Dual Representation for Human-in-the-loop Robot Learning

Dhruva Bansal
Stanford University

Yilun Hao
Stanford University

Ayano Hiranaka
Stanford University

Roberto Martin-Martin
Stanford University

Chen Wang
Stanford University

Ruohan Zhang
Stanford University

Abstract

What does it take to make reinforcement leaning (RL) work for real robots? One promising approach is to incorporate human guidance, in the forms of evaluation or preference, to speed up learning. In human-in-the-loop RL, understanding and representing the mental models of human trainers is a key challenge. In this work, we propose a dual representation framework for human-in-the-loop robot learning. The robot learner keeps two representations, one for learning the low-level control policy, and the other as a mental model of human trainers. We address the challenge of how to make these systems work together, while maintaining the unique advantage associated with each system. We show that in continuous control tasks, our proposed learning algorithms based on the dual representation hypothesis can lead to significant improvements in task performance and sample efficiency.

1. Introduction

Learning-based approaches face many challenges in solving real-world robotics tasks, and *sample efficiency* is perhaps one of the most critical ones. For example, despite their successes in simulation environments [23, 28], deep reinforcement learning (RL) algorithms may require millions of training samples to learn a good policy [23]. Different types of human guidance are introduced as sources of domain knowledge to speed up learning [34, 35], such as demonstrations [27, 2, 24], evaluation, and preference.

In recent years we have seen a great deal of research energy devoted to studying the latter two learning frameworks, due to 1) their generality, 2) the relative ease to collect such human guidance data, 3) and them complementing standard training signals such as reward or demonstrations well. In learning from human *evaluative feedback*, human trainers monitor the learning process of an agents, and provide a scalar signal to indicate whether the observed behavior is desirable. The agent then learns a policy such that maximizes the human evaluation. In *preference learning*, the learning agent query human trainers for their preferences over a set of exhibited behavior trajectories. The agent then learns a policy or an external reward function from human preference.

In these learning frameworks, understanding human trainers is perhaps as important as making learning agents more intelligent. Many previous works have shown that correctly representing, interpreting, and utilizing human teaching signals can significantly improve the learning results. The very first question is to understand the *representation* used by the humans during the training, and the differences between the representation of the robot and that of the humans. Concretely, in terms of state and action space representation, robots need a fine-grained state and action space to perform rigorous control for continuous control tasks. However, when human observes, evaluates, and guide robot behaviors, the representation humans use must be different, leading to so called "dual representation" hypothesis.

At least two reasons account for the dual representation hypothesis. First, cognitive scientists conjecture that human use an abstract, symbolic, and high-level representation to reason about other agents' (artificial or biological) decisions. That is, in the context of "fast and slow" systems proposed by Kahneman and colleagues [17], humans may use the "slow" system and its representation. Second, due to embodiment mismatch in sensors and actuators, it is unlikely for humans to evaluate the robot in fine-grained state and action space, e.g., providing precise guidance at millimeter and millisecond scale.

Inspired by the fast and slow systems hypothesis, we propose a dual representation framework for human-in-the-

Contributions: D.B, Y.H implemented models and conducted experiments, A.H, R.Z designed the Robosuite simulation tasks and real robot setup, R.Z, R.M, C.W advised the project, R.Z, Y.H, A.H wrote the paper

loop robot learning. In this framework, a robot acts like the "fast" system that adopts a fine-grained state representation for continuous control tasks. The human trainers serve as the "slow" system that evaluate and guide the robot, and use *scene graph* as an abstract state representation. The robot learner keeps two representations, one for learning the low-level control policy, and the other as a mental model of human trainers. We address the challenge of how to make these systems work together, while maintaining the unique advantage associated with each system. We show that in five continuous control tasks, when the robot either learns from human evaluation or preference, our proposed learning algorithms based on the dual representation hypothesis can lead to significant improvement in task performance and sample efficiency.

2. Related Works

2.1. Human-in-the-Loop Training for Embodied AI Agents

How to best leverage human knowledge in training AI agents has been a main research topic. Learning from human evaluative feedback has the advantage of placing minimum demand on both the human trainer's expertise and the ability to provide guidance. Depending on how human evaluative feedback is interpreted, researchers proposed Advice [13, 9], TAMER [18, 30], and COACH [22, 4]. The TAMER+RL framework extends TAMER by learning from both human feedback and environment reward simultaneously [19, 20].

