchical imitation methods to novel inverse reinforcement learning techniques, the research impact of the competition was broad in scope, yielding a diverse set of solutions [22]. With the addition of new competition features and refined submission and evaluation pipelines (see Section 1.2), we anticipate this year’s competition to garner further research progress of relevance to the NeurIPS community.

Our competition will further attract a large number of diverse participants from within and outside of the NeurIPS community. Given the broad interest and participation in the previous year (attracting over 1000 registered participants with a total of 662 full submissions²), our extensive media coverage [39, 19, 32, 36], and improvements to user-experience, we expect the number of participants to grow to 1300 users and the number of successful submission to increase to over 1000 agents. To effectuate this growth, we will deliver several improvements over prior years. First, we plan to drastically simplify the submission process and provide thorough multi-media documentation to increase the conversion-rate from registration to submission. Further, we intend on providing more compelling visualizations for the competitors’ submissions, generating external interest from outside of the research community. Expanding on media coverage and outreach channels from last year, we will utilize mailing lists and social media announcements to retain the previous competi-

²<https://www.aicrowd.com/challenges/neurips-2019-minerl-competition>

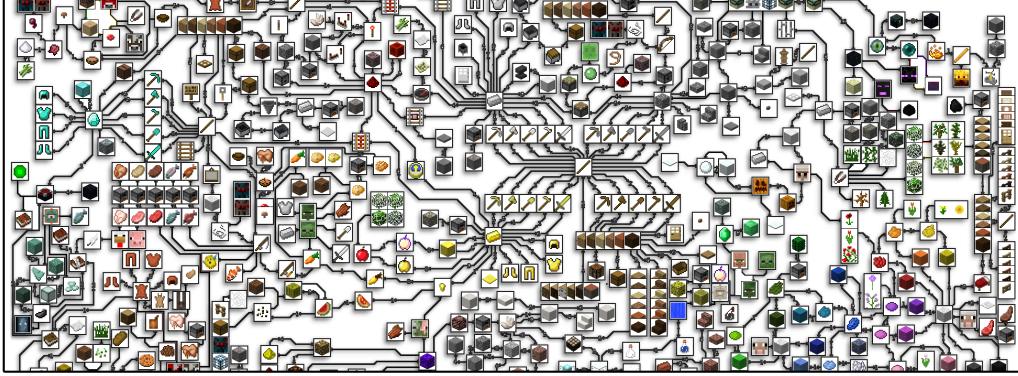


Figure 2: A subset of the Minecraft item hierarchy (totaling 371 unique items). Each node is a unique Minecraft item, block, or non-player character, and a directed edge between two nodes denotes that one is a prerequisite for another. Each item presents its own unique set of challenges, so coverage of the full hierarchy by one player takes several hundred hours.

tor pool and expand our user-base to new demographics. Moreover, the expansion of our competition to multiple tracks supporting pure imitation learning and hybridized imitation and reinforcement learning submissions will broaden the appeal of our competition as a vehicle for researching and developing new methods.

The proposed competition is ambitious, so we have taken meaningful steps to ensure its smooth execution. Specifically, we are currently securing several crucial partnerships with organizations and individuals. During the MineRL 2019 competition, our primary partner, Microsoft Research, provided significant computational resources to enable direct, fair evaluation of the participants’ training procedures. We developed a relationship with AIcrowd to provide the submission orchestration platform for our competition, as well as continued support throughout the competition to ensure that participants can easily submit their algorithms. Additionally, we partnered with Preferred Networks in the previous iteration of this competition to provide a set of standard baseline implementations, which include many state of the art reinforcement learning and imitation learning techniques. By leveraging our previous partnerships and developing new ones, we expect to largely increase the scale, success, and impact of the competition.

1.1.1 Domain Interest

Minecraft is a compelling domain for the development of reinforcement and imitation learning methods because of the unique challenges it presents: Minecraft is a 3D, first-person, open-world game centered around the gathering of resources and creation of structures and items. 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 (such as wood from trees) and constructing structures (such as shelter and storage). Since Minecraft is an embodied domain and the agent’s surroundings are varied and dynamic, it presents many of the same challenges as real-world robotics domains. Therefore, solutions created for this competition are a step toward applying these same methods to real-world problems.

Furthermore, there is existing research interest in Minecraft. With the development of Malmo [20], a simulator for Minecraft, the environment has garnered great research interest: many researchers [33, 37, 26] have leveraged Minecraft’s massive hierarchy and expressive power as a simulator to make great strides in language-grounded, interpretable multi-task option-extraction, hierarchical lifelong learning, and active perception. However, much of the existing research utilizes toy tasks in Minecraft, often restricted to 2D movement, discrete positions, or artificially confined maps unrepresentative of the intrinsic complexity that human players typically face. These restrictions reflect the difficulty of the domain, the challenge of coping with fully-embodied human state- and action-spaces, and the complexity exhibited in optimal human policies.

Our competition and the utilization of the large-scale MineRL-v0 dataset of human demonstrations will serve to catalyze research on this domain in two ways: (1) our preliminary results indicate that through imitation learning, basic reinforcement learning approaches can finally deal directly with the full, unrestricted state- and action-space of Minecraft; and (2) due to the difficult and crucial research challenges exhibited on the primary competition task, `ObtainDiamond`, we believe that the competition will bring work on the Minecraft domain to the fore of sample-efficient reinforcement learning research.

1.2 Novelty

This year’s MineRL Competition is a follow-up to the first MineRL competition held at NeurIPS 2019. We continue to encourage the development of general learning algorithms which must perform well within a *strict* computation and environment-sample budget. Based on community feedback and our retrospection, we are making the following improvements to this year’s competition:

- To further encourage competitors to develop generalizable methods, we are updating the rules on manually specified policies and pre-processing of the action space. In particular, we are improving the rule clarity and no longer allowing submissions to manually specify action choices. In the previous MineRL Competition, actions could be specified by the competitors as long as the setting did not depend on an aspect of the state.
- To ensure that competitors do not exploit the semantic meanings attached to the action labels, we are changing the action spaces used during the evaluation process. Specifically, the actions during evaluation will have a different label assignment across

all of the actions (matching for both the dataset and the environment). Previously, labels were modified during evaluation, but they were not fully shuffled.

- To further encourage the use of demonstration based methods, we are adding a second track to the competition. This track will follow the same restrictions as the original track, but competitors will not be permitted to use the environment during training. By adding this track, competitors interested in learning from demonstrations can compete without being disadvantaged compared to those who also use reinforcement learning. Additionally, this track will help quantify the performance increases possible using the addition of data.

In addition, the baseline methods have been created ahead of this year’s competition. As a result, competitors will have access to the baseline methods from the start of the competition. We expect this to increase participation over last year, when baseline methods were released after the competition began.

Our competition focuses on the application of reinforcement learning and imitation learning to a domain in Minecraft. As a result, it is related to competitions which focus on these three aspects. We briefly identify related competitions and describe the key differences between our proposed competition and the other competitions.

Reinforcement Learning. Prior to our competition series, reinforcement learning competitions have focused on the development of policies or meta-policies which perform well on extremely complex domains or generalize across a distribution of tasks [21, 25, 29]. However, the winning submissions of these competitions are often the result of massive amounts of computational resources or highly specific, hand-engineered features. In contrast, our competition directly considers the efficiency of the training procedures of learning algorithms.

We evaluate submissions solely on their ability to perform well within a *strict* computation and environment-sample budget. Moreover, we are uniquely positioned to propose such a competition due to the nature of our human demonstration dataset and environment: our dataset is constructed by directly recording the game-state as human experts play, so we are able to later make multiple renders of both the environment and data with varied lighting, geometry, textures, and game-state dynamics, thus yielding development, validation, and hold-out evaluation dataset/environment pairs. As a result, competitors are naturally prohibited from hand-engineering or warm-starting their learning algorithms and winning solely due to resource advantages.

