DIP-RL: Demonstration-Inferred Preference Learning in Minecraft

Ellen Novoseller [1] **Vinicius G. Goecks** [1] **David Watkins** [2 3] **Josh Miller** [1] **Nicholas Waytowich** [1]

Abstract

In machine learning for sequential decision-making, an algorithmic agent learns to interact with an environment while receiving feedback in the form of a reward signal. However, in many unstructured real-world settings, such a reward signal is unknown and humans cannot reliably craft a reward signal that correctly captures desired behavior. To solve tasks in such unstructured and open-ended environments, we present Demonstration-Inferred Preference Reinforcement Learning (DIP-RL), an algorithm that leverages human demonstrations in three distinct ways, including training an autoencoder, seeding reinforcement learning (RL) training batches with demonstration data, and inferring preferences over behaviors to learn a reward function to guide RL. We evaluate DIP-RL in a tree-chopping task in Minecraft. Results suggest that the method can guide an RL agent to learn a reward function that reflects human preferences and that DIP-RL performs competitively relative to baselines. DIP-RL is inspired by our previous work on combining demonstrations and pairwise preferences in Minecraft, which was awarded a research prize at the 2022 NeurIPS MineRL BASALT competition, Learning from Human Feedback in Minecraft. Example trajectory rollouts of DIP-RL and baselines are located at https://sites.google.com/view/dip-rl.

1. Introduction

In machine learning for sequential decision-making, an algorithmic agent learns to interact with an environment while receiving feedback. In particular, a typical reinforcement learning (RL) agent receives numerical reward feedback reflecting its performance; however, in many real-world settings, such a reward signal is unknown, and, furthermore, humans might not reliably handcraft a reward signal that correctly captures the desired behavior. In addition, real-world environments are often unstructured and open-ended, with complex observations and sets of possible actions, making reward shaping even more difficult. Developing algorithms to solve tasks in such open-ended and unstructured environments without rewards remains a critical challenge for artificial intelligence (AI).

Minecraft has emerged as a state-of-the-art platform for benchmarking sequential decision-making algorithms within the machine learning research community. Minecraft shares many challenges with the real world, as it is open-ended, complex, and does not have a known numerical reward signal. In fact, the Neural Information Processing Systems Conference (NeurIPS) has recently introduced the MineRL BASALT Competition (Shah et al., 2021), in which competing teams train AI agents to compete in a set of four open-ended Minecraft tasks: finding a cave, building a waterfall, building an animal pen and trapping two of the same animal within it, and building a village house.

Pairwise preference-based RL has been shown to be a successful approach for learning RL reward functions when a true reward signal is unknown (Christiano et al., 2017; Lee et al., 2021). In this method, a human compares pairs of trajectory segments and indicates which behavior is preferred within each pair. Although preference-based RL algorithms have demonstrated numerous successes, the approach inherently requires tedious human intervention and time in the decision loop. Furthermore, each pairwise preference label only provides one bit of information, potentially resulting in sample-inefficient learning. Furthermore, in early stages of learning, preference queries often consist exclusively of behavior pairs that are suboptimal or downright poor.

Demonstrations provide an effective means of acquiring examples of good behavior from the outset. Behavioral cloning (BC) is a simple and widely adopted imitation learning method that learns from demonstrations through supervised learning (Argall et al., 2009). However, despite its popularity and ease of implementation, BC can suffer from distribution drift, which hampers its long-term efficacy. Recent state-of-the-art methods in imitation learning, such as

[1]DEVCOM Army Research Laboratory [2]Columbia University, NY, USA [3]Boston Dynamics AI Institute, MA, USA. Correspondence to: Ellen Novoseller <ellen.r.novoseller.civ@army.mil>, Nicholas Waytowich <nicholas.r.waytowich.civ@army.mil>.

The Many Facets of Preference Learning Workshop at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023.

Soft Q Imitation Learning (SQIL) (Reddy et al., 2020), have made progress toward addressing these issues.