While evaluative feedback often targets a state-action pair, in preference learning [32], human trainers indicate their preference over a set of trajectories. Preferences can be used to directly learn policies [31, 8], learn a preference model [12], or learn a reward function [33, 1]. Recent works often learn a hypothesized latent human reward function from the communicated preferences, and combine preference learning with deep RL [10, 26, 5, 11].

2.2. Scene Graph and Abstract Representations

Scene graph is a form of abstract representation for state information, in which objects are represented as nodes, and relations between objects are represented as edges [16, 3, 25, 15]. Scene graphs also explicitly and compactly store information about objects' geometry, placement, semantics, and relationships, which makes them suitable for many tasks that require sophisticated reasoning about these types of information, such as visual question answering, visual search, and common sense reasoning. In this work, we argue that scene graph could also be a powerful representation in human-in-the-loop robot learning.

3. Dataset

A dataset is not required for human-in-the-loop learning. In evaluative feedback, the human rates the actions the agent take according to the learnt policy. In preference learning, agent's trajectories are generated prior to training for humans to compare and express preference.

4. Method

A standard reinforcement learning problem is formalized as a Markov decision process (MDP), defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ [29].

- Let t be a discrete timestep.
- $\mathcal{S}$ is a set of environment states s which encodes relevant information for an agent's decision. In this work, all environment states are *continuous*.
- $\mathcal{G}$ is a set of abstract, discrete states g introduced by the scene graph.
- $\mathcal{A}$ is a set of agent actions. In this work, all actions are *continuous*.
- $\mathcal{T}$ is the state transition function which describes $p(s'|s, a)$, i.e., the probability of entering state s' when an agent takes action a in state s.
- $\mathcal{R}$ is a reward function. $r(s, a, s')$ denotes the scalar reward agent received on transition from s to s' under action a.
- $\gamma \in [0, 1]$ is a discount factor that indicates how much the agent values an immediate reward compared to a future reward.
- $H(s, a)$ is the human evaluative feedback in state s for action a.

Additionally, $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is a policy which specifies the probability distribution of selecting actions in a given state. The goal for a learning agent is to find an optimal policy π^* that maximizes the expected cumulative reward.

The proposed dual representation framework is shown in Fig. 1. All learning agents in this work use deep neural networks to learn policies or optimize other objectives.

4.1. Active Learning from Evaluative Feedback with Dual Representation

For continuous control RL tasks we are interested in, Soft Actor-Critic (SAC) is a standard learning algorithm [14] which will serve as the backbone of our learning agents. As in the TAMER framework [18], human trainers watch the learning process of the SAC agent. We adopt an *active* learning setting, in which the TAMER-SAC agent