Imitation Learning. To our knowledge, no competitions have explicitly focused on the use of imitation learning alongside reinforcement learning. This is in large part due to a lack of large-scale, publicly available datasets of human or expert demonstrations. Our competition is the first to explicitly involve and encourage the use of imitation learning

to solve the given task, and in that capacity, we release the largest-ever dataset of human demonstrations on an embodied domain. The large number of trajectories and rich demonstration-performance annotations enable the application of many standard imitation learning techniques and encourage further development of new ones that use hierarchical labels, varying agent performance levels, and auxiliary state information.

Minecraft. A few competitions have previously used Minecraft due to its expressive power as a domain. The first one was The Malmö Collaborative AI Challenge³, in which agents worked in pairs to solve a collaborative task in a decentralized manner. Later, C. Salge et al. [30] organized the Generative Design in Minecraft (GDMC): Settlement Generation Competition, in which participants were asked to implement methods that would procedurally build complete cities in any given, unknown landscape. These two contests highlight the versatility of this framework as a benchmark for different AI tasks.

In 2018, Perez-Liebana et al. [29] organized the Multi-Agent Reinforcement Learning in Malmö (MARLÖ) competition. This competition pitted groups of agents to compete against each other in three different games. Each of the games was parameterizable to prevent the agents from overfitting to specific visuals and layouts. The objective of the competition was to build an agent that would learn, in a cooperative or competitive multi-agent task, to play the games in the presence of other agents. The MARLÖ competition successfully attracted a large number of entries from both existing research institutions and the general public, indicating a broad level of accessibility and excitement for the Minecraft domain within and outside of the existing research community.

In comparison with previous contests, the MineRL series of competitions tackles one main task and provides a massive number of hierarchical subtasks and demonstrations (see Section 1.3). The main task and its subtasks are not trivial; however, agent progress can be easily measured, which allows for a clear comparison between submitted methods. Further, the target of the competition series is to promote research on *efficient learning*, focusing directly on the sample- and computational-efficiency of the submitted algorithms.

1.3 Data

For this competition, we utilize two main components: a set of sequential decision making environments in Minecraft and a corresponding public large-scale dataset of human demonstrations. Through an online server which replicates these environments, we continue to engage the Minecraft community to add additional demonstrations to this dataset.

1.3.1 Environment

We define *one primary competition environment*, ObtainDiamond, and six other auxiliary environments that encompass a significant portion of human Minecraft play. We select

³<https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge>

these environment domains to highlight many of the hardest challenges in reinforcement learning, such as sparse rewards, long reward horizons, and efficient hierarchical planning.

Primary Environment. The main task of the competition is solving the `ObtainDiamond` environment. In this environment, the agent begins in a random starting location without any items, and is tasked with obtaining a diamond. The agent receives a high reward for obtaining a diamond as well as smaller, auxiliary rewards for obtaining prerequisite items. Episodes end due to the agent (a) dying, (b) successfully obtaining a diamond, or (c) reaching the maximum step count of 18000 frames (15 minutes).

The `ObtainDiamond` environment is a difficult environment; diamonds only exist in a small portion of the world and are 2-10 times rarer than other ores in Minecraft. Furthermore, obtaining a diamond requires many prerequisite items. Therefore, it is practically impossible for an agent to obtain a diamond via naive random exploration.

Auxiliary Environments. We provide six auxiliary environments (in four families), which we believe will be useful for solving `ObtainDiamond` (see Section 1.3.4):

1. **Navigate:** In this environment, the agent must move to a goal location, which represents a basic primitive used in many tasks in Minecraft. In addition to standard observations, the agent has access to a “compass” observation, which points to a set location, 64 meters from the start location. The agent is given a sparse reward (+100 upon reaching the goal, at which point the episode terminates). We also support a dense, reward-shaped version of Navigate, in which the agent receives reward every tick corresponding to the change in distance between the agent and the goal.
2. **Treechop:** In this environment, the agent must collect wood, a key resource in Minecraft and the first prerequisite item for diamonds. The agent begins in a forest biome (near many trees) with an iron axe for cutting trees. The agent is given +1 reward for obtaining each unit of wood, and the episode terminates once the agent obtains 64 units or the step limit is reached.

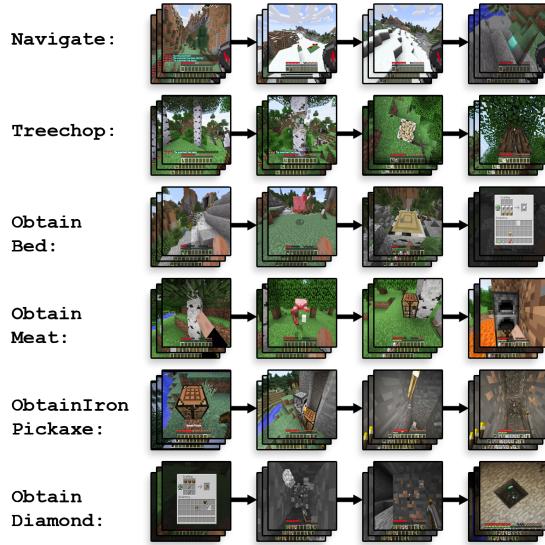


Figure 3: Images of various stages of six of seven total environments.

3. **Obtain<Item>**: We include three additional obtain environments, similar to that of `ObtainDiamond`, but with different goal items to obtain. They are:
 - (a) **CookedMeat**: cooked meat of a (cow, chicken, sheep, or pig), which is necessary for survival in Minecraft. In this environment, the agent is given a specific kind of meat to obtain.
 - (b) **Bed**: made out of dye, wool, and wood, an item that is also vital to Minecraft survival. In this environment, the agent is given a specific color of bed to create.
 - (c) **IronPickaxe**: is a final prerequisite item in obtaining a diamond. It is significantly easier to solve than `ObtainDiamond`: iron is 20 times more common in the Minecraft world than diamonds, and this environment is typically solved by humans in less than 10 minutes.
4. **Survival**: This environment is the standard, open-ended game mode used by most human players when playing the game casually. There is no specified reward function, but data from this environment can be used to help train agents in more structured tasks, such as `ObtainDiamond`.

1.3.2 Dataset

The MineRL-v0 dataset consists of over 60 million state-action-(reward) tuples of recorded human demonstrations over the seven environments mentioned above [14]. In addition, we are actively working with the community to record additional human demonstrations. Trajectories are contiguously sampled every Minecraft game tick (at 20 game ticks per second). Each state is comprised of an RGB video frame of the player’s point-of-view and a comprehensive set of features from the game-state at that tick: player inventory, item collection events, distances to objectives, player attributes (health, level, achievements), and details about the current GUI the player has open. The action recorded at each tick consists of: all the keyboard presses, the change in view pitch and yaw (mouse movements), player GUI interactions, and agglomerative actions such as item crafting.

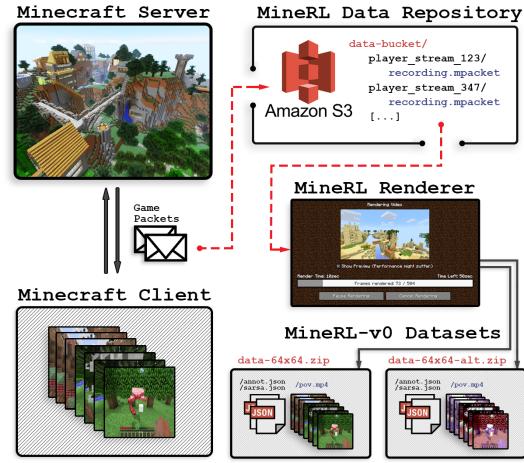


Figure 4: A diagram of the MineRL data collection platform. Our system renders demonstrations from packet-level data, so we can easily rerender our data with different parameters.

