MineRL: A Large-Scale Dataset of Minecraft Demonstrations

William H. Guss[*][†] , **Brandon Houghton**[*] , **Nicholay Topin** , **Phillip Wang** , **Cayden Codel** , **Manuela Veloso** and **Ruslan Salakhutdinov**
Carnegie Mellon University, Pittsburgh, PA 15289, USA
{wguss, bhoughton, ntopin, pkwang, ccodel, mmv, rsalakhu}@cs.cmu.edu

Abstract

The sample inefficiency of standard deep reinforcement learning methods precludes their application to many real-world problems. Methods which leverage human demonstrations require fewer samples but have been researched less. As demonstrated in the computer vision and natural language processing communities, large-scale datasets have the capacity to facilitate research by serving as an experimental and benchmarking platform for new methods. However, existing datasets compatible with reinforcement learning simulators do not have sufficient scale, structure, and quality to enable the further development and evaluation of methods focused on using human examples. Therefore, we introduce a comprehensive, large-scale, simulator-paired dataset of human demonstrations: MineRL. The dataset consists of over 60 million automatically annotated state-action pairs across a variety of related tasks in Minecraft, a dynamic, 3D, open-world environment. We present a novel data collection scheme which allows for the ongoing introduction of new tasks and the gathering of complete state information suitable for a variety of methods. We demonstrate the hierarchality, diversity, and scale of the MineRL dataset. Further, we show the difficulty of the Minecraft domain along with the potential of MineRL in developing techniques to solve key research challenges within it.

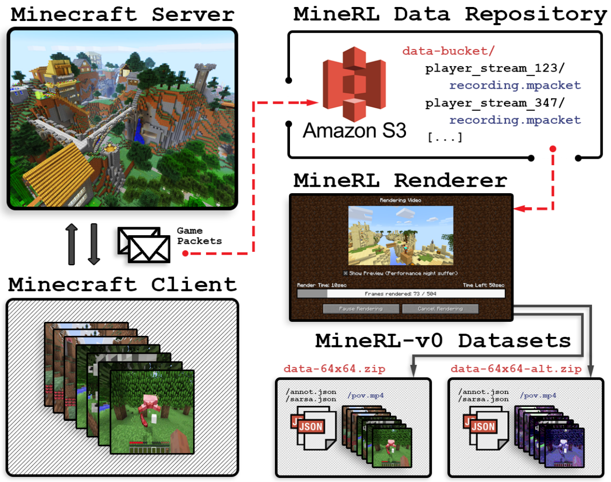

Figure 1: A diagram of the MineRL data collection platform. Our system renders demonstrations from packet-level data, so the game state and rendering parameters can be changed.

1 Introduction

As deep reinforcement learning (DRL) methods are applied to increasingly difficult problems, the number of samples used for training increases. For example, Atari 2600 games [Bellemare *et al.*, 2013] have been used to evaluate DQN [Mnih *et al.*, 2015], A3C [Mnih *et al.*, 2016], and Rainbow DQN, which each require from 44 to over 200 million frames (200 to over 900 hours) to achieve human-level performance [Hessel *et al.*, 2018]. On more complex domains: OpenAI Five utilizes 11,000+ years of Dota 2 gameplay [OpenAI, 2018], AlphaGoZero uses 4.9 million games of self-play in Go [Silver *et al.*, 2017], and AlphaStar uses 200 years of Starcraft II gameplay [DeepMind, 2018].

This inherent sample inefficiency precludes the application of standard DRL methods to real-world problems without leveraging data augmentation techniques [Tobin *et al.*, 2017], [Andrychowicz *et al.*, 2018], domain alignment methods [Wang *et al.*, 2018], or carefully designing real-world environments to allow for the required number of trials [Levine *et al.*, 2018]. Recently, techniques leveraging trajectory examples, such as imitation learning and Bayesian reinforcement learning methods, have been successfully applied to older benchmarks and real-world problems where samples from the environment are costly.