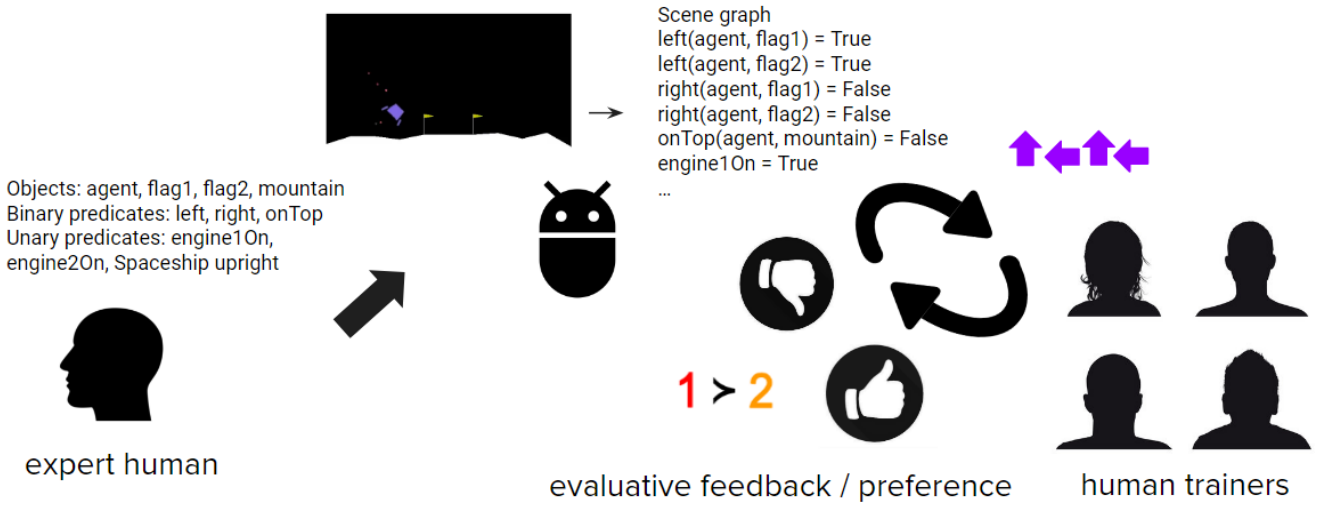

Figure 1. Overview of the proposed dual representation framework. The robot maintain two state representations: one for learning a fine-grained, low-level continuous control policy. The other representation is specified by an expert human in the format of symbolic scene graph. The robot uses this representation to actively communicate with human trainers for evaluation or preference during the training process.

will *ask* humans for their evaluative feedback. The key challenge here is when to ask for such feedback – hypothetically, asking feedback in the right state will lead to better learning results with less human effort.

The key motivation here is that, adopting scene graph representation allow us to use RL methods that are only suited for discrete state representation, such as algorithms developed for multi-armed bandit problems. Recall the Upper Confidence Bound (UCB1) score for each state is defined as:

$$UCB1(s_t, \pi) = V^{\pi}(s_t) + c\sqrt{\frac{2\log t}{N_t(\pi(s_t, a))}} \quad (1)$$

where t is the current timestep and $N_t(a)$ is the number of times we took action a at s_t. We can slightly modify this to estimate the upper confidence bound for abstract states as:

$$UCB1(g_t) = FPE(g_t) + c\sqrt{\frac{2\log t}{N_t(g_t)}} \quad (2)$$

where $N_t(g_t)$ is the number of times a human has given feedback for the abstract state g_t. $FPE(g_t)$ is the average feedback prediction error for all the environment states encountered so far that belongs to g_t:

$$FPE(g_t) = \frac{1}{N_t(g_t)} \sum_{i=0}^{t} 1(s_i \in g_t)||\hat{H}(s_i, a_i) - H(s_i, a_i)||_2^2 \quad (3)$$

Note that we can calculate $H(s, a)$ using the an additional critic head in SAC. UCB1 also uses a constant c to weight the exploitation and exploration terms. Specifically, this becomes a coefficient to the exploration term.

We can calculate running mean of UCB value for each abstract state and rank then them. Then we give feedback at a specific state s_t if its abstract state g_t is one of the first x in the ranking. The consequence is that the agent will actively ask for feedback in abstract states that it is uncertain about how human would evaluate its decisions. This effectively turn active learning from evaluative feedback into a multi-armed bandit problem, in which the agent tries to estimate the average value of human feedback for each abstract state. With the original high-dimensional continuous state space, this is not possible.

4.2. Preference learning with dual representation

Selecting the optimal trajectory length is a challenging problem in preference learning [35]. Shorter trajectories allow humans to provide feedback of high granularity at the cost of more frequent human interactions. [10] used random 1-2 seconds clips to create trajectory segments for query. This results in users' dissatisfaction since there are a number of meaningless comparisons such as trajectory start compare with trajectory end. Our method proposes a way to select query as well as fragments that reduces meaningless comparisons and raises the performance. As shown in the dual representation framework in Fig. 1, for each state s_t, there is a corresponding discrete scene graph state g_t. In a trajectory, the agent often experiences a few scene graph state changes until the task is completed. The key points that need human to provide preference is where a scene graph state change happens. To ensure we are mak-

ing meaningful comparison, our method asks queries which starts from similar stages, i.e., starts with the same scene graph state, and ends with the next scene graph change.

Since our goal is to estimate the reward function of human based on information from human preferences, we assume the reward is linear in some state-action features that are known, as stated in [6], that is, we assume $r(s_t, a_t) = \omega^T \phi(s_t, a_t)$. Thus, we design the reward of a trajectory ξ as:

$$R(\xi) = \sum_{(s,a)\in\xi} r(s,a) = \omega^T \sum_{(s,a)\in\xi} \phi(s,a) = \omega^T \Phi(\xi) \tag{4}$$