Combining the strengths of imitation and preference learning, this work extends previous work on the combination of demonstrations and pairwise preferences in Minecraft (Shah et al., 2022) and is inspired by our solution for the 2022 MineRL BASALT Competition at NeurIPS (Milani et al., 2023), which was awarded a research prize. This hybrid method, which we call Demonstration-Inferred Preference Reinforcement Learning (DIP-RL), aims to harness the benefits of both pairwise preference learning and demonstrations by inferring preferences from an existing human demonstration dataset while avoiding the burden normally incurred by collecting pairwise preferences.

Our approach leverages the key insight that even when it is difficult for people to specify numerical reward signals or online feedback, human demonstrations may still encode significant intuition about human preference for how a task should be performed. Therefore, our approach uses human demonstrations to guide the learning process. In particular, we infer a task reward function using the demonstration data by comparing behavior segments from the expert demonstrations and from agent rollouts obtained during learning, forming a dataset of pairwise comparisons in which demonstrations are preferred to agent behaviors. Pairwise comparison data are beneficial, since qualitative comparisons can be more reliable than handcrafted absolute numerical scores (Basu et al., 2017; Sui et al., 2018; Joachims et al., 2017). Using pairwise comparisons between demonstrated behaviors and agent-environment interaction, we model the underlying reward that captures the desired behavior.

The contributions of this work are as follows:

1. We propose Demonstration-Inferred Preference Reinforcement Learning (DIP-RL), a framework for learning from human demonstrations to solve complex tasks. DIP-RL leverages demonstrations in three distinct ways: a) to train an autoencoder that transforms images to vector embeddings, b) to seed an RL replay buffer, and c) to provide expert trajectory segments to train a reward function.

2. We provide an evaluation of DIP-RL in a tree-chopping task in Minecraft and compare the performance of DIP-RL with a) human demonstrations, b) BC from demonstrations, c) RL without human demonstrations, and d) SQIL (Reddy et al., 2020), a state-of-the-art imitation learning algorithm. The results suggest that DIP-RL performs competitively relative to these baselines.

2. Related Work

2.1. Learning from Demonstrations and Preferences

Pairwise preference-based feedback has shown promise as an intuitive method for incorporating human feedback into policy learning algorithms (Akrour et al., 2011; Christiano et al., 2017; Lee et al., 2021). In a seminal work, Christiano et al. (2017) take advantage of deep RL to train an agent from human feedback, learning a deep reward model from human evaluations in the form of pairwise comparisons between trajectory segments. The application of pairwise preferences was further advanced by active querying methods for designing informative preference queries (Sadigh et al., 2017; Bıyık et al., 2020). PEBBLE (Lee et al., 2021) improves on prior preference-based RL work by leveraging unsupervised pre-training methods, updating a policy and critic via off-policy RL, and relabeling rewards in the replay buffer as the reward model improves. These preference-based RL approaches demonstrate the ability to effectively utilize real-time human feedback to learn complex tasks.

While preference-based RL methods consider comparisons over the learning agent's behaviors, several works learn rewards from relative rankings over demonstrations (Brown et al., 2019; 2020b;a). For instance, Trajectory-ranked Reward EXtrapolation (T-REX) (Brown et al., 2019) employs ranked suboptimal demonstrations to infer and optimize toward a user's intent beyond the quality of the demonstrations. In contrast to works that consider pairwise comparisons purely between the learning agent's behaviors or between demonstrations, DIP-RL leverages both demonstrations *and* agent experience to generate pairwise comparisons, and thus can learn rewards while benefiting from both initial high-quality examples and from the agent's online experience.