However, these techniques are still not sufficiently sample efficient for a large class of complex real-world domains.

As noted by [Kurin *et al.*, 2017], several subfields of machine learning have been catalyzed by the introduction of datasets and efficient large-scale data collection schemes, such as Switchboard [Godfrey *et al.*, 1992] and ImageNet [Deng *et al.*, 2009]. Though the reinforcement learning community has created an extensive range of benchmark simulators, there is currently a lack of large-scale labeled datasets of human demonstrations for domains with a broad

[*] Equal contribution.
[†] Contact Author.

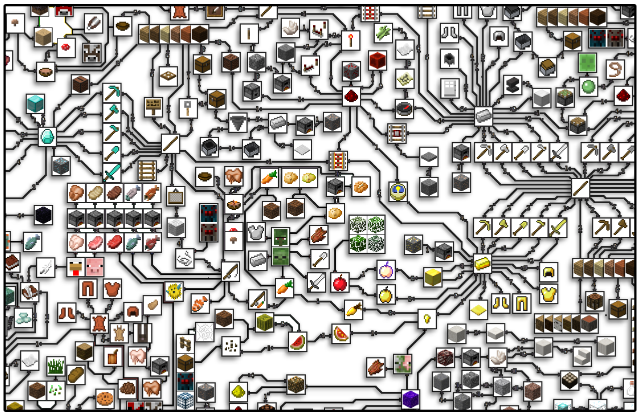

Figure 2: A subset of the Minecraft item hierarchy (totaling 371 unique items). Each node is a unique Minecraft item, block, or non-player character, and a directed edge between two nodes denotes that one is a prerequisite for another. Each item presents is own unique set of challenges, so coverage of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

range of structural constraints and tasks.

Therefore, we introduce MineRL, a large-scale, dataset of over 60 million state-action pairs of human demonstrations across a range of related tasks in Minecraft. To capture the diversity of gameplay and player interactions in Minecraft, MineRL includes six tasks with a variety of research challenges including open-world multi-agent interactions, long-term planning, vision, control, and navigation, as well as explicit and implicit subtask hierarchies. We provide implementations of these tasks as sequential decision-making environments in an existing Minecraft simulator. *Additionally, we introduce a novel platform and methodology for the continued collection of human demonstrations in Minecraft.* As users play on our publicly available game server, we record packet-level information, which allows perfect reconstruction of each player's view and actions. This platform enables the addition of new tasks to the MineRL dataset and automatic annotation to complement current and future methods applied to Minecraft. Demo videos and more details about the dataset can be found at http://minerl.io.

2 Environment: Minecraft

2.1 Description

Minecraft is a compelling domain for the development of reinforcement and imitation learning based 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. It can be played in a single-player mode or a multi-player mode, where all players exist in and affect the same world. Games are played across many sessions, for tens of hours total, per player. Notably, the procedurally generated world is composed of discrete blocks which 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).

As an open-world game, Minecraft has no single definable objective. Instead, players develop their own subgoals which form a multitude of natural hierarchies. Though these hierarchies can be exploited, their size and complexity contribute to Minecraft's inherent difficulty. One such hierarchy is that of *item collection*: for a large number of objectives in Minecraft, players must create specific tools, materials, and items which require the collection of a strict set of requisite items. The aggregate of these dependencies forms a large-scale task hierarchy (see Figure 2).

In addition to obtaining items, implicit hierarchies emerge through other aspects of gameplay. For example, players (1) construct structures to provide safety for themselves and their stored resources from naturally occurring enemies and (2) explore the world in search of natural resources, often engaging in combat with non-player characters. Both of these gameplay elements have long time horizons and exhibit flexible hierarchies due to situation dependent requirements (such as farming a certain resource necessary to survive, enabling exploration to then gather another resource, and so on).