Thus, our goal now is to learn the weight ω. We start by collecting demonstrations $\mathcal{D}$ and generating a prior $p(\omega)$ over the true weight ω based on $\mathcal{D}$. With demonstrations collected, we have a set of scene graph state changes. To generate one query Q, we randomly pick one scene graph state change, and then randomly pick two fragments that starts with this scene graph state change. We will learn the weight ω by asking a sequence of queries. Suppose q is the human response, we compute our posterior over ω as:

$$P(\omega|Q,q) \propto P(q|Q,\omega)P(\omega) \tag{5}$$

5. Experiments

We use five continuous control tasks to evaluate proposed algorithms: Luna-Lander-Continuous-v2, Reaching-Simulation, PlacingBall-Simulation, Reaching-Real, PlacingBall-Real. We use Robosuite [36] as the simulation platform for the second and third tasks. The visualizations of these tasks, along with the abstract state for that particular frame represented as a scene graph instance, can be seen in Fig. 2. This section describes each task and their experimental setups.

5.1. Task 1: Lunar-Lander-Continuous-v2

We use OpenAI Gym's Lunar-Lander-Continuous-v2 [7] as shown in Fig. 2. The lander aims to land safely in the middle of the two flags without crashing. The action space has 2 dimensions. The first dimension of an action determines the throttle of the main engine, while the second dimension specifies the throttle of the lateral boosters. The states are defined within a 8-dimensional state space, which includes: the coordinates of the lander in x and y direction, its linear velocities in x and y direction, its angle, its angular velocity, and two booleans that represent whether each leg is in contact with the ground or not. The agent is scored based on how close it is to the landing pad (between two flags), if the lander crashes or not (-100), and if it comes to rest (+100). Additional penalty is also applied when the lander fires the engine.

5.2. Tasks 2 and 4: Reaching

In the Reaching task, the robot's goal is to move it's gripper to a target position marked on the table. At the beginning of each session, the robot's end effector position is initialized randomly around a point in the area surrounded by the red, green, and blue lines in Fig. 2. The point is located 25cm from the target. In each time step, the only motions the robot can take are to move on the xy-plane located above and parallel to the table surface. The height of the end effector above the table is fixed. This motion defines a 2-dimensional action space $\mathcal{A} = \{\delta x, \delta y \in [-0.1, 0.1]\}$. The robot end effector's position on the plane defines the states $\mathcal{S} = [x, y]$. When the robot's end effector moves inside a 10cm by 25cm target placed at the center of the table, a completion reward of +1 is given. The reward is 0 in all other states. In the simulated experiment (Task 2), the robot executes the action at each time step using Robosuite's operational space controller in 2-dimensions. For the physical robot experiment (Task 4), interfacing between the learning algorithm and the Sawyer robot, as well as the control of Sawyer robot are established using the perls2, a library connecting simulated and real world robot learning experiments. Additionally, to prevent the robot from reaching its joint limits, a rectangular boundary is defined to restrict the robot's workspace. Experimental setups in simulation and in the real world are shown in Fig. 2.

5.3. Tasks 3 and 5: PlacingBall

The PlacingBall task mimics disposal of trash into a trashcan, and the goal is to drop a ball through a hoop. We begin with a simplified setup where we assume that the robot starts with the ball in its gripper, and the position of the hoop is fixed and known. Similarly to the Reaching task, the robot's end effector position is randomly initialized. The action space is 4-dimensional: motion in each of the Cartesian direction, and gripper opening and closing ($\mathcal{A} = \{\delta x, \delta y, \delta z, gripper\}$). The states are defined as end effector position in 3-dimensional Cartesian space and the gripper state $\mathcal{S} = \{x, y, z, gripper\}$. When the robot releases the ball into the hoop from above the hoop, a completion reward of +1 is given. If the robot drops the ball outside of the hoop, a negative reward of -1 is given. The controllers used for the simulated (Task 5) and the real world (Task 5) experiments are identical to those used in Tasks 2 and 3, with two additional dimensions in the action space. A rectangular boundary limits the robot's workspace in each of the 3 Cartesian dimensions to avoid joint limits being reached. An additional boundary is defined around the hoop to generate a no-entry zone, preventing collision between the robot and the hoop object. If an unexpected collision occur, the session is terminated. The simulation and real world experiment setups are shown in Fig. 2.