This work extends the method presented in the 2021 MineRL BASALT competition by Team NotYourRL (Shah et al., 2022). The approach in Shah et al. (2022) builds on the method proposed in Ibarz et al. (2018), and integrates Deep Q-learning from Demonstrations (DQfD) with a reward model learned from comparisons between demonstrations and agent behaviors. The team used prioritized experience replay and autolabeling of preferences to train a reward model. However, despite promising results, the team found that model performance was not significantly improved by this reward signal, suggesting a potential area for future research. We propose here the use of an autoencoder that transforms images into embeddings for more sample-efficient use with RL. Our main baseline comparison is Soft Q Imitation Learning (SQIL) (Reddy et al., 2020), which labels human demonstration data with +1 rewards while labeling agent experience with rewards of 0. This differs from our method in that SQIL directly labels experience with binary rewards, while DIP-RL uses the demonstrations and agent experience to infer preferences and learn a continuous reward.

2.2. Minecraft as a Learning Environment

Minecraft has emerged as a popular platform for RL research due to its open-world nature and complex dynamics, and offers researchers a rich and diverse environment in which to train and test RL agents. Johnson et al. (2016) introduced Project Malmo, a platform for AI experimentation built on top of Minecraft that provides a sophisticated interface for RL research, opening a wide range of complex tasks to study. To promote the development of sample-efficient RL algorithms, Guss et al. (2019) subsequently introduced MineRL, a large-scale dataset of human demonstrations in Minecraft.

Based on the MineRL project, the MineRL Benchmark for Agents that Solve Almost-Lifelike Tasks (MineRL BASALT[1]) competition (Shah et al., 2021) used the Minecraft environment to promote research in learning from human feedback to enable agents to accomplish tasks that lack easily-definable reward functions. Tasks were defined by a human-readable description with no reward function. Shah et al. (2022); Goecks et al. (2021); Milani et al. (2023) describe some of the most promising solutions from the 2021 and 2022 competitions.

3. Problem Setting

We consider a learning agent that interacts with the Minecraft environment. In this setting, the agent does not observe the full-world state but instead receives an image observation based on its current location and orientation. Additionally, the agent does not observe numerical rewards.

Therefore, we consider a reinforcement learning (RL) problem setting characterized by an episodic, partially-observed Markov decision process without rewards (POMDP\R), $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, P, P_e, \mu, T)$. Here, $\mathcal{S}$ is the underlying state space, $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ yields state transition probabilities, $P_e : \mathcal{S} \times \mathcal{O} \to [0, 1]$ yields observation emission probabilities, $\mu : \mathcal{S} \to [0, 1]$ is the initial state probability distribution, and T is the episode time horizon.

The agent interacts with the environment through a series of roll-out trajectories $\tau = (o_1, a_1, o_2, a_2, \ldots, o_T, a_T, o_{T+1})$, in which the agent receives observations $o_1, \ldots, o_{T+1} \in \mathcal{O}$ and takes actions $a_1, \ldots, a_T \in \mathcal{A}$. A *policy* is a mapping of observations to actions, $\pi : \mathcal{O} \times \mathcal{A} \to [0, 1]$, such that $\pi(a \mid o)$ yields the probability that the agent selects action $a \in \mathcal{A}$ given observation $o \in \mathcal{O}$.

We assume that the agent has access to a set of demonstrations of human interaction with the environment, $\mathcal{D}_{\text{demo}} = \{(o_i, a_i)\}_{i=1}^{M}$, where M is the number of experience tuples in the demonstration dataset.

Learning Objective. While the environment does not include a numerical reward signal, we assume that the human demonstrations in $\mathcal{D}_{\text{demo}}$ reflect an unknown underlying reward function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The agent's objective is to learn a behavior that maximizes r, such that the optimal policy π^* is given by:

$$\pi^* = \text{argmax}_\pi \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim \mu, P, P_e, \pi}[r(s_t, a_t)]. \quad (1)$$

4. Demonstration-Inferred Preference Reinforcement Learning

We present Demonstration-Inferred Preference Reinforcement Learning (DIP-RL), illustrated in Figure 1. DIP-RL infers pairwise preferences from human demonstrations and agent experience to facilitate learning. We learn a reward function from pairwise preferences that compare agent behaviors and demonstration segments. This reward function is then used to inform an RL process, similarly to preference-based RL (Christiano et al., 2017; Lee et al., 2021), but in which we also inject demonstration data. Finally, we use the demonstrations to train an autoencoder that transforms image observations into embeddings to achieve a more sample-efficient RL policy.