), player GUI interactions, and agglomerative actions such as item crafting.

Human trajectories are accompanied by a large set of automatically generated annotations. For all of the environments, we include metrics which indicate the quality of the demonstration, such as timestamped rewards, number of no-ops, number of deaths, and total score. Additionally, trajectory meta-data includes timestamped markers for hierarchical labelings; e.g. when a house-like structure is built or certain objectives such as chopping down a tree are met. Data is made available both in the competition materials as well as through a standalone website⁴.

1.3.3 Data Collection

In the previous MineRL competition, we used our novel platform for the collection of player trajectories in Minecraft, enabling the construction of the MineRL-v0 dataset. In this second iteration of the competition, we will continue to utilize the platform with the hope of drastically expanding the existing dataset. As shown in Figure 4, our platform consists of (1) *a public game server and website*, where we obtain permission to record trajectories of Minecraft players in natural gameplay; (2) *a custom Minecraft client plugin*, which records all packet level communication between the client and the server, so we can re-simulate and re-render human demonstrations with modifications to the game state and graphics; and (3) *a data processing pipeline*, which enables us to produce automatically annotated datasets of task demonstrations.

Data Acquisition. Minecraft players find the MineRL server on standard Minecraft server lists. Players first use our webpage to provide IRB⁵ consent to have their gameplay anonymously recorded. Then, they download a plugin for their Minecraft client, which records and streams users’ client-server game packets to the MineRL data repository. When playing on our server, users select an environment to solve and receive in-game currency proportional to the amount of reward obtained. For the `Survival` environment (where there is no known reward function), players receive rewards only for duration of gameplay, so as not to impose an artificial reward function.

Data Pipeline. Our data pipeline allows us to resimulate recorded trajectories into several algorithmically consumable formats. The pipeline serves as an extension to the core Minecraft game code and synchronously sends each recorded packet from the MineRL data repository to a Minecraft client using our custom API for automatic annotation and game-state modification. This API allows us to add annotations based on any aspect of the game state accessible from existing Minecraft simulators. Notably, it allows us to rerender the same data with different textures, shaders, and lighting-conditions which we use to create test and validation environment-dataset pairs for this competition.

⁴<http://minerl.io>

⁵The data collection study was approved by Carnegie Mellon University’s institutional review board as STUDY2018_00000364.

1.3.4 Data Usefulness

Human Performance. A majority of the human demonstrations in the dataset fall within the range of expert level play. Figure 5 shows the distribution over trajectory length for each environment. The red region in each histogram denotes the range of times which correspond to play at an expert level, computed as the average time required for task completion by players with at least five years of Minecraft experience. The large number of expert samples and rich labelings of demonstration performance enable application of many standard imitation learning techniques which assume optimality of the base policy. In addition, the beginner and intermediate level trajectories allow for the further development of techniques which leverage imperfect demonstrations.

Hierarchy. As shown in Figure 2, Minecraft is deeply hierarchical, and the MineRL data collection platform is designed to capture these hierarchies both explicitly and implicitly. Due to the sub-task labelings provided in MineRL-v0, we can inspect and quantify the extent to which these environments overlap. Figure 6 shows precedence frequency graphs constructed from MineRL trajectories on the `ObtainDiamond`, `ObtainCookedMeat`, and `ObtainIronPickaxe` tasks. In order to complete the `ObtainDiamond` task, an agent must complete the sub-goals of obtaining wood and stone, as well as constructing crafting tables and furnaces. These subtasks also appear in `ObtainIronPickaxe` and `ObtainCookedMeat`.

There is even greater overlap between

`ObtainDiamond` and `ObtainIronPickaxe`: most of the item hierarchy for `ObtainDiamond` consists of the hierarchy for `ObtainIronPickaxe`.

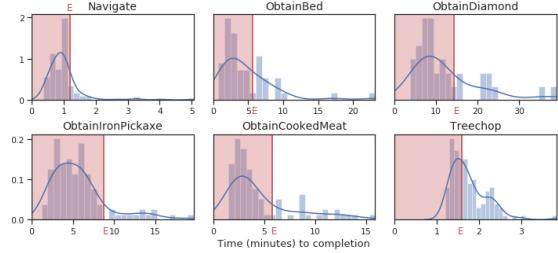


Figure 5: Normalized histograms of the lengths of human demonstration on various MineRL tasks. The red E denotes the upper threshold for expert play on each task.

standard imitation learning techniques which assume optimality of the base policy. In addition, the beginner and intermediate level trajectories allow for the further development of techniques which leverage imperfect demonstrations.

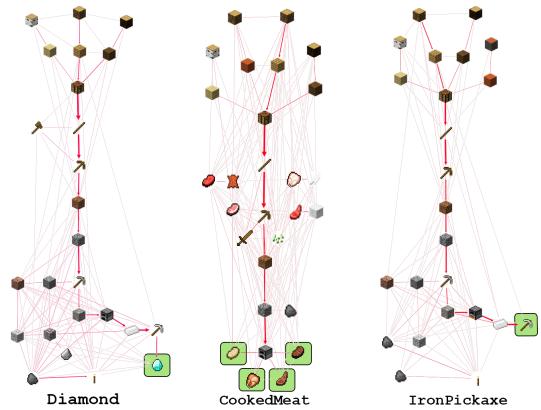


Figure 6: Item precedence frequency graphs for `ObtainDiamond` (left), `ObtainCookedMeat` (middle), and `ObtainIronPickaxe` (right). The thickness of each line indicates the number of times a player collected item A then subsequently item B .

```

import malmo_env

env = malmo_env.make("Obtain-Diamond-v0")
observation = env.reset()
done = False
while not done:
    env.render()
    action = env.action_space.sample() # your agent here (this takes random actions)
    observation, reward, done, info = env.step(action)

```

Figure 7: Example code for running a single episode of a random agent in ObtainDiamond.

```

import minerl
dat = minerl.make("obtain_diamond")
while dat.has_trajectory():
    trajectory = dat.sample_trajectory()
    for state, action in trajectory:
        print(state, action) # your learning procedure here

```

Figure 8: Utilizing individual trajectories of the MineRLdataset.

Interface Interacting with the environment and our data is as simple as a few lines of code. Participants will be provided with an OpenAI Gym [5] wrapper for the environment and a simple interface for loading demonstrations from the MineRL-v0 dataset as illustrated in Figures 7, 8, and 9. Our data will be released in the form of Numpy .npz files composed of state-action-reward tuples in vector form, and can be found along with accompanying documentation on the competition website.

```

import minerl
dat = minerl.make("obtain_diamond")
dat.batch_size = 32
dat.seq_len = 50
dat = dat.filter(lambda meta: meta['episode_length'] < 15000)
while dat.has_batch():
    batch = minerl.sample_batch() # returns 32 samples of length 50

```

Figure 9: Using the MineRLwrapper to filter demonstrations based on metadata

1.4 Tasks and application scenarios

1.4.1 Task

The primary task of the competition is solving the `ObtainDiamond` environment. As previously described (see Section 1.3), agents begin at a random position on a randomly generated Minecraft survival map with no items in their inventory. The task consists of controlling an embodied agent to obtain a single diamond. This task can only be accomplished by navigating the complex item hierarchy of Minecraft. The learning algorithm will have direct access to a 64x64 pixel point-of-view observation from the perspective of the embodied Minecraft agent, as well as a set of discrete observations of the agent’s inventory for every item required for obtaining a diamond (see Figure 6). The action space of the agent 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. The agent is rewarded for completing the full task. Due to the difficulty of the task, the agent is also rewarded for reaching a set of milestones of increasing difficulty that form a set of prerequisites for the full task (see Section 1.5).

