# Policy Shaping in Continuous State Spaces

**Thomas Wei, Taylor A. Kessler Faulkner, Andrea L. Thomaz**

## Abstract

Human-in-the-loop Reinforcement Learning (HRL) adds human feedback to RL algorithms. Policy Shaping [Griffith *et al.*, 2013], an HRL algorithm, uses human feedback to influence a learning agent's policy. We extend this work to continuous, high-dimensional state spaces using Deep Reinforcement Learning (RL) and test in simulation. Our algorithm, Deep Policy Shaping (DPS), learns a generalized feedback neural network that models the ground truth optimality of binary feedback as a latent random variable and combines that output with any off-policy RL algorithm. In simulated experiments, we find that DPS outperforms or matches baselines on average over multiple hyperparameter settings and varying levels of feedback correctness.

## 1 Introduction

Reinforcement learning has become increasingly effective in continuous and high dimensional state spaces by leveraging deep neural networks [Mnih *et al.*, 2015]. The addition of human teachers to the reinforcement learning process allows for increased sample efficiency. Many methods of using human feedback to assist in the reinforcement learning process have also made the leap to large state spaces [Warnell *et al.*, 2018; Xiao *et al.*, 2020; Arakawa *et al.*, 2018; Arumugam *et al.*, 2019]. However, many of these methods either only use human feedback without a reward function, use the feedback to shape the reward, or do not consider the correctness of feedback. Teachers may be able to give correct feedback to the robot throughout a majority of a task, but if a percentage of their feedback is wrong it can negatively effect the robot's learning. In the worst case, it can lead the robot to learn the task incorrectly. Policy Shaping [Griffith *et al.*, 2013] uses potentially noisy feedback to inform the policy, rather than reward function, of an RL agent in countable state spaces. In this work we extend Policy Shaping to continuous and high dimensional state spaces.

Some methods of HRL interpret human feedback as proportional to a human-defined reward function [Warnell *et al.*, 2018; Xiao *et al.*, 2020; Arakawa *et al.*, 2018] or as a measure of how much the policy deviates from the policy desired by the user [Arumugam *et al.*, 2019]. We introduce Deep Policy Shaping (DPS), a method that combines information from a feedback-generalizing neural network with any off-policy RL algorithm. DPS learns concurrently from feedback and environmental rewards (Figure 1). To incorporate the consistency of the human's feedback, we model the ground truth optimality as a latent random variable that determines the distribution for human feedback for a particular state-action pair. This allows us to use techniques from research on deep learning with noisy labels, specifically modifying the loss function to reflect our confidence in the feedback given by the user [Mnih and Hinton, 2012].

We chose neural networks to generalize the insight gained from feedback because of their ability to learn task-specific feature extraction from data. This is related to the idea of us using feedback in the first place in that it is easier for a non-expert end user of DPS to give feedback rather than design a reward function. Similarly, when learning from images, or comparably high dimensional state spaces, many "shallow" generalization methods require manually defined feature extraction to be performed on the state beforehand. In the case of images, this requires significant computer vision knowledge and experience while other state spaces require analogous expertise.

We compare DPS to baselines on the OpenAI [Brockman *et al.*, 2016] and Mujoco's [Todorov *et al.*, 2012] Reacher-v2 environment, varying the percentage of feedback labels that are correct. As DPS is designed to work with human teachers, optimizing hyperparameter settings prior to learning may be impossible. We find that DPS outperforms or matches the baselines, particularly when hyperparameter settings are not tuned.

## 2 Background

Some prior work in the field of HRL replaces the reward function with human feedback [Knox and Stone, 2009; Warnell *et al.*, 2018], or uses human input to shape an existing environmental reward function [Xiao *et al.*, 2020]. The TAMER framework [Knox and Stone, 2009], expanded to deep learning in [Warnell *et al.*, 2018], learns a human reward function from feedback, which the agent then learns from in place of an environmental reward function. The FRESH algorithm similarly learns a human reward function from feedback, and an agent learns from the sum of the scaled, estimated human

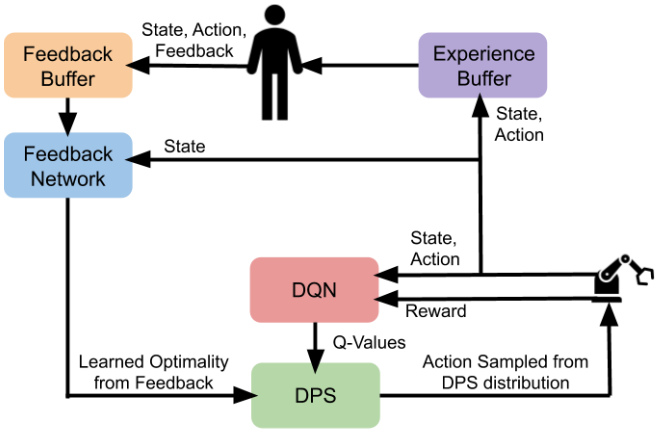


Figure 1: Deep Policy Shaping Pipeline.

reward and the environment reward [Xiao *et al.*, 2020]. Differing from both, DPS interprets human input as feedback on the policy, rather than a reward.