4.1. Preferences between Agent and Human Behaviors

We use pairwise preferences to learn a reward function $\hat{r}_\psi : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$, parameterized by ψ, to inform the RL process. Each preference is given between a demonstration segment τ_{demo} and an agent segment τ_{agent}. Although most of the work on preference-based RL compares pairs of agent behaviors (Christiano et al., 2017; Lee et al., 2021), we hypothesize that comparing demonstration-agent pairs will result in preference queries in which the demonstrated trajectory is clearly preferable; in contrast, standard preference-based RL often initially only generates preference queries involving clearly bad behaviors, since no well-performing agent behaviors are yet available.

Preference labels could be assigned either by automatically preferring demonstrated behaviors to agent behaviors or manually by a human. Our experiments consider preferences in which demonstrated behaviors are always preferred to agent behaviors, as first proposed by Team NotYourRL in Shah et al. (2022).

Given a dataset of pairwise preferences $\mathcal{D}_{\text{pref}} = \{\tau_1^{(i)} \succ \tau_2^{(i)}\}_{i=1}^{N}$, where N is the number of pairwise preferences in the dataset, we can model the probability of each preference in terms of the learned reward $\hat{r}_\psi$ via the Bradley-Terry

[1]MineRL BASALT Competition: https://minerl.io/basalt/.

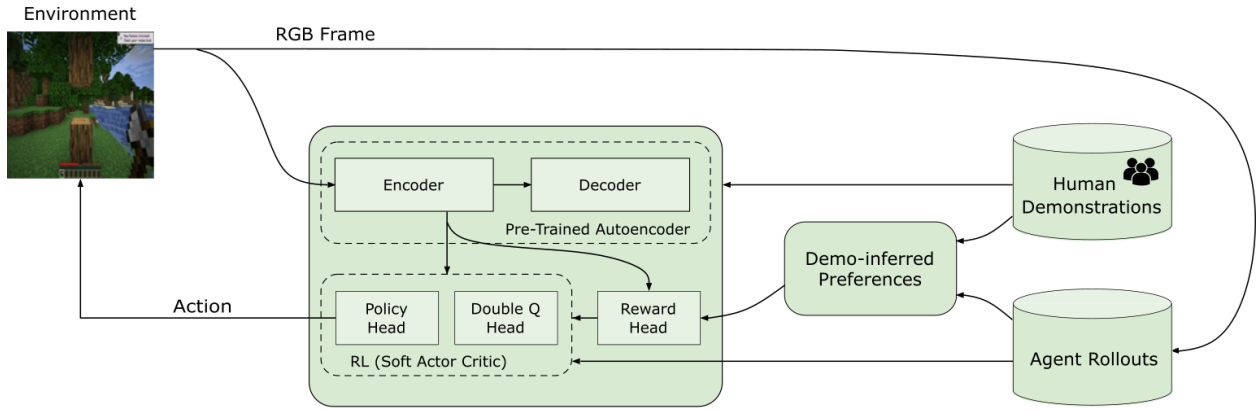


Figure 1. System diagram of the Demonstration-Inferred Preference Reinforcement Learning (DIP-RL) algorithm. DIP-RL leverages human demonstrations in three distinct ways: to 1) train an autoencoder to learn a compact state representation (the autoencoder training data can include nontask-specific demonstrations as well as task-specific trajectories), 2) provide trajectory segments for pairwise preference queries, and 3) provide experience to seed the RL replay buffer. The demonstration-inferred pairwise preferences are used to learn a reward function to inform a reinforcement learning algorithm.

model (Christiano et al., 2017; Lee et al., 2021):

$$P(\tau_i \succ \tau_j) = \frac{1}{1 + \exp\{-(\hat{R}_\psi(\tau_i) - \hat{R}_\psi(\tau_j))\}}, \quad (2)$$

where $\hat{R}_\psi(\tau) = \sum_{(o,a)\in\tau} \hat{r}_\psi(o,a)$ is the total predicted reward in trajectory τ.