The competition task embodies two crucial challenges in reinforcement learning: sparse rewards and long time horizons. The sparsity of the posed task (in both its time structure and long time horizon) necessitates the use of efficient exploration techniques, human priors for policy bootstrapping, or reward shaping via inverse reinforcement learning techniques. Although this task is challenging, preliminary results indicate the potential of existing and new methods utilizing human demonstrations to make progress in solving it (see Section 1.6).

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. In this sense, the application scenarios of the competition are those which stand to benefit from the development of such algorithms; to that end, we believe that this competition is a step towards democratizing access to deep reinforcement learning based techniques and enabling their application to real-world problems.

Previous Years Task Stability in metrics across years is crucial for tracking and assessing long-term impact and progress. In the MineRL 2019 competition, no team obtained a diamond, but teams came close obtaining an item penultimate to this final item. For this reason, we keep the same task from last year.

1.5 Metrics

Following training, participants will be evaluated on the average score of their model over 500 episodes. Scores are computed as the sum of the milestone rewards achieved by the agent in a given episode as outlined in Table 1. A milestone is reached when an agent obtains the first instance of the specified item. Ties are broken by the number of episodes required to achieve the last milestone. An automatic evaluation script will be included with starter code. For official evaluation and validation, a fixed map seed will be selected for each episode. These seeds will not be available to participants during the competition.

1.6 Baselines, code, and material provided

Preliminary Baselines We present preliminary results showing the usefulness of the data for improving sample efficiency and overall performance. We compare algorithms by the highest average reward obtained over a 100-episode window during training. We also report the performance of random policies and 50th percentile human performance. The results are summarized in Table 2.

In the presented comparison, the DQN is an implementation of Double Dueling DQN [38] and Behavioral Cloning is a supervised learning method trained on expert trajectories. PreDQN denotes a version of DQN pre-trained on the MineRL-v0 data: specifically, PreDQN is trained by performing Bellman updates on minibatches drawn from expert trajectories with accompanying reward labels. Before training, we initialize the replay buffer with expert demonstrations.

In all environments, the learned agents perform significantly worse than humans. `Treechop` exhibits the largest difference: on average, humans achieve a score of 64, but reinforcement agents achieve scores of less than 4. These results suggest that our environments are quite challenging, especially given that the `Obtain<Item>` environments build upon the `Treechop` environment by requiring the completion of several additional sub-goals. We hypothesize that a large source of difficulty stems from the environment’s inherent long-horizon credit assignment problems. For example, it is hard for agents to learn to navigate through water because it takes many transitions before the agent dies by drowning.

In light of these difficulties, our data is useful in improving performance and sample

Milestone	Reward	Milestone	Reward
log	1	furnace	32
planks	2	stone_pickaxe	32
stick	4	iron_ore	64
crafting_table	4	iron_ingot	128
wooden_pickaxe	8	iron_pickaxe	256
stone	16	diamond	1024

Table 1: Rewards for sub-goals and main goal (diamond) for `Obtain Diamond`.

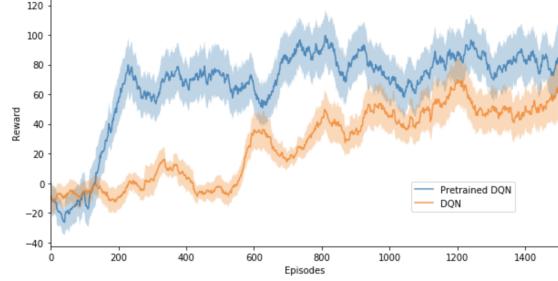


Figure 10: Performance graphs over time with DQN and PreDQN on `Navigate(Dense)`

	Treechop	Navigate (S)	Navigate (D)
DQN (Minh et al., 2015[23])	3.73 ± 0.61	0.00 ± 0.00	55.59 ± 11.38
A2C (Minh et al. 2016[24])	2.61 ± 0.50	0.00 ± 0.00	-0.97 ± 3.23
Behavioral Cloning	43.9 ± 31.46	4.23 ± 4.15	5.57 ± 6.00
PreDQN	4.16 ± 0.82	6.00 ± 4.65	94.96 ± 13.42
Human	64.00 ± 0.00	100.00 ± 0.00	164.00 ± 0.00
Random	3.81 ± 0.57	1.00 ± 1.95	-4.37 ± 5.10

Table 2: Results in Treechop, Navigate (S)parse, and Navigate (D)ense, over the best 100 contiguous episodes. \pm denotes standard deviation. Note: humans achieve the maximum score for all environments shown.

efficiency: in all environments, methods that leverage human data perform better. As seen in Figure 10, the expert demonstrations were able to achieve higher reward per episode and attain high performance using fewer samples. Expert demonstrations are particularly helpful in environments where random exploration is unlikely to yield any reward, like Navigate (Sparse). These preliminary results indicate that human demonstrations will be crucial in solving the main competition environment.

2019 Baselines For the 2019 MineRL Competition, Preferred Networks⁶ provided extensive baselines⁷, including behavioral cloning, deep Q-learning from demonstrations (DQfD) [17], Rainbow [16], generative adversarial inverse RL (GAIL) [18], and proximal policy optimization (PPO) [31]. These baselines are implemented using ChainerRL [10], and MineRL 2019 participants found them to be incredibly helpful for developing their algorithms. These baselines⁸ will be available to participants to freely use in this iteration of the competition.

2020 Baselines We are in discussion with potential partners to implement and provide participants with new and improved baselines for this year’s competition. We will also be providing code from top participants in the MineRL 2019 competition. However, these solutions will not conform to the rules of this year. We hope competitors will be able to take inspiration from these methods.

Starting Code and Documentation. We will release an open-source Github repository with starting code including the baselines mentioned above, an OpenAI Gym interface for the Minecraft simulator, and a data-loader to accompany the data. Additionally, we will

⁶<https://preferred.jp/en/>

⁷<https://github.com/minerllabs/baselines>

⁸Preferred Network’s writeup of their experiments using their baselines and the MineRL environments can be found here.

release a public Docker container for ease of use. We will also provide participants with the code for the solutions from last year’s top participants.

1.7 Tutorial and documentation

We have a competition page that contains instructions, documentation⁹, and updates to the competition. For this competition, we plan to include a step-by-step demonstration showing participants how to submit their learning procedures. Although top participants in MineRL 2019 stated that they found the documentation to be helpful, we plan to extend the documentation in the hopes that even more people can participate this year.

2 Organizational aspects

2.1 Protocol

2.1.1 Submission Protocol

The evaluation of the submissions will be managed by AICrowd, an open-source platform for organizing machine learning competitions. Throughout the competition, participants will work on their code bases as git repositories¹⁰. Participants must package their intended runtime in their repositories to ensure that the AICrowd evaluators can automatically build relevant Docker images from their repositories and orchestrate them as needed. This approach also ensures that all successfully-evaluated, user-submitted code is both versioned across time and completely reproducible.

