Model-Free Shared Autonomy through Deep Human-in-the-Loop Reinforcement Learning

Siddharth Reddy, Anca Dragan, Sergey Levine
Department of Electrical Engineering and Computer Science
University of California, Berkeley
`{sgr, anca, svlevine}@berkeley.edu`

Abstract

In shared autonomy, user input is combined with semi-autonomous control to achieve a common goal. The goal is unknown ex-ante, so the agent infers the goal from user input and assists with the task. Existing methods assume knowledge of the dynamics of the environment, the user's policy, and the representation of the goal, which severely limits their application to real-world scenarios. We propose a model-free shared autonomy framework that lifts these assumptions. The key idea is using human-in-the-loop reinforcement learning with neural network function approximation to learn an end-to-end mapping from user input to agent action, with task reward as the only form of supervision. Preliminary experiments with a real human ($n = 1$) and synthetic pilots playing the Lunar Lander game showcase the ability of our algorithm to assist users with a real-time control task in which the agent cannot directly access the goal through observations, but receives a reward signal and user input that both depend on the goal. The agent learns to decode goal-related information from user input and complete the task more effectively than the user or agent could on their own. This paper is a proof of concept that illustrates the potential for deep reinforcement learning to enable flexible and practical assistive systems.

1 Introduction

Sequential decision-making under uncertainty is challenging for both humans and machines, but in different ways. Humans are limited by bounded rationality and physical constraints, but excel at sensory perception and modeling the mental states of other humans [28, 30]. Reinforcement learning algorithms fail in messy, partially observable environments, but can learn to control many degrees of freedom simultaneously and execute actions with machine precision [29, 1]. Most real-world tasks, like driving and surgery, require both sophisticated perception and fine-grained control. *Shared autonomy* tackles this problem by combining user inputs with semi-autonomous control to achieve common goals [14]. Existing work in shared autonomy focuses on enabling human-robot collaboration and shared-control teleoperation through a combination of predicting user intent and optimizing task performance [19, 15].

State-of-the-art shared autonomy algorithms are limited to settings with known (1) dynamics, (2) user behavior, and (3) goal representation. For many real-world tasks, these are severely limiting assumptions. (1) Fitting an accurate global dynamics model can be more difficult than learning to perform the task. (2) User input can exhibit systematic errors (e.g., time-inconsistent planning [6] or custom keybindings that scramble action labels) that prevent standard inverse reinforcement learning and goal inference algorithms from recovering user intent by inverting a generative model of behavior. (3) The user's goals may not be representable in a fixed goal space – in laboratory experiments, we can engineer the task to allow the user to specify the goal exactly, e.g., the coordinates of the desired landing site for an aerial drone; but in the real world, the true goal might fall outside a pre-specified set, e.g., flying the drone to the desired location without crashing into moving obstacles.

The primary contribution of this paper is a model-free reinforcement learning algorithm for shared autonomy. Our key insight is that training an end-to-end mapping from environmental observation and user input to agent action with task reward as the only form of supervision can overcome the three limitations of model-based shared autonomy discussed above. From the agent's perspective, the user acts like an additional sensor generating observations from which the agent can implicitly decode the user's goals. From the user's perspective, the agent behaves like an adaptive interface that learns a personalized mapping from user commands to actions that maximizes task reward. In this paper, we present a proof of concept that illustrates how our approach can be applied to a real-time assistive control task: the Lunar Lander game (see Figure 1).