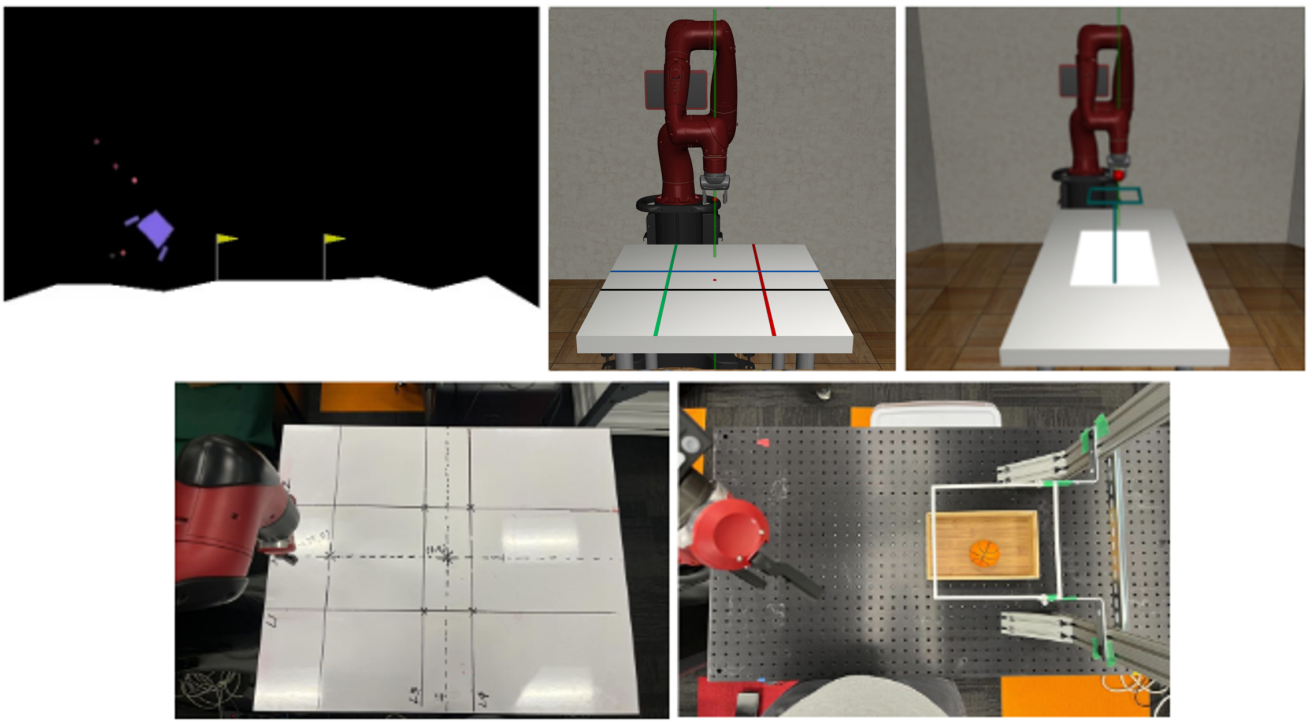

Figure 2. Five continuous control tasks we use for our experiments.

6. Results

Results have been collected for Tasks 1 and 2. Tasks 3, 4, and 5 are in progress, and these experiments will be conducted in the coming weeks.

6.1. Learning from evaluative feedback

We compare our method to the following baselines: SAC (pure RL without feedback), TAMER+RL-100 (asking for feedback at every timestep), TAMER+RL-50 (asking for feedback 50% of the time, uniformly random) and TAMER+RL-25 (25%). The results with synthetic humans are shown in Figs 3 for Luna-Lander-Continuous-v2, and Figs 4 for Reaching-Simulation. From these results, we can see that our algorithm leads to much more efficient learning from human feedback.

6.2. Preference learning

We compare our method with random fragment query, which randomly select the trajectories to query, and segment the query with length equal to the average query length using our method. To evaluate our method, we generate a random ω_{true} as our true weight vector. For Lunar-Lander-Continuous-v2, since the lander aims to land safely in the middle of two flags without crashing, we select a 5-dimensional feature vector containing: distance between lander and the middle of two flags, the velocity of lander, the absolute angular velocity, the left leg contact with ground, the right leg contact with ground. We provide true reward with ω_{true} and the feature vector using formula (4). After each query, we update the posterior belief and calculate the cosine similarity of ω_{est} and ω_{true} as the alignment score, which is:

$$\text{alignment score} = \frac{\omega_{true}^T \omega_{est}}{\|\omega_{true}\| \|\omega_{est}\|} \tag{6}$$

In this experiment, we provide query with an oracle user who knows the true weight ω_{true}. We calculated the alignment scores of 100 queries for both our methods and random fragment query over 5 seeds, and include the result in Fig. 5. We can see our method outperform random fragment especially when asking more queries.