Software Runtime Packaging. Packaging and specification of the software runtime is among the most time consuming (and frustrating) tasks for many participants. To simplify this step, we will support numerous approaches to easily package the software runtime with the help of `aicrowd-repo2docker`¹¹. The `aicrowd-repo2docker` is a tool which lets participants specify their runtime using Anaconda environment exports, `requirements.txt`, or a traditional Dockerfile. This significantly decreases the barrier to entry for less technically-inclined participants by transforming an irritating debug cycle to a deterministic one-liner that performs the work behind the scenes.

Submission Mechanism. Participants will collaborate on their git repository throughout the competition. Whenever they are ready to make a submission, they will create and push a *git tag*, which triggers the evaluation pipeline.

⁹<http://minerl.io/docs/>

¹⁰<https://gitlab.aicrowd.com>

¹¹<https://pypi.org/project/aicrowd-repo2docker/>

Orchestration of the Submissions. The ability to reliably orchestrate user submissions over large periods of time is a key determining feature of the success of the proposed competition. We will use the evaluators of AICrowd, which use custom Kubernetes clusters to orchestrate the submissions against pre-agreed resource usage constraints. The same setup has previously been successfully used in numerous other machine learning competitions, such as NeurIPS 2017: Learning to Run Challenge, NeurIPS 2018: AI for Prosthetics Challenge, NeurIPS 2018: Adversarial Vision Challenge, and the 2018 MarLO challenge. The evaluation setup allows for evaluations of arbitrarily long time-periods, and also can privately provide feedback about the current state of the evaluation to the respective participants.

2.1.2 General Competition Structure

Round 1: General Entry. In this round, participants will register on the competition website, and receive the following materials:

- Starter code for running the MineRL environments for the competition task.
- Basic baseline implementations provided by Preferred Networks, the competition organizers, and the top teams from the MineRL 2019 competition (see Section 1.6).
- Two different renders of the human demonstration dataset (one for methods development, the other for validation) with modified textures, lighting conditions, and minor game state changes.
- The Docker Images and quick-start template that the competition organizers will use to validate the training performance of the competitor’s models.

Competitors may submit solutions to two tracks. The main track will provide access to both the simulator and paired demonstrations during training, while the alternate, demonstration only track, will only provide agents with the demonstration dataset during training. Both tracks will be evaluated by measuring average performance over 100 episodes on the `ObtainDiamond` task. Competitors may submit to both tracks.

When satisfied with their models, participants will follow the submission protocols (described in Section 2.1.1) to submit their code for evaluation, specifying either the main track or alternate track. The automated evaluation setup will evaluate the submissions against the validation environment, to compute and report the metrics (described in Section 1.5) to the respective public leaderboard on the AICrowd website. Because the full training phase is quite resource intensive, it is not be possible to run the training for all the submissions in this round; however, the evaluator will ensure that the submitted code includes the relevant subroutines for the training of the models by running a short integration test on the training code before doing the actual evaluation on the validation environment.

Once Round 1 is complete, the organizers will examine the code repositories of the top submissions from each track to ensure compliance with the competition rules. For the main track, the top 15 verified teams will be invited to the second round. For the alternate demonstration-only track 5 teams will move on to round 2. To verify the top submissions comply with the competition rules, they will be automatically trained on the validation dataset and environment by the competition orchestration platform. The code repositories associated with the corresponding submissions will be forked, and scrubbed of any files larger than 15MB to ensure that participants are not using any pretrained models in the subsequent round. The resulting trained models will then be evaluated over several hundred episodes. Their performance will be compared with the submission’s final model performance during Round 1 to ensure that no warm-starting or adversarial modifications of the evaluation harness was made. In the case of the alternate track, we additionally verify that no environment interactions were used in the development of the model. For those submissions whose end-of-round and organizer-ran performance distributions disagree, the offending teams will be contacted for appeal. Unless a successful appeal is made, the organizers will remove those submissions from the competition and then evaluate additional submissions until each track is at capacity: 15 teams for the main track, and 5 teams for the alternate track. Teams may qualify for the second round in both tracks, therefore fewer than 20 teams may qualify for round 2 among the two tracks.

Round 2: Finals. In this round, the top performing teams will continue to develop their algorithms. Their work will be evaluated against a confidential, held-out test environment and test dataset, to which they will not have access. This environment includes perturbations to both the action-space as well as the observation space.

Specifically, participants in each track will be able to make a submission to that track (as described in Section 2.1.1) twice during Round 2. The automated evaluator will execute their algorithms on the test dataset and simulator, and report their score and metrics back to the participants. This is done to prevent competitors from over-fitting to the training and validation datasets/simulators.

Again all submitted code repositories will be scrubbed to remove any files larger than 15MB to ensure participants are not including any model weights pre-trained on the previously released training dataset. While the container running the submitted code will not have external network access, relevant exceptions are added to ensure participants can download and use popular frameworks like PyTorch¹² and Tensorflow¹³. Participants can request to add network exceptions for any other publicly available resource, which will be validated by AIcrowd on a case by case basis.

Further, participants will submit a written report of their technical approach to the problem; this report will be used to bolster the impact of this competition on sample-

¹²<https://pytorch.org>

¹³<http://tensorflow.org>

efficient reinforcement learning research. They will also be encouraged to submit their papers to relevant workshops at NeurIPS in order to increase interest in their work.

At the end of the second period, the competition organizers will execute a final run of the participants’ algorithms and the winners will be selected for each of the competition tracks.

User Submitted Code. At the end of the competition, all the participants will be provided a time window of 3 weeks to appeal the mandatory open-sourcing policy and categorically object if they do not want their code repositories associated with this competition to be open sourced. Such appeals will be handled by the competition organizers, but competitors are typically prohibited from participating in the competition if they are not willing to release their submissions publicly for reproducibility. By default, all the associated code repositories will be made public and available¹⁴ after the 3 week window at the end of the competition.

NeurIPS Workshop. After winners have been selected, there will be a public NeurIPS workshop to exhibit the technical approaches developed during the competition. We plan to invite teams from Round 2 to attend and present their results at the workshop. Further details of this workshop are to be determined.

2.2 Rules

The aim of the competition is to develop sample-efficient training algorithms. Therefore, we discourage the use of environment-specific, hand-engineered features because they do not demonstrate fundamental algorithmic improvements. The following rules attempt to capture the spirit of the competition and any submissions found to be violating the rules may be deemed ineligible for participation by the organizers.

- The submission must train a machine learning model. A manually specified policy may not be used as a component of this model. Meta-controllers and otherwise must be dependent on learned parameters and may not be manually codified in source code or otherwise by participants.
- Submissions may re-use open-source code with proper attribution. At the end of the competition, submissions need be open-sourced to enable reproducibility.
- Participants are limited to the provided dataset, no additional resources in source or available over the internet may be used.
- In Round 1, participants must submit their code along with self-reported scores. The submitted code must finish training within two days on the provided platform

¹⁴<https://gitlab.aicrowd.com>

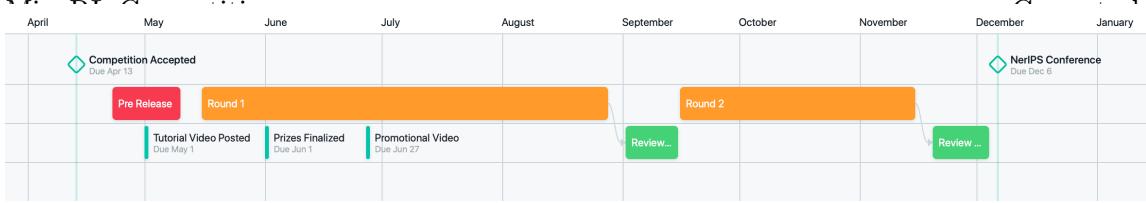


Figure 11: Proposed timeline for competition.

and attain a final performance not significantly less than the self-reported score. This training must be “from scratch” (i.e., no information may be carried over from previous training through saved model weights or otherwise). Submissions which fail to train or which do not attain the self-reported score are not eligible for the next round.