2.2 Existing Interest

With the development of Malmo [Johnson *et al.*, 2016], a simulator for Minecraft, the environment has garnered great research interest: [Shu *et al.*, 2017], [Tessler *et al.*, 2017], and [Oh *et al.*, 2016] have leveraged Minecraft's massive hierarchality 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 as well as the inability of current approaches to cope with fully embodied human state- and action-spaces and the complexity exhibited in optimal human policies. This inability is further evidenced by the large-body of work developed on Minecraft-like domains which specifically captures restricted subsets of the features of Minecraft [Salge *et al.*, 2014], [Andreas *et al.*, 2017], [Liu *et al.*, 2017].

Bridging the gap between these restricted Minecraft environments and the full domain encountered by humans is a driving force behind the development of MineRL. To do this, MineRL-v0 captures core aspects of Minecraft that have motivated its use as a research domain, including its hierarchality and its large family of intrinsic subtasks. At the same time, MineRL-v0 provides the human priors and rich, automatically generated meta-data necessary to enable current and future research to tackle the full Minecraft domain.

3 Methods: MineRL Data Collection Plaform

Classification and natural language datasets have benefited greatly from the existence of data collection platforms like Mechanical Turk, but, in contrast, the collection of gameplay data usually requires the implementation of a new platform and user acquisition scheme for each game. To that end, we introduce the first end-to-end platform for the collection of player trajectories in Minecraft, enabling the construction of the MineRL-v0 dataset. As shown in Figure 1, 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 IRB consent for having their gameplay anonymously recorded. They then 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 a stand-alone task to complete and receive in-game currency proportional to the amount of reward obtained. For the `Survival` game mode (where there is no known reward function), players receive rewards only for duration of gameplay so as not to impose an artificial reward function. We implement each of these stand-alone tasks in Malmo.

Data Pipeline. Our data pipeline enables the continued expansion of the structured information accompanying MineRL dataset releases; it allows us to resimulate, modify, and augment recorded trajectories into several algorithmically consumable formats. The pipeline serves as an extension to the core Minecraft game code and synchronously resends 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.

Extensibility. Our aim is to use our platform to provide an exhaustive and broad set of multi-task datasets (beyond MineRL-v0) paired with RL environments, spanning natural language, embodied reasoning, hierarchical planning, and multi-agent cooperation. The modular design of the server allows us to obtain data for a growing number of stand-alone tasks. Furthermore, the in-game economy and server community create consistent engagement from the user-base allowing us to collect data at a growing rate without incurring additional costs. The modularity, simulator compatibility, and configurability of the data pipeline also allows new datasets to be created to compliment new techniques leveraging human demonstrations. For example, it is possible to conduct large-scale generalization studies by repeatedly re-rendering the data with different constraints: altered lighting, camera positions (embodied and non-embodied), and other video rendering conditions; the injection of artificial noise in observations, rewards, and actions; and game hierarchy rearrangment (swapping the function and semantics of game items).

4 Results: MineRL-v0

In this section, we introduce and analyze the MineRL-v0 dataset. We first give details about the dataset including its size, form, and packaging. Then we indicate the wide applicability of this initial release by giving a detailed account of the included tasks families, followed by an analysis of the data quality, coverage, and hierarchicality. To frame the usefulness of the MineRL-v0 dataset, in Section 5, we demonstrate the difficulty of our tasks with respect to out-of-the-box methods and the performance increase achieved through basic imitation learning techniques using MineRL-v0.