7. Conclusion and Discussion

In this work, we propose the dual representation hypothesis. We show that in continuous control tasks, our proposed learning algorithms based on the dual representation hypothesis can lead to significant improvements in task performance and sample efficiency.

In the context of human-in-the-loop robot training and learning, human representation differ significantly from agent's representation, hence the learning robot needs the "fast system" and its representation to learn the continuous control policy, and the "slow system" to represent the

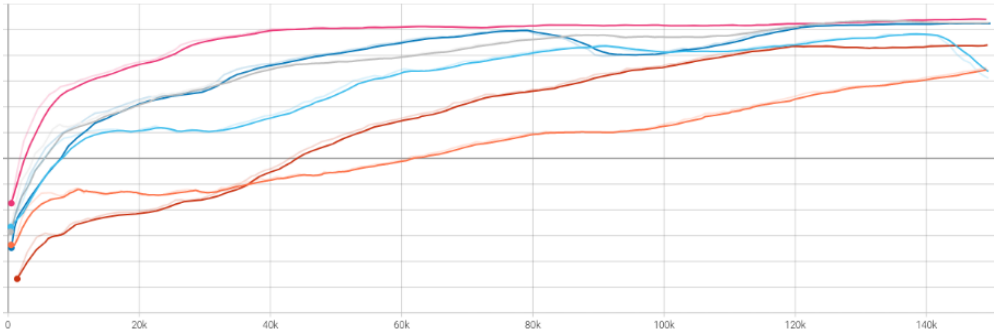

Figure 3. Learning from human evaluative feedback results (learning curves) on Luna-Lander. Pink: corrective feedback, in which the oracle provides the correct action label (performance upperbound). **Gray:** our algorithm with feedback on only **6.27%** states. Dark blue, light blue, orange: TAMER+RL-100, TAMER+RL-50, TAMER+RL-25. Red: SAC. Results are averaged across 5 runs for each algorithm. Our algorithm can achieve comparable performance with 100% feedback, leading to approximately 16X improvement in human feedback sample efficiency.

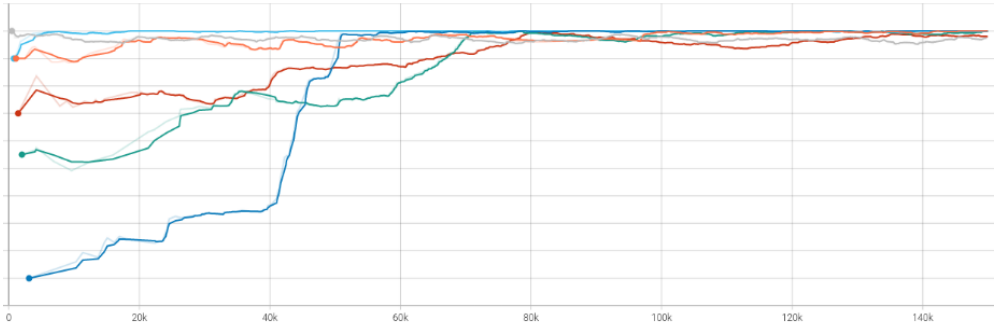

Figure 4. Learning from human evaluative feedback results (learning curves) on Reaching-Simulation. Gray: corrective feedback (upperbound). Orange: ours, with feedback on only **0.86%** states. Light blue, red, green: TAMER+RL-100, TAMER+RL-50, TAMER+RL-25. Dark blue: SAC. Again, our algorithm achieves comparable performance with much less human feedback data.

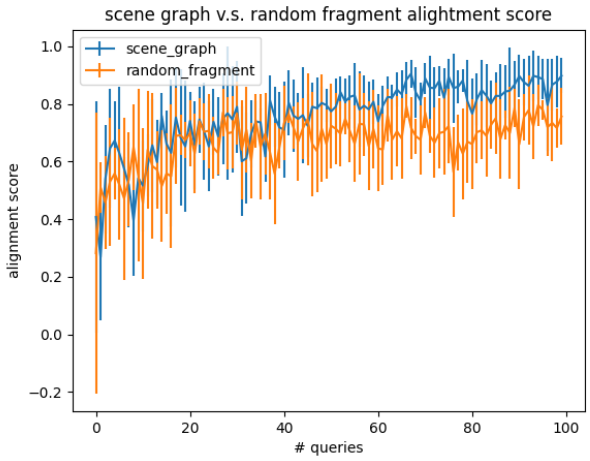

Figure 5. The alignment score of 100 queries for both ours(scene graph) and random fragment over 5 seeds.