The reward function $\hat{r}_\psi$ is then optimized by minimizing the negative log-likelihood of the preference dataset $\mathcal{D}_{\text{pref}}$:

$$J_{\hat{r}}(\psi) = -\log(\mathcal{D}_{\text{pref}}) = -\sum_{(\tau_i \succ \tau_j)\in\mathcal{D}_{\text{pref}}} \log P(\tau_i \succ \tau_j).$$

We regularize the reward learning objective via both a weight decay term regularizing the reward network weights and an L2-penalty on the magnitude of the predicted rewards, which we found necessary to achieve stable learning.

4.2. Off-Policy RL via Soft Actor-Critic

Our method leverages Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as its RL engine. Notably, DIP-RL does not require the use of SAC, but rather could be paired with any off-policy RL algorithm. SAC is a state-of-the-art off-policy actor-critic RL algorithm that attempts to learn a policy that optimizes the maximum entropy objective:

$$J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)}[r_\psi(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))], \quad (3)$$

where $\mathcal{H}$ is entropy. SAC iterates between updating a critic (Q-function) and performing a policy improvement step. The critic is trained by minimizing the objective,

$$J_Q(\theta) = \mathbb{E}_{(o_t,a_t,o_{t+1})\sim\mathcal{D}}\left[\left(Q_\theta(o_t,a_t) - \hat{Q}(o_t,a_t,o_{t+1})\right)^2\right],$$

where θ are the parameters of the Q-network Q_θ, $\mathcal{D}$ is an experience replay buffer, and

$$\hat{Q}(o_t,a_t,o_{t+1}) = \hat{r}_\psi(o_t,a_t) + \gamma\bar{V}_{\bar{\theta}}(o_{t+1}),$$

where $\bar{V}_{\bar{\theta}}(o)$ is the soft target value function,

$$\bar{V}_{\bar{\theta}}(o) = \mathbb{E}_{a\sim\pi_\phi(\cdot|o)}[Q_{\bar{\theta}}(o,a) - \hat{\alpha}\log\pi_\phi(a \mid o)],$$

where ϕ parameterizes the policy, $\bar{\theta}$ is a slowly-moving average of the weights θ and parameterizes the critic target network $Q_{\bar{\theta}}(o,a)$, $\hat{\alpha}$ is a hyperparameter, and the expectation is approximated via Monte Carlo estimation. As in the PEBBLE algorithm (Lee et al., 2021), the most current reward model $\hat{r}_\psi$ is used to label each sampled batch of RL training data.

SAC then performs policy updates by minimizing the KL-divergence between the policy and a Boltzmann distribution given by the Q-function:

$$J_\pi(\phi) = \mathbb{E}_{o\sim\mathcal{D}}\left[KL(\pi_\phi(\cdot \mid o) \;||\; \mathcal{Q}_\theta(o,\cdot))\right],$$

where $\mathcal{Q}_\theta(o,\cdot) \propto \exp\{Q_\theta(o,\cdot)\}$.

Finally, DIP-RL samples RL training batches that mix together data from the experience replay buffer $\mathcal{D}$ and the demonstration dataset $\mathcal{D}_{\text{demo}}$ in pre-specified proportions.

4.3. Autoencoder and Model Architecture

We transform image observations into vector embeddings by training an autoencoder on Minecraft image data. This

approach is inspired by Yarats et al. (2021), in which the authors propose using an autoencoder to improve the sample efficiency of model-free RL with image observations. Similarly to Yarats et al. (2021), we regularize autoencoder training through both weight decay and an L2 penalty on image reconstructions. Unlike in Yarats et al. (2021), however, we do not continue to update the pre-trained autoencoder during RL, since we did not find that this improved performance. This may be because we pre-trained the autoencoder on a sufficiently diverse Minecraft demonstration dataset.