- Agents will be evaluated using unique seeds per episode. These seeds will not be available to participants until the competition has completed.
- During the evaluation of the submitted code, the individual containers will not have access to any external network to avoid any information leak.
- Hard-coded agents are not allowed, where hard-coding includes but is not limited to: using pre-defined logic to influence game behavior; stacking a pre-defined set of actions to achieve any agent behavior (e.g. crafting a pickaxe or mining a tree and then placing a crafting table) with the exception of frame-skipping; manually using any part of the state to engage or disengage any action, meta-action, or sub-policy; and other forms of manual codification of mechanisms contributing to the agent behavior which utilize priors specific to the task(s).
- All non-pixel-based states and actions will be fed through a hold-out embedding provided by the organizers during the development, evaluation, and test phases of various rounds. Such embeddings will be isomorphic but directly obfuscate and desemantize non-pixel-based state and action-space components. Any attempt to bypass these desemantizing embeddings will constitute a violation of the rules.

Cheating. The competition is designed to prevent rule breaking and to discourage submissions that circumvent the competition goals. Submissions will be tested on variants of the environment/data with different textures and lighting, discouraging the any priors that are not trained from scratch. Inherent stochasticity in the environment, such as different world and spawn locations, as well as the desemantization and isomorphic embedding of state and action-space components directly discourage the use of hard-coded policies. Furthermore, we will use automatic evaluation scripts to verify the participants’ submitted scores in the first round and perform a manual code review of the finalists of each round in the competition. We highlight that the evaluation dataset/environment pair on which participants will be evaluated is *completely inaccessible* to competitors, and measures are taken to prevent information leak.

2.3 Schedule and readiness

2.3.1 Schedule

Given the difficulty of the problem posed, ample time shall be given to allow participants to fully realize their solutions. Our proposed timeline gives competitors over 120 days to prepare, evaluate, and receive feedback on their solutions before the end of the first round.

April 13 **Competition Accepted.**

May **Pre-Release:** Submission framework finalized.

May **First Round Begins:** Participants invited to download starting materials and baselines and to begin developing their submission.

September **End of First Round:** Submissions close. Models evaluated by organizers and partners.

September **First Round Results Posted:** Official results posted notifying finalists.

September **Final Round Begins:** Finalists invited to submit their models against the held out validation texture pack.

November **End of Final Round:** Submissions close. Organizers train finalists latest submission for evaluation.

November **Final Results Posted:** Official results of model training and evaluation posted.

December 6 **NeurIPS 2020:** Winning teams invited to the conference to present their results. Awards given at conference.

2.3.2 Readiness.

At the time of writing this proposal the following key milestones are complete:

- The dataset is fully collected, cleaned, and automatically annotated;
- The competition environments have been finalized and implemented;
- The advisory committee is fully established;
- The partnership with AIcrowd has been confirmed, and we are in discussion with last year’s sponsors;
- A specific plan for attracting underrepresented groups is finalized;
- The competition infrastructure has been developed, including the submission harness.

If accepted to the NeurIPS competition track, there are no major roadblocks preventing the execution of the competition.

2.4 Competition promotion

Partnership with Affinity Groups We plan to partner with a number of affinity groups to promote the participation of groups traditionally underrepresented at NeurIPS in our competition. We plan to reach out to organizers of Women in Machine Learning (WiML)¹⁵, LatinX in AI (LXAI)¹⁶, Black in AI (BAI)¹⁷, and Queer in AI¹⁸. We will also reach out to organizations, such as Deep Learning Indaba¹⁹ and Data Science Africa²⁰, to work with them to determine how to increase participation of individuals often underrepresented in North American conferences and competitions. Specifically, we plan to form a selection committee for the Inclusion@NeurIPS scholarships consisting of some of our organizers and members from those groups. We also plan to encourage competition participants to submit write-ups of their solutions to relevant affinity group workshops at NeurIPS.

Promotion through General Mailing Lists To promote participation in the competition, we plan to distribute the call to general technical mailing lists, such as Robotics Worldwide and Machine Learning News, company mailing lists, such as DeepMind’s internal mailing list, and institutional mailing lists. We plan to promote participation of underrepresented groups in the competition by distributing the call to affinity group mailing lists, including, but not limited to Women in Machine Learning, LatinX in AI, Black in AI, and Queer in AI. Furthermore, we plan to reach out to researchers and/or lab directors who are members of underrepresented groups, such as those employed at historically black or all-female universities and colleges, to encourage their participation in the competition. By contacting these researchers, we will be able to promote the competition to individuals who are not on any of the aforementioned mailing lists, but are still members of underrepresented groups.

Media Coverage To increase general interest and excitement surrounding the competition, we will reach out to the media coordinator at Carnegie Mellon University. By doing so, our competition will be promoted by popular online magazines and websites, such as Wired. We will also post about the competition on relevant popular subreddits, such as `r/machinelearning` and `/r/datascience`, and promote it through social media. We will utilize our industry and academic partners to post on their various social media platforms, such as the Carnegie Mellon University Twitter and the Microsoft Facebook page.

The previous iteration of the MineRL competition was featured by several notable news outlets including Nature News [19], BBC [32], The Verge [39], and Synced [36]. This

¹⁵<https://wimlworkshop.org/>

¹⁶<https://www.latinxinai.org/>

¹⁷<https://blackinai.github.io/>

¹⁸<https://sites.google.com/view/queer-in-ai/>

¹⁹<http://www.deeplearningindaba.com/>

²⁰<http://www.datascienceafrica.org/>

widespread publication and coverage of the competition led to a drastic influx of new users and spectators from outside of the NeurIPS community. We intend on further leveraging these media connections to increase the reach of our call for competitors.

3 Resources

3.1 Organizing team

3.1.1 Organizers

William H. Guss. William Guss is a research scientist at OpenAI and Ph.D. student in the Machine Learning Department at CMU. William co-created the MineRL dataset and lead the MineRL competition at NeurIPS 2019. He is advised by Dr. Ruslan Salakhutdinov and his research spans sample-efficient reinforcement learning and deep learning theory. William completed his bachelors in Pure Mathematics at UC Berkeley where he was awarded the Regents' and Chancellor's Scholarship, the highest honor awarded to incoming undergraduates. During his time at Berkeley, William received the Amazon Alexa Prize Grant for the development of conversational AI and co-founded Machine Learning at Berkeley. William is from Salt Lake City, Utah and grew up in an economically impacted, low-income neighborhood without basic access to computational resources. As a result, William is committed to working towards developing research and initiatives which promote socioeconomically-equal access to AI/ML systems and their development.

Sam Devlin. Sam Devlin is a Senior Researcher in the Game Intelligence and Reinforcement Learning research groups at Microsoft Research, Cambridge. Previously, he received an MEng degree in Computer Systems and Software Engineering from the University of York in 2009 including a year working with the human factors team at BAE Systems. In 2013, he completed his PhD on multi-agent reinforcement learning at the University of York and visited Oregon State University funded by a Santander International Connections Award.

Brandon Houghton. Brandon Houghton is a Machine Learning Engineer at OpenAI and co-creator of the MineRL dataset. Graduating from the School of Computer Science at Carnegie Mellon University, Brandon’s work focuses on developing techniques to enable agents to interact with the real world through virtual sandbox worlds such as Minecraft. He has worked on many machine learning projects, such as discovering model invariants in physical systems as well as learning lane boundaries for autonomous driving.