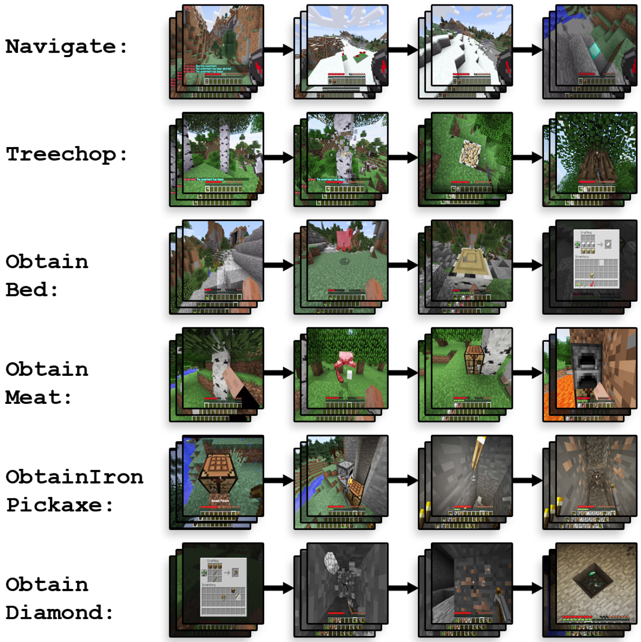

Figure 3: Images of various stages of the six stand-alone tasks (`Survial` gameplay not shown).

4.1 Dataset Details

Size. The MineRL-v0 dataset consists of 500+ hours of recorded human demonstrations over six different tasks from the data collection platform. The released data is comprised of four different versions of the dataset rendered with varied resolutions (64 × 64 and 192 × 256) and textures (default Minecraft and simplified). Each version individually totals to over 60 million state-action pairs with a size of 130 GB and 734 GB for the low and medium resolution datasets respectively.

Form. Each trajectory is a contiguous set of state-action pairs 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 of the keyboard presses on the client, the change in view pitch and yaw (caused by mouse movement), all player GUI click and interaction events, chat messages sent, and agglomorative actions such as item crafting.

Additional Annotations. Human trajectories are accompanied by a large set of automatically generated annotations. For all the stand-alone tasks, we record metrics which indicate the quality of the demonstration, such as timestamped re-

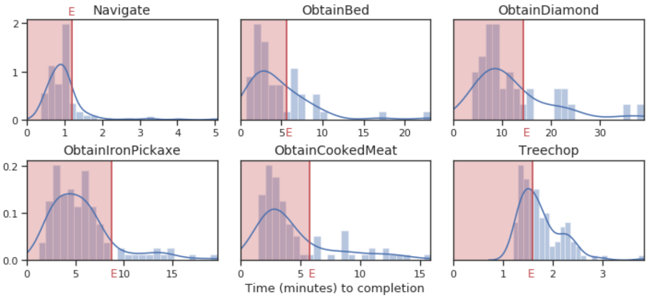

Figure 4: Normalized histograms of the lengths of human demonstration on various MineRL tasks. The red E denotes the upper threshold for expert play on each task.

wards, number of no-ops, number of deaths, and total score. Additionally, the 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.

Packaging. Each version of the dataset is packaged as a Zip archive with one folder per task family and one sub-folder per demonstration. In each trajectory folder, the states and actions are stored as an H.264 compressed MP4 video of the player's POV with a max bit rate of 18Mb/s and a JSON file containing all of the non-visual features of game-state as well as the player's actions corresponding to every frame of the video. Additionally, for specific *task configurations* (simplifications of action and state space) we provide Numpy `.npz` files composed of state-action-reward tuples in vector form, promoting the accessibility of the dataset. The packaged data and accompanying documentation can be downloaded from http://minerl.io.

4.2 Tasks

The initial MineRL-v0 dataset consists of six stand-alone tasks chosen to represent difficult aspects of Minecraft that reflect challenges widely researched in the domain: hierarchality, long-term planning, and complex orienteering. Throughout all tasks, the agent has access to the same set of actions and observations as a human player, as outlined in Section 4.1 All tasks have a time limit, which is part of the observation. Details for each task follow below.

Navigation. In the `Navigate` task, the agent must move to a random goal location over procedurally generated, nonconvex terrain with variable material type and geometry. This is a subtask for many tasks throughout Minecraft. In addition to standard observations, the agent has access to a "compass" observation, which points to a set location, 64 blocks (meters) from the start location. The goal has a small random horizontal offset from this location and may be slightly below surface level, so the agent must find the final goal by searching based on visual features. There are two variants of the provided reward function: sparse (+1 upon reaching the goal, at which point the episode terminates), and dense (reward proportional to distance moved towards the goal).