2 Related Work

The closest related prior work that studies shared user-machine control comes from the fields of robotic teleoperation and brain-computer interfaces. The shared-control teleoperation literature explores different ways to incorporate user input into a semi-autonomous system that is initially unaware of the user's goal [10, 5]. Prior work suggests that instead of asking the user to supply a cumbersome, explicit representation of the goal (e.g., the coordinates of an object to be grasped), the system should try to infer the user's goal from intuitive inputs like suggested controls or feedback on previous actions [11]. Intent prediction in prior work is usually accomplished by inverting a generative model of user actions (e.g., maximum entropy inverse optimal control [35]) [19, 15]. By making assumptions about user behavior, model-based intent prediction enables goal inference without an expensive training phase for learning a user model, but reduces the flexibility of the system to learn from different styles of user input. It also fails to directly optimize the true objective: task reward, not goal prediction accuracy. We tackle this problem by using model-free reinforcement learning with neural network function approximation to learn to decode goal-related information that is useful for decision-making from arbitrary user-generated signals, with task reward as the only form of supervision.

Most direct teleoperation interfaces use a carefully hand-coded function to retarget human commands to robot movements, which makes it difficult to personalize the interface to individual operators. It also requires significant engineering effort to craft new retargeting functions for robots with different morphologies. [21] proposes a data-driven motion mapping algorithm for fitting a user model to demonstration queries, which present the human operator with a robotic motion and request an example of the corresponding human commands. This approach is attractive in that it formulates interface adaptation as a supervised learning problem, but suffers from the need to either hand-engineer a set of demonstration queries or rely on an active learning algorithm with a query complexity that scales with the dimensionality of the action space but not the difficulty of the task. The core problem is that instead of maximizing task reward, this method maximizes intended-action prediction accuracy.

A large body of work in brain-machine interfaces uses optimal control and reinforcement learning for closed-loop decoder adaptation [27] and learning prosthetic limb controllers that respond to neural control signals from myoelectric sensors [22]. These algorithms track desired motion (e.g., the position of a robotic arm end affector) using instantaneous snapshots of EMG signals. Our problem setting differs in that the agent has access to observations generated by the environment in addition to those generated by the user, providing the necessary context to interpret user input. Additionally, our setting motivates the use of temporal context (e.g., a recurrent policy that remembers previous user inputs) to infer stable, long-term goals from noisy and intermittent user inputs.

Popular human-in-the-loop reinforcement learning frameworks like TAMER [32, 16] and Arbiter [17] reduce the sample complexity of modern deep reinforcement learning techniques by relying on a human supervisor for reward shaping and guided exploration. These frameworks are applicable to settings where the agent has access to all task-relevant information (e.g., goals), but training is too slow for practical applications. We focus on the orthogonal setting where the agent does not have direct access to the goal, and will always need to leverage user input to accomplish the task; even after training.

Similar ideas for building adaptive human-computer interfaces and sharing autonomy have been explored in computer graphics (e.g., using structured geometric models and unsupervised learning on demonstrations to animate virtual characters using motion capture data from humans [4]), formal methods [26], and natural language processing (e.g., learning to act on natural language instructions from humans [31, 2]).

3 Background

In this section, we briefly recap the reinforcement learning and shared autonomy problem statements.

3.1 Reinforcement Learning

Consider a Markov decision process (MDP) with a set of states $\mathcal{S}$, actions $\mathcal{A}$, transition distribution $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1]$. The expected future discounted return of taking action a in state s with policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is expressed by the state-action value function:

$$Q^\pi(s, a) = \mathbb{E}_{s_0,a_0,\ldots} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid \pi, s_0 = s, a_0 = a \right]$$

The agent's goal is to find a policy π^* that maximizes expected future discounted return, or equivalently, satisfies the Bellman equation [29]:

$$Q^*(s, a) = R(s, a) + \gamma \mathbb{E}_{s' \sim T(\cdot|s,a)} \left[\max_{a' \in \mathcal{A}} Q^*(s', a') \right]$$

In this paper, we use Q-learning [33] to train an assistive agent to help a human with a real-time control task.