Noboru Sean Kuno. Noboru Sean Kuno is a Senior Research Program Manager at Microsoft Research in Redmond, USA. He is a member of Artificial Intelligence Engaged team of Microsoft Research Outreach. He leads the design, launch and development of

research programs for AI projects such as Project Malmo, working in partnership with research communities and universities worldwide.

Stephanie Milani. Stephanie Milani is a Ph.D. student in the Machine Learning Department at Carnegie Mellon University. She is advised by Dr. Fei Fang and her research interests include sequential decision-making problems, with an emphasis on reinforcement learning. She completed her B.S. in Computer Science and her B.A. in Psychology at the University of Maryland, Baltimore County in 2019. Since 2016, she has worked to increase the participation of underrepresented groups in CS and AI by developing curriculum at the local and state level, co-founding a mentoring and tutoring program, and organizing outreach events to introduce middle and high school students to CS. She has been nationally recognized for her outreach efforts in CS education through a Newman Civic Fellowship.

Sharada Mohanty. Sharada Mohanty is the CEO and Co-founder of AIcrowd, an open-source platform encouraging reproducible artificial intelligence research. He was the co-organizer of many large-scale machine learning competitions, such as NeurIPS 2017: Learning to Run Challenge, NeurIPS 2018: AI for Prosthetics Challenge, NeurIPS 2018: Adversarial Vision Challenge, NeurIPS 2019 : MineRL Competition, NeurIPS 2019: Disentanglement Challenge, NeurIPS 2019: REAL Robots Challenge. During his Ph.D. at EPFL, he worked on numerous problems at the intersection of AI and health, with a strong interest in reinforcement learning. In his current role, he focuses on building better engineering tools for AI researchers and making research in AI accessible to a larger community of engineers.

Ruslan Salakhutdinov. Ruslan Salakhutdinov received his Ph.D. in machine learning (computer science) from the University of Toronto in 2009. After spending two post-doctoral years at the Massachusetts Institute of Technology Artificial Intelligence Lab, he joined the University of Toronto as an Assistant Professor in the Department of Computer Science and Department of Statistics. In February of 2016, he joined the Machine Learning Department at Carnegie Mellon University as an Associate Professor. Ruslan’s primary interests lie in deep learning, machine learning, and large-scale optimization. His main research goal is to understand the computational and statistical principles required for discovering structure in large amounts of data. He is an action editor of the Journal of Machine Learning Research and served on the senior programme committee of several learning conferences including NeurIPS and ICML. He is an Alfred P. Sloan Research Fellow, Microsoft Research Faculty Fellow, Canada Research Chair in Statistical Machine Learning, a recipient of the Early Researcher Award, Connaught New Researcher Award, Google Faculty Award, Nvidia’s Pioneers of AI award, and is a Senior Fellow of the Canadian Institute for Advanced Research.

John Schulman. John Schulman is a researcher and founding member of OpenAI, where he leads the reinforcement learning team. He received a PhD from UC Berkeley in 2016, advised by Pieter Abbeel. He was named one of MIT Tech Review’s 35 Innovators Under 35 in 2016.

Nicholay Topin. Nicholay Topin is a Machine Learning Ph.D. student advised by Dr. Manuela Veloso at Carnegie Mellon University. His current research focus is explainable deep reinforcement learning systems. Previously, he has worked on knowledge transfer for reinforcement learning and learning acceleration for deep learning architectures.

Oriol Vinyals. Oriol Vinyals is a Principal Scientist at Google DeepMind, and a team lead of the Deep Learning group. His work focuses on Deep Learning and Artificial Intelligence. Prior to joining DeepMind, Oriol was part of the Google Brain team. He holds a Ph.D. in EECS from the University of California, Berkeley and is a recipient of the 2016 MIT TR35 innovator award. His research has been featured multiple times at the New York Times, Financial Times, WIRED, BBC, etc., and his articles have been cited over 65000 times. His academic involvement includes program chair for the International Conference on Learning Representations (ICLR) of 2017, and 2018. He has also been an area chair for many editions of the NIPS and ICML conferences. Some of his contributions such as seq2seq, knowledge distillation, or TensorFlow are used in Google Translate, Text-To-Speech, and Speech recognition, serving billions of queries every day, and he was the lead researcher of the AlphaStar project, creating an agent that defeated a top professional at the game of StarCraft, achieving Grandmaster level, also featured as the cover of Nature. At DeepMind he continues working on his areas of interest, which include artificial intelligence, with particular emphasis on machine learning, deep learning and reinforcement learning.

3.1.2 Advisors

Anca Dragan. Anca Dragan is an Assistant Professor in the EECS Department at UC Berkeley. Her goal is to enable robots to work with, around, and in support of people. She runs the InterACT Lab, where the focus is on algorithms for human-robot interaction – algorithms that move beyond the robot’s function in isolation, and generate robot behavior that also accounts for interaction and coordination with end-users. The lab works across different applications, from assistive robots, to manufacturing, to autonomous cars, and draw from optimal control, planning, estimation, learning, and cognitive science. She also helped found and serve on the steering committee for the Berkeley AI Research (BAIR) Lab, and am a co-PI of the Center for Human-Compatible AI. She was also honored by the Sloan Fellowship, MIT TR35, the Okawa award, and an NSF CAREER award.

Chelsea Finn. Chelsea Finn is an Assistant Professor in Computer Science and Electrical Engineering at Stanford University. Finn’s research interests lie in the capability of robots and other agents to develop broadly intelligent behavior through learning and interaction. To this end, her work has included deep learning algorithms for concurrently learning visual perception and control in robotic manipulation skills, inverse reinforcement methods for scalable acquisition of nonlinear reward functions, and meta-learning algorithms that can enable fast, few-shot adaptation in both visual perception and deep reinforcement learning. Finn received her Bachelor’s degree in Electrical Engineering and Computer Science at MIT and her PhD in Computer Science at UC Berkeley. Her research has been recognized through the ACM doctoral dissertation award, an NSF graduate fellowship, a Facebook fellowship, the C.V. Ramamoorthy Distinguished Research Award, and the MIT Technology Review 35 under 35 Award, and her work has been covered by various media outlets, including the New York Times, Wired, and Bloomberg. Throughout her career, she has sought to increase the representation of underrepresented minorities within CS and AI by developing an AI outreach camp at Berkeley for underprivileged high school students, a mentoring program for underrepresented undergraduates across four universities, and leading efforts within the WiML and Berkeley WiCSE communities of women researchers.

David Ha. David Ha is a Research Scientist at Google Brain. His research interests include Recurrent Neural Networks, Creative AI, and Evolutionary Computing. Prior to joining Google, He worked at Goldman Sachs as a Managing Director, where he co-ran the fixed-income trading business in Japan. He obtained undergraduate and graduate degrees in Engineering Science and Applied Math from the University of Toronto.

Sergey Levine. Sergey Levine received a BS and MS in Computer Science from Stanford University in 2009, and a Ph.D. in Computer Science from Stanford University in 2014. He joined the faculty of the Department of Electrical Engineering and Computer Sciences at UC Berkeley in fall 2016. His work focuses on machine learning for decision making and control, with an emphasis on deep learning and reinforcement learning algorithms. Applications of his work include autonomous robots and vehicles, as well as computer vision and graphics. He has previously served as the general chair for the Conference on Robot Learning, program co-chair for the International Conference on Learning Representations, and organizer for numerous workshops at ICML, NeurIPS, and RSS. He has also served as co-organizer on the *Learning to Run* and *AI for Prosthetics* NeurIPS competitions.