Tree Chopping. The `Treechop` task replicates obtaining wood for producing further items. Wood is a key resource in Minecraft since it is a prerequisite for all tools (as seen by the placement of sticks in Figure 2 and Figure 6). The agent begins in a forest biome (near many trees) with an iron axe for cutting the trees. The agent is given +1 reward for obtaining each unit of wood, and the episode terminates once the agent obtains 64 units.

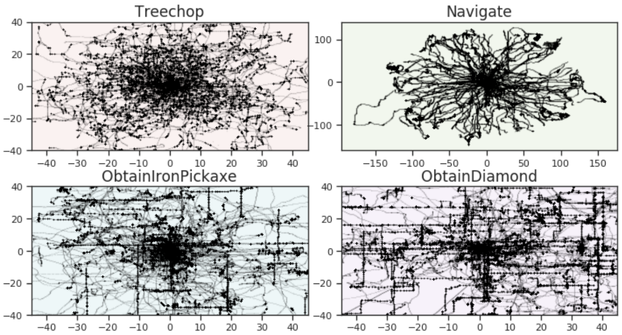

Figure 5: Plots of the XY positions of players in `Treechop`, `Navigate`, `ObtainIronPickaxe`, and `ObtainDiamond` overlaid so each player's individual, random initial location is $(0, 0)$.

Obtain Item. We include four related tasks which require the agent to obtain an item further in the item hierarchy: `ObtainIronPickaxe`, `ObtainDiamond`, `ObtainCookedMeat`, and `ObtainBed`. The agent always begins in a random location without any items; this matches the starting conditions for human players in Minecraft. Different task variants correspond to a different, frequently used item: iron pickaxe, diamond, cooked meat (four variants, one per animal source), and bed (three variants, one per dye color needed). Iron pickaxes are tools required for obtaining key materials. Diamonds are central to high-level Minecraft play, and large portion of gameplay centers around their discovery. Cooked meat is used to replenish stamina, and a bed is required for sleeping. Together, these items represent what a player would need to obtain to survive and access further areas of the game. The agent is given $+1$ reward for obtaining the required item, at which point the episode terminates.

Survival. In addition to data on specific, designed tasks, we provide data in `Survival`, the standard open-ended game mode used by most players. Starting from a random location without any items, players formulate their own high-level goals and obtain items to complete these goals. Data from this task can be used for learning the intricate reward functions followed by humans in open play and the corresponding policies. This data can also be used to train agents attempting to complete the other, structured tasks, or further for extracting policy sketches as in [Andreas *et al.*, 2017].

4.3 Analysis

Human Performance

A majority of the human demonstrations in the dataset fall squarely within expert level play. Figure 4 shows the distribution over players of time required to complete each standalone task. 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.

Coverage

MineRL-v0 has near complete coverage of Minecraft. Within the `Survival` game mode, a large majority of the 371 subtasks for obtaining different items have been demonstrated by players hundreds to tens of thousands of times. Further, some of these subtasks require hours to complete, requiring a long sequence of mining, building, exploring, and combatting enemies. As a result of the large number of task-level annotations, the dataset can be used for large-scale option extraction and skill acquisition, enabling the extension of the work of [Shu *et al.*, 2017] and [Andreas *et al.*, 2017]. Moreover, the rich label hierarchy of the `Obtain<Item>` tasks can be utilized in constructing metrics for the interpretability and quality of extracted options.

In addition to item coverage, the MineRL data collection platform is structured to promote a broad representation of game conditions. The current dataset consists of a diverse set of demonstrations extracted from 1,002 unique player sessions. In the `Survival` game mode, the recorded trajectories collectively cover $24,393,057$ square meters of game content, where a square meter corresponds to one Minecraft block. For all other tasks, each demonstration occurs in a randomly initialized game world, so we collect a large number of unique, disparate trajectories for each task: In Figure 5, we show the top-down position of players over the course of completing each task where the starting state is at $(0, 0)$. Not only does each player act in a different game world, but each player also explores a large region during each task.