3.2 Shared Autonomy

In our shared autonomy setting, the reward function depends on a goal $g \in \mathcal{G}$ that is initially known to the user and unknown to the agent. Prior work assumes that the goal space $\mathcal{G}$ is a discrete set known to the agent. The user follows a goal-conditioned policy $\pi_h : \mathcal{S} \times \mathcal{G} \times \mathcal{H} \to [0, 1]$ known to the agent, where $\mathcal{H}$ is the space of possible user inputs – if the user suggests actions, then $\mathcal{H} = \mathcal{A}$. The transition distribution T is also known to the agent. The agent's uncertainty in the goal can be formalized as a partially-observable Markov decision process (POMDP) identical to the MDP described in Section 3.1 with the following modifications: the state space $\tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{G}$ is augmented with the goal, the transition distribution $\tilde{T}((s_{t+1}, g) \mid s_t, g, a_t) = T(s_{t+1} \mid s_t, a_t)$ maintains a constant goal, and the observation distribution $O(s, a^h \mid s, g) = \pi_h(a^h \mid s, g)$ is given by the user policy where $a^h \in \mathcal{H}$ is the user input. Computing the optimal policy for the POMDP specified by the tuple $(\tilde{\mathcal{S}}, \mathcal{A}, \tilde{T}, \tilde{R}, \mathcal{H}, O)$ is intractable, so approximate methods like hindsight optimization are used in practice [15]. We improve upon these methods by relaxing the assumptions that $\mathcal{G}$, T, and π_h are known ex-ante.

4 Model-Free Shared Autonomy

In this section, we introduce a deep human-in-the-loop reinforcement learning algorithm for implementing model-free shared autonomy.

4.1 Incorporating User Input

Our formulation of shared autonomy is identical to the one described in Section 3.2, minus the assumptions that $\mathcal{G}$, T, and π_h are known ex-ante. In our preliminary experiments, we assume that the user input is a suggested action (i.e., $\mathcal{H} = \mathcal{A}$) and use a straightforward scheme to incorporate user input into the agent's policy: jointly embed the usual observation s_t with the user input a^h by concatenating them:

$$\tilde{s}_t = \begin{bmatrix} s_t \\ a_t^h \end{bmatrix}$$

Although this setup benefits from simplicity and ease of implementation, richer and more expressive joint embedding models like multiplicative integrations and bilinear pooling might improve agent performance [9, 34, 7].

4.2 Learning to Assist

Training an assistive agent with a human in the loop poses two challenges: (1) maintaining user comfort in order to preserve input quality, and (2) minimizing the number of interactions with the environment in order to prevent the speed of user input from becoming a significant bottleneck. In our experiments, we use deep Q-learning [33] to learn an approximate state-action value function that can be used to select and evaluate actions. Specifically, we implement neural fitted Q-iteration (NFQI) [23], which gets around a practical issue with using vanilla deep Q-networks (DQN) [18] for human-in-the-loop learning: DQN performs a gradient update after each step, which causes the simulator to lag and disrupts human gameplay, whereas NFQI performs all gradient updates at the end of each episode. We choose Q-learning because (1) it is an off-policy algorithm, so we can maintain user comfort by implementing a behavior policy that explicitly trades off control between the user and the agent without changing the agent's function approximation architecture, and (2) bootstrapping via temporal-difference learning tends to be more sample-efficient than policy gradient and Monte Carlo value-based methods.

Control sharing. If the user input is a suggested control, consistently ignoring the suggestion and taking a different action can degrade the quality of user input, since humans rely on feedback from their actions to perform real-time control tasks[1]. To avoid this, we use the following behavior policy to select actions during Q-learning:

$$\pi(a \mid \tilde{s}, a^h) = \delta\left(a = \underset{\{a : Q'(\tilde{s},a) \geq (1-\alpha) Q'(\tilde{s},a^*)\}}{\arg\max} f(a, a^h)\right)$$

where f is an action-similarity function and $Q'(\tilde{s}, a) = Q(\tilde{s}, a) - \min_{a' \in \mathcal{A}} Q(\tilde{s}, a')$ maintains a sane comparison for negative Q values – if $Q(\tilde{s}, a) < 0 \; \forall a$ and $0 < \alpha < 1$, then the set of feasible actions would be empty if we didn't subtract a baseline from the Q values. The intuition behind this behavior policy π is that the system should select a feasible action closest to the user's suggestion, where an action is feasible if it isn't that much worse than the optimal action. The constant $\alpha \in [0, 1]$ is a hyperparameter that controls the tolerance of the system to suboptimal human suggestions, or equivalently, the degree of user assistance.