mental models of human trainers. We show that the abstract scene graph representation allows us to utilize the algorithms only suitable for that representation, and two representations can work seamlessly together to facilitate learning. In the context of robot learning, abstract representation has long been a research topic. However, adopting it in decision learning often comes with a price: the critical information needed to learn a good policy may be lost in the process of abstraction. Researchers typically avoid this problem by having a hierarchical representation, e.g., as in hierarchical RL [21]. Our design avoids the pitfalls of abstraction by only using it as a mental model of human trainers and a means to facilitate communication with humans.

For our future work, the next step is to continue our unfinished experiments with evaluative feedback in PlacingBall-Simulation, and two real robot tasks, as well as run all our experiments with actual human trainers. Similarly we will continue our experiments with preference learning in Reaching and Placing-Ball-Simulation, and two real robot tasks, as well as run all our experiments with actual human trainers.

References

[1] R. Akrour, M. Schoenauer, M. Sebag, and J.-C. Souplet. Programming by feedback. In *International Conference on Machine Learning (ICML)*, volume 32, pages 1503–1511. JMLR. org, 2014. 2

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 1

[3] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5664–5673, 2019. 2

[4] D. Arumugam, J. K. Lee, S. Saskin, and M. L. Littman. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257*, 2019. 2

[5] A. Bestick, R. Pandya, R. Bajcsy, and A. D. Dragan. Learning human ergonomic preferences for handovers. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. 2

[6] E. Bıyık, A. Talati, and D. Sadigh. Aprel: A library for active preference-based reward learning algorithms. *arXiv preprint arXiv:2108.07259*, 2021. 4

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 4

[8] R. Busa-Fekete, B. Szörényi, P. Weng, W. Cheng, and E. Hüllermeier. Preference-based evolutionary direct policy search. In *ICRA Workshop on Autonomous Learning*, 2013. 2

[9] T. Cederborg, I. Grover, C. L. Isbell, and A. L. Thomaz. Policy shaping with human teachers. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3366–3372. AAAI Press, 2015. 2

[10] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017. 2, 3

[11] Y. Cui and S. Niekum. Active reward learning from critiques. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6907–6914. IEEE, 2018. 2

[12] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1-2):123–156, 2012. 2

[13] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*, pages 2625–2633, 2013. 2

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 2

[15] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization. *arXiv preprint arXiv:2201.13360*, 2022. 2

[16] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3668–3678, 2015. 2

[17] D. Kahneman. *Thinking, fast and slow*. Macmillan, 2011. 1

[18] W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM, 2009. 2

[19] W. B. Knox and P. Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems, 2010. 2

[20] W. B. Knox and P. Stone. Reinforcement learning from simultaneous human and mdp reward. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 475–482. International Foundation for Autonomous Agents and Multiagent Systems, 2012. 2

[21] H. Le, N. Jiang, A. Agarwal, M. Dudik, Y. Yue, and H. Daumé. Hierarchical imitation and reinforcement learning. In *International Conference on Machine Learning*, pages 2923–2932, 2018. 6

[22] J. MacGlashan, M. K. Ho, R. Loftin, B. Peng, G. Wang, D. L. Roberts, M. E. Taylor, and M. L. Littman. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2285–2294. JMLR. org, 2017. 2

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 1

[24] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018. 1

[25] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022. 2

[26] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017. 2

[27] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999. 1

[28] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 1

[29] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2

[30] G. Warnell, N. Waytowich, V. Lawhern, and P. Stone. Deep tamer: Interactive agent shaping in high-dimensional state

spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2

[31] A. Wilson, A. Fern, and P. Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems*, pages 1133–1141, 2012. 2

[32] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz. A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research*, 18(1):4945–4990, 2017. 2

[33] C. Wirth, J. Fürnkranz, and G. Neumann. Model-free preference-based reinforcement learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2222–2228, 2016. 2

[34] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone. Leveraging human guidance for deep reinforcement learning tasks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 6339–6346. AAAI Press, 2019. 1

[35] R. Zhang, F. Torabi, G. Warnell, and P. Stone. Recent advances in leveraging human guidance for sequential decision-making tasks. *Autonomous Agents and Multi-Agent Systems*, 35(2):1–39, 2021. 1, 3

[36] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020. 4