Hierarchality

As exemplified by the item graph shown in Figure 2, Minecraft is deeply hierarchical, and the MineRL data collection platform is designed to capture these hierarchies both explicitly and implicitly. As a primary example, the `Obtain<Item>` stand-alone tasks isolate difficult yet overlapping core paths in the item hierarchy. Due to the subtask labelings provided in MineRL-v0, we can inspect and quantify the extent to which these tasks overlap.

A direct measure of hierachality emerges through *item precedence frequency graphs*, graphs where nodes correspond to items obtained in a task and directed edges correspond to the number of times players obtained the source-node item immediately before the target-node item.

These graphs provide a statistical view of the meta-policies of humans and the extent to which their subpolicies transfer between tasks. Figure 6 shows precedence frequency graphs constructed from MineRL trajectories on the `ObtainDiamond` , `ObtainCookedMeat` , and `ObtainIronPickaxe` tasks. Inspection reveals that policies for obtaining a diamond consist of subpolicies which obtain wood, torches, and iron ore. All of these are also required for the `ObtainIronPickaxe` task, but only some of them are used within the `ObtainCookedMeat` task. The effects of these overlapping subpolicies can be seen in Figure 5: players move similarly in tasks with overlapping hierarchies (such as `ObtainIronPickaxe` and `ObtainDiamond`) and move differently in tasks with less overlap. Moreover, these graphs paint a distributional picture of human meta-policies within a task: despite there being necessary graph traversal modes (e.g. wood → stone-pickaxe), depending on the situation, players adapt their strategies by acquiring items typically found later in the item precedence graph through longer paths when earlier items are unavailable. This, in turn, enables the use of MineRL-v0 in developing distributional hierarchical reinforcement learning methods.

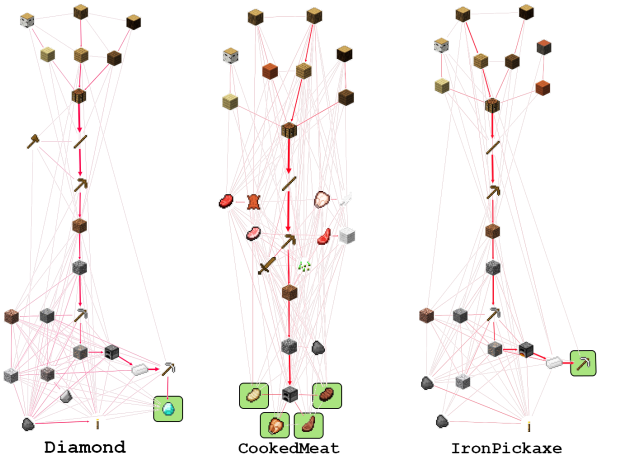

Figure 6: Item precedence frequency graphs for `ObtainDiamond` (left), `ObtainCookedMeat` (middle), and `ObtainIronPickaxe` (right). The thickness of each line indicates the number of times a player collected item A then subsequently item B.

5 Experiments

5.1 Experiment Configuration

To showcase the difficulty of Minecraft, we evaluate the performance of three reinforcement learning methods and one behavioral cloning method on the easiest of our tasks (`Treechop` and `Navigate` (Sparse)), as well as a simplified task with additional, shaped rewards, `Navigate` (Dense). Specifically, we evaluate (1) Dueling Double Deep Q-networks (DQN) [Mnih *et al.*, 2015], an off-policy, Q-learning based method; (2) Pretrain DQN (PreDQN), DQN with additional pretraining steps and the replay buffer initialized with expert demonstrations from MineRL-v0; (3) Advantage Actor Critic (A2C) [Mnih *et al.*, 2016], an on-policy, policy gradient method; and (4) Behavioral Cloning (BC), a method using standard classification techniques to learn a policy from demonstrations. To ensure reproducibility and an accurate evaluation of these methods, we build atop the OpenAI baseline implementations [Dhariwal *et al.*, 2017].