Beyond maintaining user comfort, control sharing may also provide a more natural exploration mechanism than standard approaches for Q-learning like ϵ-greedy action selection.

5 Experiments

Task. Our experiments aim to answer the following research question: can our model-free shared autonomy framework combine human and machine intelligence to accomplish a task that neither could on their own, when the environment dynamics, user behavior, and goal representation are initially unknown? As a preliminary step in this direction, we pick a relatively simple task with the same key ingredients as a real-world semi-autonomous control problem: the Lunar Lander game in OpenAI Gym [3] (see Figure 1). The objective of the game is to pilot the lunar lander vehicle to a specified landing site on the ground without crashing using two lateral thrusters and a main engine. Each episode lasts at most 1000 steps, and runs at 50 frames per second. An episode ends when the lander crashes, flies out of bounds, remains stationary on the ground, or time runs out. The action space $\mathcal{A}$ consists of six discrete actions that correspond to the {left, right, off} steering commands

[1]The same issue arises when running DAgger [24] with a human expert in the loop labeling states with actions.

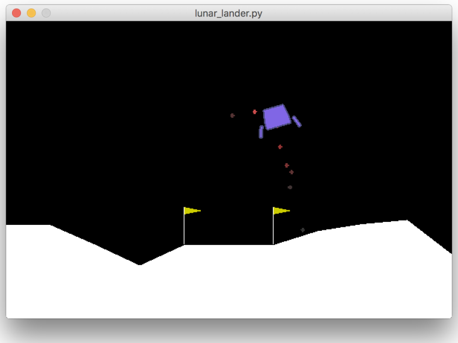

Figure 1: A screenshot of the Lunar Lander game environment. The terrain, including the location of the landing pad, is randomized at the beginning of each episode.

Table 1: Evaluation of pilot-copilot teams playing Lunar Lander. Rewards are shown with their standard error on ten different random seeds and the last 100 episodes of copilot training for teams with a copilot; on 100 episodes for teams without a copilot.

	Without Copilot			With Copilot		
Pilot	Reward	Crash Rate	Success Rate	Reward	Crash Rate	Success Rate
None	-178 ± 28	1.00	0.00	-21 ± 31	0.23	0.14
Noisy	-67 ± 43	0.95	0.02	35 ± 3	0.53	0.27
Laggy	-132 ± 112	0.87	0.12	76 ± 2	0.42	0.43
Human	126 ± 17	0.30	0.70	88 ± 71	0.38	0.41

and {on, off} main engine settings. The state $s \in \mathbb{R}^8$ is an eight-dimensional vector that encodes the lander's position, velocity, angle, angular velocity, and indicators for contact between the legs of the vehicle and the ground. The reward function is shaped to penalize distance from the landing site, speed, and tilt. At the end of the episode, the agent gets a large reward bonus if it successfully lands at the site, or a large negative reward if it crashes or goes out of bounds. The x-coordinate of the landing site is selected uniformly at random at the beginning of each episode, and is not directly accessible to the agent through the state s. A human playing the game can see two flags demarcating the landing site, and can supply a suggested control $a^h \in \mathcal{A}$ that implicitly encodes the location or relative direction of the landing site. Thus, in order to accomplish the task, the agent needs to leverage a^h to maneuver toward the landing site.

The agent uses a multi-layer perceptron with two hidden layers of 64 units each to approximate the Q function $\hat{Q} : \mathcal{S} \times \mathcal{A}^2 \rightarrow \mathbb{R}$. The action-similarity function $f(a, a^h)$ in the agent's behavior policy counts the number of dimensions in which actions a and a^h agree (e.g., $f((\text{left}, \text{on}), (\text{left}, \text{off})) = 1$).