The image embeddings are then passed through a policy head, two Q-heads, and a reward prediction head. Note that we utilize two Q-heads, as this has shown success in previous work with model-free RL (Van Hasselt et al., 2016; Haarnoja et al., 2018).

5. Results

5.1. Task Setup and Demonstration Data Collection

We evaluate DIP-RL and comparisons on a custom variant of the MineRLTreechop-v0 (MineRL documentation) environment. In this task, the agent must collect wood blocks by hitting trees in the environment. We modify the environment to yield 128x128 image observations (rather than 64x64) and require the agent to collect a maximum of 4 logs. Because this environment provides a numerical reward signal (+1 reward every time the agent collects a log), we can straightforwardly evaluate algorithm performance; notably, however, this reward information is hidden from DIP-RL. We fix the environment-world seed in all trials (i.e., the agent always spawns in identical surroundings), and collect 25 human demonstrations of the task.

5.2. Methods Compared

We compare our method, DIP-RL, with the following baseline comparisons: Behavioral Cloning (BC) trained on the demonstration dataset, RL with SAC (which receives the numerical environment reward), and Soft-Q Imitation Learning (SQIL) (Reddy et al., 2020), in which agent experience is labeled with a reward of 0, while demonstration experience tuples are labeled with rewards of $+1$. We also report the performance achieved in the human demonstrations.

5.3. Performance Metrics

In order to quantify the success of our approach, we report 1) the number of logs collected by the agent versus the number of environment steps taken during training, and 2) the maximum number of logs collected by each method compared in this work. These numbers are obtained from experience collected as part of algorithm training for the DIP-RL, SAC, and SQIL comparisons, while BC—which does not interact with the environment during training—is evaluated after completion of training.

5.4. Implementation Details

We resized the Minecraft images (originally in $\mathbb{R}^{360\times640\times3}$) to RGB images in $\mathbb{R}^{128\times128\times3}$. The autoencoder was trained on the combination of 1) the TreeChop dataset described in Section 5.1 and 2) the publicly available FindCave demonstration dataset from the BASALT competition (Milani et al., 2023), in which the demonstrators navigate the environment until they find a cave. We found that this combination balances task-specific images with a diverse range of Minecraft images. Each autoencoder training data batch was composed $\approx 10\%$ of images from the TreeChop demonstration data, while the remainder of the training data was drawn from the cave dataset.

Note that DIP-RL, SQIL, and the SAC baseline all use the same pre-trained autoencoder and leverage SAC as the underlying RL method.

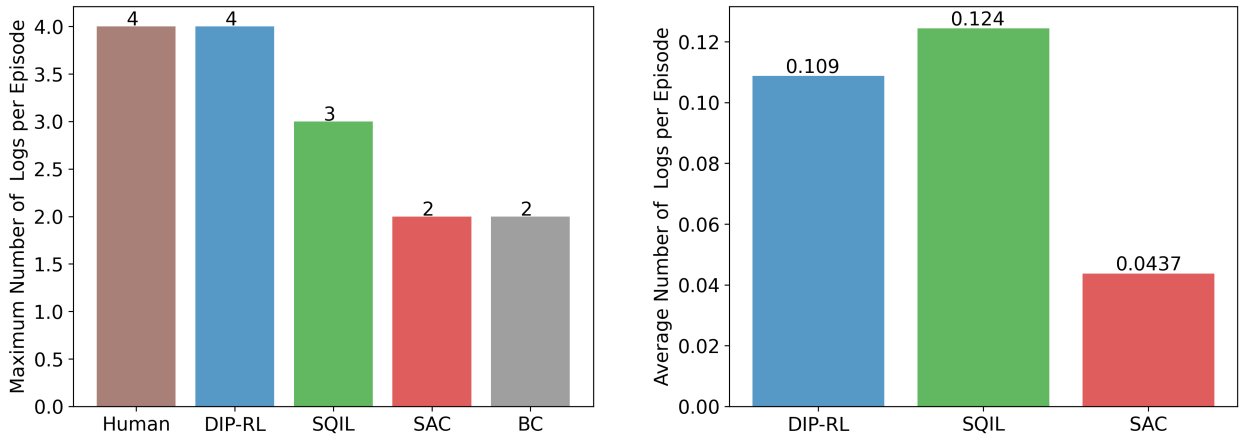
In the 2022 MineRL BASALT competition, the baseline agent was controlled by the hierarchical discrete action space in Baker et al. (2022), which comprises all possible combinations of binary buttons (e.g. attack, sprint, jump, sneak) and discretized camera commands in, by default, 11 bins each for the horizontal and vertical directions. This scheme led to an action space too large to learn with our dataset, with 8641 possible button combinations and 121 possible camera commands. This presented an unnecessary challenge for our RL algorithms since not all button combinations are relevant for completing the proposed task. To reduce the action space, we disabled all nonrelevant actions, giving the agent access to only the attack, move forward, move backward, and jump buttons, and restricted camera movements to a single increment to the left, right, up, and down directions. This reduced the action space from 8641 buttons and 121 discretized camera combinations to 24 buttons and 9 discretized camera combinations.