Observations are converted to grey scale and resized to 64x64. Due to the thousands of action combinations in Minecraft and the limitations of the baseline algorithms, we simplify the action space to be 10 discrete actions. However, behavioral cloning does not have such limitations, and performs similarly without the action space simplifications. To

	`Treechop`	`Navigate` (S)	`Navigate` (D)
DQN	3.73 ± 0.61	0.00 ± 0.00	55.59 ± 11.38
A2C	2.61 ± 0.50	0.00 ± 0.00	-0.97 ± 3.23
BC	0.75 ± 0.39	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 1: Results in `Treechop`, `Navigate` (S)parse, and `Navigate` (D)ense, over the best 100 contiguous episodes. ± denotes standard deviation. Note: humans achieve the maximum score for all tasks shown.

use human demonstrations with Pretrained DQN and Behavioral Cloning, we approximate each action with one of our 10 action primitives. We train each reinforcement learning method for 1500 episode (approximately 12 million frames). To train Behavioral Cloning, we use expert trajectories from each respective task family and train until policy performance reaches its maximum.

5.2 Evaluation and Discussion

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 1.

In all tasks, the learned agents perform significantly worse than human performance. `Treechop` exhibits the largest difference: humans achieve a score of 64, but agents achieve scores of less than 4. This suggests that our tasks are quite difficult, especially given that the `Obtain<Item>` tasks build upon the `Treechop` task by requiring the completion of several additional subgoals (≥ 3). We hypothesize that a large source of difficulty comes 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 efficiency: in all tasks, methods that leverage human data perform better. As seen in Figure 7, 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).

6 Related Work

A number of domains have been previously solved through imitation learning and a dataset of human demonstrations. These include the Atari domain using the Atari Grand Challenge dataset [Kurin *et al.*, 2017] and the Super Tux Kart domain using an on-demand dataset [Ross *et al.*, 2011]. Unlike Minecraft, these are simple domains: they have shallow dependency hierarchies and are not open-world. Due to the small action- and state-spaces, these domains have been solved using imitation learning using relatively few samples (9.7 million frames across five games in [Kurin *et al.*, 2017] and 20 thousand frames in [Ross *et al.*, 2011]). In contrast, we present 60 million automatically annotated state-action pairs and do not achieve human performance.

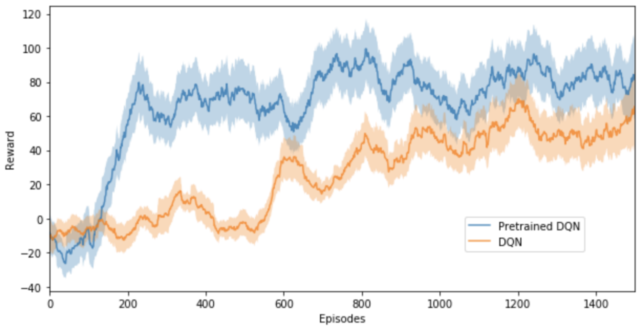

Figure 7: Performance graphs over time with DQN and pretrained DQN on `Navigate` (Dense).

Existing datasets for challenging, unsolved domains are primarily for real-world tasks where a lack of simulators limits the pace of development. The KITTI dataset [Geiger *et al.*, 2013], for example, is a dataset of 3 hours of 3D information on real-world traffic. Similarly, Dex-Net [Mahler *et al.*, 2019] is a dataset of five million grasps with corresponding 3D pointclouds for robotic manipulation. Unlike these datasets, MineRL is directly compatible with a simulator, Malmo, thereby allowing training in the same domain as the data was gathered and comparison to methods not based on imitation learning. Additionally, the scale of MineRL is larger relative to the domain difficulty than the KITTI and Dex-Net datasets.