Users. To simplify terminology, we henceforth refer to the user as the *pilot* and the semi-autonomous agent as the *copilot*. In our experiments, we use both a real human pilot (the first author) as well as simulated pilots. We simulate a pilot using an agent called LaggyPilot, which is trained as follows: augment the state vector with the landing site coordinates, train the agent using vanilla DQN, and corrupt the trained policy by forcing it to repeat the previously-executed action with fixed probability $p = 0.85$. This causes each action to repeat for a number of steps that follows a geometric distribution. We also examined another synthetic pilot model, NoisyPilot, which uses the same training procedure as LaggyPilot but follows an ϵ-greedy behavior policy at test time ($\epsilon = 0.3$). We find that LaggyPilot qualitatively matches the control style of a real human better than NoisyPilot, most likely due to the human expert being approximately optimal and the environment frame rate exceeding human reaction time.

Results. The results in Figure 2 show that a copilot which leverages input from LaggyPilot outperforms the solo LaggyPilot and solo copilot: the combined pilot-copilot team crashes and goes out of bounds less often, uses less fuel, follows stabler trajectories, and finds the landing site

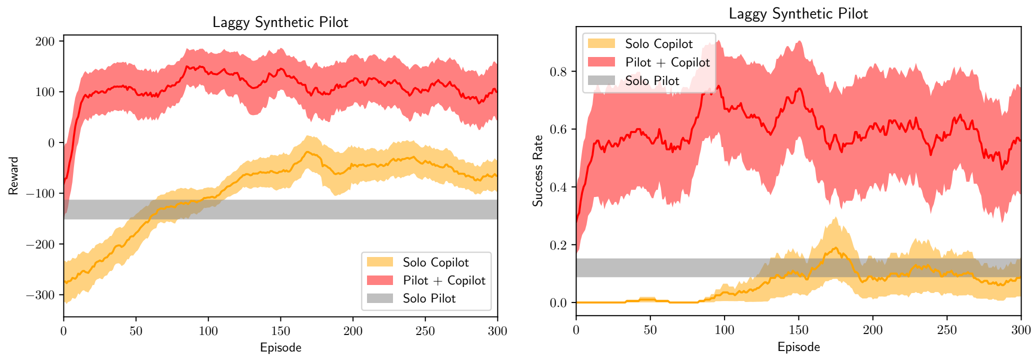

Figure 2: A copilot that leverages input from the synthetic LAGGYPILOT dramatically outperforms the solo LAGGYPILOT and solo copilot. The colored bands illustrate the standard error of rewards and success rates for ten different random seeds. Rewards and success rates are smoothed using a moving average with a window size of 20 episodes. Success is defined as safely landing at the site.

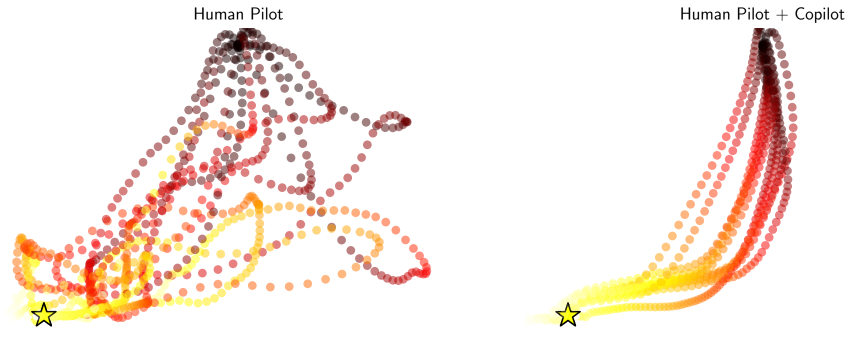

Figure 3: A sample of trajectories followed by a real human pilot with (right) and without (left) a copilot. The beginning of a trajectory is colored black, and follows a gradient to yellow over time. The landing pad is marked by a star. For the sake of illustration, we only show trajectories for one particular landing site (which is selected uniformly at random at the beginning of each episode).

more often than the other two solo teams. The solo copilot and combined pilot-copilot teams learn from experience, whereas the solo LAGGYPILOT team is pretrained and frozen; hence the stationarity of the gray curve. For the combined pilot-copilot team, we set $\alpha = 0.75$, which is heuristically the smallest tolerance value such that a real human pilot still feels like they are in control and supplies useful inputs.