DIP-RL, SQIL, and SAC were each trained in an experiment run that included 800,000 steps in the environment.

5.5. Results

Results are illustrated in Figure 2, which presents the maximum and average numbers of logs collected by each method. For DIP-RL, SQIL, and SAC, the maximum and average are taken over all training episodes during the algorithm run. The human demonstrations serve as an oracle baseline, since the human always collects the maximum number of logs, while BC illustrates the performance of the agent trained with no reward function, either learned or returned from the environment.

Figure 2(a) suggests that our proposed methodology, DIP-



(a) (b)

Figure 2. a) Maximum number of logs collected over all episodes by DIP-RL and RL-based baseline comparisons during training, as well as by BC during evaluation (since BC does not perform environment rollouts during training) and by the human demonstrator. b) Number of logs collected per episode by DIP-RL and each RL-based baseline, averaged over all RL training episodes.

RL, may best align with human performance in terms of the the maximum number of logs collected over all episodes. It is closely followed by SQIL, with SAC and BC demonstrating comparable performance to each other. Figure 2(b) reflects the performance of the three RL-based methods on average over all training episodes. SQIL appears the most consistent over time, exhibiting a 13.8% increase in the average quantity of logs collected per episode relative to our proposed DIP-RL method.

In terms of the mean log count collected per episode, both DIP-RL and SQIL surpass the standalone RL method, with enhancements of 149.4% and 183.7%, respectively.

6. Discussion

Our results suggest that Demonstration-Inferred Preference Reinforcement Learning (DIP-RL) can effectively utilize human demonstrations to learn reward functions from inferred preferences to train RL agents. This indicates that DIP-RL may be a valuable tool in complex and unstructured environments that lack reward signal information.

From the results presented in Figure 2, we see that DIP-RL is able to reach the maximum number of logs collectable in a single episode, aligning with human performance. Inconsistency in performance across training episodes suggests a possible sensitivity to initial conditions or algorithm hyperparameters. The Soft Q-learning from Imitation (SQIL) method gathered more logs on average per episode than DIP-RL, despite being outperformed in the maximum log count. SQIL's increased average log collection relative to DIP-RL may occur because DIP-RL learns from a reward signal that evolves over time, which could lead to learning instability. However, DIP-RL may hold more potential to represent nuances in reward, since unlike SQIL, it has the potential to assign high reward labels when the agent performs well relative to the demonstrations. It would be interesting to further compare the rewards learned in DIP-RL to the binary reward labels assigned in SQIL.

7. Conclusion

This work presented DIP-RL, an approach that uses human demonstrations in three ways to inform an RL agent. In particular, DIP-RL uses both demonstration and agent trajectories to infer preference comparisons and learn a reward function to train RL agents in unstructured environments. We tested DIP-RL on a tree chopping task in Minecraft and found that it could match human performance in terms of the maximum log count per episode and is competitive with baselines.