Manuela Veloso. Manuela Veloso is a Herbert A. Simon University Professor at Carnegie Mellon University and the head of AI research at JPMorgan Chase. She received her Ph.D. in computer science from Carnegie Mellon University in 1992. Since then, she has been a faculty member at the Carnegie Mellon School of Computer Science. Her research focuses on artificial intelligence and robotics, across a range of planning, execution, and learning

algorithms. She cofounded the RoboCup Federation and served as president of AAAI from 2011 to 2016. She is a AAAI, IEEE, AAAS, and ACM fellow.

3.1.3 Partners and Sponsors

We are currently in conversation with potential partners for this year’s competition. Last year, we partnered with and/or received support from Microsoft Research, Preferred Networks, NVIDIA, and Artificial Intelligence Journal (AIJ).

3.2 Resources provided by organizers, including prizes

Mentorship. We will facilitate a community forum through our publicly available Discord server to enable participants to ask questions, provide feedback, and engage meaningfully with our organizers and advisory board. We hope to foster an active community to collaborate on these hard problems.

Computing Resources. In concert with our efforts to provide open, democratized access to AI, we are in conversation with potential sponsors to provide compute grants for teams that self identify as lacking access to the necessary compute power to participate in the competition, as we did in the last iteration of the competition. We will also provide groups with the evaluation resources for their experiments in Round 2, as we did in the last iteration of the competition.

Travel Grants and Scholarships. The competition organizers are committed to increasing the participation of groups traditionally underrepresented in reinforcement learning and, more generally, in machine learning (including, but not limited to: women, LGBTQ individuals, underrepresented racial and ethnic groups, and individuals with disabilities). To that end, we will offer Inclusion@NeurIPS scholarships/travel grants for Round 1 participants who are traditionally underrepresented at NeurIPS to attend the conference. These individuals will be able to apply online for these grants; their applications will be evaluated by the competition organizers and partner affinity groups. We also plan to provide travel grants to enable all of the top participants from Round 2 to attend our NeurIPS workshop. We are in conversation with potential sponsors about providing funding for these travel grants.

Prizes. Currently in discussion with sponsors / partners. In the previous competition we offered 5 NVIDIA GPUs and 10 NVIDIA Jetsons to the top teams. In addition, we provided two research prizes for solutions that used pure reinforcement learning and pure imitation learning.

3.3 Support and facilities requested

Due to the quality of sponsorships and industry partnerships secured last year, we only request facility resources and ticket reservations. We aim to present at the NeurIPS 2020 Competition Workshop. We will invite guest speakers, organizers, Round 2 participants, and some Round 1 participants. To allow these people to attend NeurIPS, we request 30 reservations for NeurIPS. We will fund travel for participants (as mentioned above), and the others will be present at their own expense.

References

- [1] Dario Amodei and Danny Hernandez. <https://blog.openai.com/ai-and-compute/>, May 2018. URL <https://blog.openai.com/ai-and-compute/>.
- [2] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.
- [3] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.
- [4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Matheus Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- [5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [6] Gabriel V Cruz Jr, Yunshu Du, and Matthew E Taylor. Pre-training neural networks with human demonstrations for deep reinforcement learning. *arXiv preprint arXiv:1709.04083*, 2017.
- [7] DeepMind. Alphastar: Mastering the real-time strategy game starcraft ii, 2018. URL <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/>.
- [8] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *The 33rd International Conference on Machine Learning*, pages 49–58, 2016.
- [9] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.
- [10] Yasuhiro Fujita, Toshiki Kataoka, Prabhat Nagarajan, and Takahiro Ishikawa. Chain-erl: A deep reinforcement learning library. In *The 23rd Conference on Neural Information Processing Systems, Deep Reinforcement Learning Workshop*, 2019.
- [11] Yang Gao, Ji Lin, Fisher Yu, Sergey Levine, Trevor Darrell, et al. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

- [12] William H Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, et al. The minerl competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:1904.10079*, 2019.
- [13] William H. Guss, Cayden Codel*, Katja Hofmann*, Brandon Houghton*, Noboru Kuno*, Stephanie Milani*, Sharada Mohanty*, Diego Perez Liebana*, Ruslan Salakhutdinov*, Nicholay Topin*, Manuela Veloso*, and Phillip Wang*. The MineRL competition on sample efficient reinforcement learning using human priors. In *The 23rd Conference on Neural Information Processing Systems Competition Track*, 2019.
- [14] William H. Guss*, Brandon Houghton*, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. In *The 28th International Joint Conference on Artificial Intelligence*, 2019.
- [15] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *The 32nd AAAI Conference on Artificial Intelligence*, 2018.
- [16] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad G. Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *The 32nd AAAI Conference on AI*, 2018.
- [17] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *The 32nd AAAI Conference on Artificial Intelligence*, 2018.
- [18] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *The 20th Conference on Neural Information Processing Systems*, 2016.
- [19] Jeremy Hsu. Ai takes on popular minecraft game in machine-learning contest, Nov 2019. URL <https://www.nature.com/articles/d41586-019-03630-0>.
- [20] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *The 25th International Joint Conference on Artificial Intelligence*, pages 4246–4247, 2016.
- [21] Łukasz Kidziński, Sharada P Mohanty, Carmichael F Ong, Jennifer L Hicks, Sean F Carroll, Sergey Levine, Marcel Salathé, and Scott L Delp. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In

- The NIPS’17 Competition: Building Intelligent Systems*, pages 101–120. Springer, 2018.
- [22] Stephanie Milani, Nicholay Topin, Brandon Houghton, William H. Guss, Sharada P. Mohanty, Oriol Vinyals, and Noboru Sean Kuno. The MineRL competition on sample-efficient reinforcement learning using human priors: A retrospective. *arXiv preprint, arXiv:2003.05012*, 2020.
 - [23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
 - [24] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *The 33rd International Conference on Machine Learning*, pages 1928–1937, 2016.
 - [25] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.
 - [26] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of memory, active perception, and action in Minecraft. *arXiv preprint arXiv:1605.09128*, 2016.
 - [27] OpenAI. Openai five, Sep 2018. URL <https://blog.openai.com/openai-five/>.
 - [28] Ameya Panse, Tushar Madheshia, Anand Sriraman, and Shirish Karande. Imitation learning on atari using non-expert human annotations. 2018.
 - [29] Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, Andre Kramer, Sam Devlin, Raluca D Gaina, and Daniel Ionita. The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) Competition. *arXiv preprint arXiv:1901.08129*, 2019.
 - [30] Christoph Salge, Michael Cerny Green, Rodgrigo Canaan, and Julian Togelius. Generative Design in Minecraft (GDMC): Settlement Generation Competition. In *The 13th International Conference on the Foundations of Digital Games*, page 49. ACM, 2018.
 - [31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

- [32] Sam Shead. Minecraft diamond challenge leaves ai creators stumped, Dec 2019. URL <https://www.bbc.com/news/technology-50720823>.
- [33] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. *arXiv preprint arXiv:1712.07294*, 2017.
- [34] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.
- [35] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362, 2018.
- [36] Synced. Neurips 2019 will host minecraft reinforcement learning competition, May 2019. URL <https://medium.com/syncedreview/neurips-2019-will-host-minecraft-reinforcement-learning-competition-146e8bc8da1>.
- [37] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *The 31st AAAI Conference on AI*, 2017.
- [38] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *The 30th AAAI Conference on AI*, 2016.
- [39] James Vincent. Ai has bested chess and go, but it struggles to find a diamond in minecraft, Dec 2019. URL <https://www.theverge.com/2019/12/13/21020230/ai-minecraft-minerl-diamond-challenge-microsoft-reinforcement-learning>.
- [40] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michael Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 2019.