The results in Table 1 show that while NOISYPILOT also performs better with a copilot than without a copilot, a real human pilot actually performs better without a copilot than with a copilot in terms of reward, crash rate, and success rate. These summary metrics do not tell the whole story: Figure 3 shows that there is a clear qualitative benefit to combining a real human pilot with a copilot. Although the human is able to land successfully most of the time, they follow a tortuous path with sudden drops and difficult course corrections. With a copilot, the human follows a smooth, gradual descent to the landing site, but occasionally takes too long to make the journey (i.e., the episode ends before they actually land), hence the smaller reward.

Code and data for reproducing these results are publicly available at `https://github.com/rddy/deepassist`.

6 Discussion

6.1 Summary

In this paper, we develop the first shared autonomy framework for tasks in which the environment dynamics, user policy, and goal representation are unknown. Our primary contribution is an algorithm that uses model-free reinforcement learning with neural network function approximation to learn an end-to-end mapping from environmental observation and user input to agent action, with task reward as the only form of supervision.

Preliminary experiments with one real huma pilot and simulated pilots playing the Lunar Lander game, a real-time control task with the ingredients of a real-world shared-control teleoperation problem, hint that the algorithm is capable of learning to assist users who exhibit different types of suboptimality, enabling user-machine teams to outperform solo users and machines in quantitative and qualitative metrics.

6.2 Limitations and Future Work

Several open questions remain to be addressed:

Known goal representation. When the goal representation is known but the environment dynamics and user model remain unknown, would it make more sense to separately train a goal decoder using supervised learning from demonstrations and a goal-conditioned policy using a universal value function approximator [25] then plug them together, instead of using the end-to-end approach described in this paper?

Temporal context. To infer stable, long-term goals from noisy or intermittent user input, the copilot needs to remember past observations and user inputs. The most straightforward way to accomplish this is to concatenate m previous frames together with the current observation, possibly with a frame-skip to increase the backward-looking horizon of the previous frames. More sophisticated modifications to the copilot architecture could involve adding recurrent connections to the copilot policy or a temporal convolution over previous frames.

Stochastic policies. In this paper, we implemented a copilot with a deterministic policy, but there are several reasons to want a stochastic policy, including robustness to uncertain dynamics and achieving multimodal objectives [12]. A control sharing mechanism that conservatively fine-tunes the pilot policy using KL-control [13] would enable the pilot to share autonomy with a stochastic copilot. We have some preliminary results in this direction: we implemented G-learning [8] with a simple kernel density estimation technique that smooths the one-hot encoding of the discrete action suggested by the pilot by adding ϵ to all dimensions with zeros and re-normalizing; equivalent to imposing a uniform Dirichlet prior on the policy at the current state, treating the pilot input as a sample from a categorical distribution over actions, and supplying the resulting posterior distribution over actions to the copilot. A heuristic that decays the smoothing constant ϵ exponentially with the average temporal-difference error for the current episode in Q-learning performed best, but required tuning three hyperparameters that govern the decay schedule of ϵ over training episodes. Further work is needed to understand how to mix the pilot and copilot policies in a more principled way.

Mutual adaptation. Explicitly modeling the mutual adaptation process, in which the human adapts their policy to an interface while the interface simultaneously adapts itself to the human, may improve the speed of human-in-the-loop copilot training and inform theoretical guarantees on the convergence of our algorithm [20].