Our initial results support the potential of DIP-RL, and of pairwise comparisons between agent and demontration behaviors, especially in scenarios where reward signals are difficult to define but demonstrations are available. In future work, we aim to improve the learning stability of DIP-RL and to evaluate its performance across different tasks. We believe that DIP-RL can contribute significantly to RL and imitation learning as a method to efficiently leverage demonstrations and to infer preferences from demonstrations, and that DIP-RL can be a useful tool for machine learning practitioners working in open-ended and unstructured environments.

Acknowledgements

This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Numbers W911NF-23-2-0072, W911NF-18-2-0244, and W911NF-22-2-0084. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Akrour, R., Schoenauer, M., and Sebag, M. Preference-based policy learning. In Gunopulos, D., Hofmann, T., Malerba, D., and Vazirgiannis, M. (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 12–27, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-23780-5.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (VPT): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Basu, C., Yang, Q., Hungerman, D., Singhal, M., and Dragan, A. D. Do you want your autonomous car to drive like you? In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 417–425, 2017.

Brown, D., Goo, W., Nagarajan, P., and Niekum, S. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pp. 783–792. PMLR, 2019.

Brown, D., Coleman, R., Srinivasan, R., and Niekum, S. Safe imitation learning via fast Bayesian reward inference from preferences. In *International Conference on Machine Learning*, pp. 1165–1177. PMLR, 2020a.

Brown, D. S., Goo, W., and Niekum, S. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pp. 330–359. PMLR, 2020b.

Bıyık, E., Palan, M., Landolfi, N. C., Losey, D. P., Sadigh, D., et al. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning*, pp. 1177–1190. PMLR, 2020.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Goecks, V. G., Waytowich, N., Watkins, D., and Prakash, B. Combining learning from human feedback and knowledge engineering to solve hierarchical tasks in minecraft. *arXiv preprint arXiv:2112.03482*, 2021.

Guss, W. H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., and Salakhutdinov, R. MineRL: a large-scale dataset of Minecraft demonstrations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 2442–2448, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in Atari. *Advances in neural information processing systems*, 31, 2018.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. Accurately interpreting clickthrough data as implicit feedback. In *Acm Sigir Forum*, volume 51, pp. 4–11. Acm New York, NY, USA, 2017.

Johnson, M., Hofmann, K., Hutton, T., and Bignell, D. The Malmo platform for artificial intelligence experimentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 4246–4247. AAAI Press, 2016. ISBN 9781577357704.

Lee, K., Smith, L. M., and Abbeel, P. PEBBLE: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, pp. 6152–6163. PMLR, 2021.

Milani, S., Kanervisto, A., Ramanauskas, K., Schulhoff, S., Houghton, B., Mohanty, S., Galbraith, B., Chen, K., Song, Y., Zhou, T., et al. Towards solving fuzzy tasks with human feedback: A retrospective of the MineRL BASALT 2022 competition. *arXiv preprint arXiv:2303.13512*, 2023.

MineRL documentation. MineRLTreechop-v0 environment. https://minerl.io/docs/environments/#minerltreechop-v0. Accessed: 2023-06-15.

Reddy, S., Dragan, A. D., and Levine, S. SQIL: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations*, 2020.

Sadigh, D., Dragan, A. D., Sastry, S., and Seshia, S. A. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

Shah, R., Wild, C., Wang, S. H., Alex, N., Houghton, B., Guss, W., Mohanty, S., Kanervisto, A., Milani, S., Topin, N., et al. The MineRL BASALT competition on learning from human feedback. *arXiv preprint arXiv:2107.01969*, 2021.

Shah, R., Wang, S. H., Wild, C., Milani, S., Kanervisto, A., Goecks, V. G., Waytowich, N., Watkins-Valls, D., Prakash, B., Mills, E., et al. Retrospective on the 2021 MineRL BASALT competition on learning from human feedback. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 259–272. PMLR, 2022.

Sui, Y., Zoghi, M., Hofmann, K., and Yue, Y. Advancements in dueling bandits. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 5502–5510, 2018.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10674–10681, 2021.