The only complex, unsolved domain with an existing simulator and large-scale dataset is StarCraft II. However, StarCraft II is not open-world so cannot be used to evaluate methods designed for embodied tasks in 3D environments. The largest dataset is currently StarData [Lin *et al.*, 2017]. Unlike MineRL, it consists of unlabeled, extracted trajectories of standard gameplay. In contrast, MineRL includes a growing number of related tasks which represent different components of the overall Minecraft task hierarchy. In addition, MineRL consists of rich automatically generated annotations including subtask completion, player skill-level, and an API to extend these labels. Together, these properties allow the use and evaluation of methods which exploit hierarchical structures.

7 Conclusion and Future Work

MineRL-v0 currently features 60 million state-action pairs of procedurally annotated human demonstrations in an open-world, simulator-paired environment. It currently contains data for six tasks, none of which can be fully solved with standard deep reinforcement learning methods. Our platform allows for the ongoing collection of demonstrations for both existing and new tasks. Thus, we host MineRL-v0 at a community accessible website, http://minerl.io, and will gather feedback on adding new annotations and tasks to MineRL. As we expand MineRL, we expect it to be increasingly useful for a range of methods including inverse reinforcement learning, hierarchical learning, and life-long learning. We hope MineRL will become a central resource for sequential decision-making research, bolstering many branches of AI toward the common goal of developing methods capable of solving a wider range of real-world environments.

References

[Andreas *et al.*, 2017] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th ICML-Volume 70*, pages 166–175. JMLR. org, 2017.

[Andrychowicz *et al.*, 2018] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279, 2013.

[DeepMind, 2018] DeepMind. Alphastar: Mastering the real-time strategy game starcraft ii, 2018.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.

[Dhariwal *et al.*, 2017] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines, 2017.

[Geiger *et al.*, 2013] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 32(11):1231–1237, 2013.

[Godfrey *et al.*, 1992] John J Godfrey, Edward C Holliman, and Jane McDaniel. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE, 1992.

[Hessel *et al.*, 2018] 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 *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Johnson *et al.*, 2016] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, pages 4246–4247, 2016.

[Kurin *et al.*, 2017] Vitaly Kurin, Sebastian Nowozin, Katja Hofmann, Lucas Beyer, and Bastian Leibe. The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*, 2017.

[Levine *et al.*, 2018] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *IJRR*, 37(4-5):421–436, 2018.

[Lin *et al.*, 2017] Zeming Lin, Jonas Gehring, Vasil Khalidov, and Gabriel Synnaeve. Stardata: A starcraft ai research dataset. In *Thirteenth AIDE Conference*, 2017.

[Liu *et al.*, 2017] Jerry Liu, Fisher Yu, and Thomas Funkhouser. Interactive 3d modeling with a generative adversarial network. In *2017 IC3DV*, pages 126–134. IEEE, 2017.

[Mahler *et al.*, 2019] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

[Mnih *et al.*, 2015] 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.

[Mnih *et al.*, 2016] 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 *ICML*, pages 1928–1937, 2016.

[Oh *et al.*, 2016] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. *arXiv preprint arXiv:1605.09128*, 2016.

[OpenAI, 2018] OpenAI. Openai five, Sep 2018.

[Ross *et al.*, 2011] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th ICIAS*, pages 627–635, 2011.

[Salge *et al.*, 2014] Christoph Salge, Cornelius Glackin, and Daniel Polani. Changing the environment based on empowerment as intrinsic motivation. *Entropy*, 16(5):2789–2819, 2014.

[Shu *et al.*, 2017] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multitask reinforcement learning. *arXiv preprint arXiv:1712.07294*, 2017.

[Silver *et al.*, 2017] 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, 2017.

[Tessler *et al.*, 2017] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Thirty-First AAAI*, 2017.

[Tobin *et al.*, 2017] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

[Wang *et al.*, 2018] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.