7 Acknowledgements

We would like to thank Oleg Klimov for developing and releasing the Lunar Lander game environment in OpenAI Gym. This work was supported in part by a Berkeley EECS Department Fellowship for first-year Ph.D. students.

References

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[2] Satchuthananthavale RK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 82–90. Association for Computational Linguistics, 2009.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[4] Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 409–416. ACM, 2003.

[5] Anca D Dragan and Siddhartha S Srinivasa. A policy-blending formalism for shared control. *The International Journal of Robotics Research*, 32(7):790–805, 2013.

[6] Owain Evans, Andreas Stuhlmüller, and Noah D Goodman. Learning the preferences of ignorant, inconsistent agents. In *AAAI*, pages 323–329, 2016.

[7] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[8] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

[9] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*, 2016.

[10] Ray C Goertz. Manipulators used for handling radioactive materials. *Human factors in technology*, pages 425–443, 1963.

[11] Scott A Green, Mark Billinghurst, XiaoQi Chen, and J Geoffrey Chase. Human-robot collaboration: A literature review and augmented reality approach in design. *International Journal of Advanced Robotic Systems*, 5(1):1, 2008.

[12] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

[13] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, pages 1645–1654, 2017.

[14] Shervin Javdani. Acting under uncertainty for information gathering and shared autonomy. 2017.

[15] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization. *arXiv preprint arXiv:1503.07619*, 2015.

[16] W Bradley Knox and Peter 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.

[17] Zhiyu Lin, Brent Harrison, Aaron Keech, and Mark O Riedl. Explore, exploit or listen: Combining human feedback and policy model to speed up deep reinforcement learning in 3d worlds. *arXiv preprint arXiv:1709.03969*, 2017.

[18] 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):529–533, 2015.

[19] Katharina Muelling, Arun Venkatraman, Jean-Sebastien Valois, John E Downey, Jeffrey Weiss, Shervin Javdani, Martial Hebert, Andrew B Schwartz, Jennifer L Collinger, and J Andrew Bagnell. Autonomy infused teleoperation with application to brain computer interface controlled manipulation. *Autonomous Robots*, pages 1–22, 2017.

[20] Stefanos Nikolaidis, Yu Xiang Zhu, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in shared autonomy. *arXiv preprint arXiv:1701.07851*, 2017.

[21] Rebecca M. Pierce and Katherine J. Kuchenbecker. A data-driven method for determining natural human-robot motion mappings in teleoperation. In *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 169–176, 2012.

[22] Patrick M Pilarski, Michael R Dawson, Thomas Degris, Farbod Fahimi, Jason P Carey, and Richard S Sutton. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1–7. IEEE, 2011.

[23] Martin Riedmiller. Neural fitted q iteration-first experiences with a data efficient neural reinforcement learning method. In *ECML*, volume 3720, pages 317–328. Springer, 2005.

[24] Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[25] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1312–1320, 2015.

[26] Sanjit A Seshia, Dorsa Sadigh, and S Shankar Sastry. Formal methods for semi-autonomous driving. In *Proceedings of the 52nd Annual Design Automation Conference*, page 148. ACM, 2015.

[27] Maryam M Shanechi, Amy L Orsborn, and Jose M Carmena. Robust brain-machine interface design using optimal feedback control modeling and adaptive point process filtering. *PLoS computational biology*, 12(4):e1004730, 2016.

[28] Herbert A Simon. Theories of bounded rationality. *Decision and organization*, 1(1):161–176, 1972.

[29] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[30] Tomer Ullman, Chris Baker, Owen Macindoe, Owain Evans, Noah Goodman, and Joshua B Tenenbaum. Help or hinder: Bayesian models of social goal inference. In *Advances in neural information processing systems*, pages 1874–1882, 2009.

[31] Sida I Wang, Percy Liang, and Christopher D Manning. Learning Language Games through Interaction.

[32] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. *arXiv preprint arXiv:1709.10163*, 2017.

[33] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[34] Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 2856–2864, 2016.

[35] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.