There is existing prior work on interpreting human input as policy feedback. DPS is directly inspired by the original Policy Shaping algorithm, which enables efficient RL in discrete spaces using human feedback [Griffith *et al.*, 2013]. COACH [MacGlashan *et al.*, 2017] and Deep COACH [Arumugam *et al.*, 2019] both use the similarity of human feedback to the advantage function of an MDP. This similarity is applied directly to an actor-critic method, with human feedback taking the place of advantage in the gradient update. Deep COACH, unlike DPS, does not integrate environment feedback and rather only learns the human desired policy, regardless of optimality. One of the methods of integrating TAMER signal with RL tested in [Knox and Stone, 2010] was extended to Deep TAMER and a Deep Q Network by [Arakawa *et al.*, 2018]. Like DPS, their approach modified the policy of the RL agent; however, it differs in the loss function used to generalize feedback and how it integrated said generalized feedback with the RL agent.

There has already been considerable research done into learning from noisy labels. [Mnih and Hinton, 2012] showed a neural net trained with a loss function that modeled noise as an asymmetric Bernoulli outperformed a neural net trained with binary cross entropy loss in the classification of pixels from aerial images. Their approach relied on *a priori* information on the probability of label flip noise. There is previous work on learning from noisy labels in RL [Lin *et al.*, 2017; Sridharan, 2011; Kurenkov *et al.*, 2019]. [Sridharan, 2011] utilized bootstrapping and an ensemble of policies to estimate the consistency of a human-reward-defined policy with the optimal policy and weight its influence appropriately. [Lin *et al.*, 2017] utilize online updates to the probability of listening to advice given by a potentially incorrect teacher, also based off of consistency with actions learned by the RL agent to that point. Neither [Sridharan, 2011] nor [Lin *et al.*, 2017] consider the issue of generalizing the human feedback received to use in states where the human has not provided feedback before as DPS does.

# 3 Method

## 3.1 Policy Shaping

Policy Shaping arose from the observations that humans give feedback with the intention of guiding behavior [Thomaz and Breazeal, 2008] and that integrating human feedback with RL methods via influencing behavior outperforms other forms of integration [Knox and Stone, 2010]. Unlike prior methods of integrating human feedback with RL models, Policy Shaping interprets feedback as a statement on the optimality of an action in a specific state [Griffith *et al.*, 2013]. It then uses that interpretation to compute the probability of the optimality of a state-action pair from feedback. Assuming a uniform prior on the optimality of actions, the probability is computed as follows:

$$P(a \text{ is optimal}|s, a, D) = \frac{C^{\Delta_{s,a}}}{C^{\Delta_{s,a}} + (1-C)^{\Delta_{s,a}}} \qquad (1)$$

where $s, a$ are the state and action respectively, $C$ is the consistency of the feedback with the true optimality of $a$ in $s$, and $\Delta_{s,a}$ is the difference between the positive and negative feedback given on action $a$ in state $s$ and $D$ is all the feedback received by the agent. This distribution is then multiplied together with the distribution created by the RL agent in order to shape exploration, since multiplying distributions is the Bayes optimal method of combining information from conditionally independent sources [Griffith *et al.*, 2013]. Note, the computing the probability of optimality requires feedback to have been given for that state action pair, otherwise the equation reduces back to the uniform prior. However, in continuous and high dimensional state spaces, multiple instances of feedback being given in the same state is unlikely at best. This motivates Deep Policy Shaping.

## 3.2 Feedback Generalization

We use a neural network to approximate the conditional probability of an action's optimality in a state given the user's feedback. The state is given as input to the network. Since we assume any number of actions can be optimal in a given state, we treat the problem as a multi-label classification problem and use a different sigmoid head to represent $P(a \text{ is optimal}|s, a, D)$ for each action.

### Deep Ensembles with Shared Parameters

Because Policy Shaping uses its estimate of the optimality of an action to scale the distribution with which a RL algorithm explores the state space, it is important that the feedback neural network accurately represents areas it is uncertain about. [Lakshminarayanan *et al.*, 2016] showed that deep ensembles increase the quality of uncertainty in prediction, outperforming Monte Carlo dropout. To save on computation, we utilize a shared base for our ensemble. [Lee *et al.*, 2015] showed that TreeNets, as they dubbed them, performed as well as full ensembles in terms of accuracy. FRESH also utilizes a neural net with multiple heads sharing a base [Xiao *et al.*, 2020].

### Loss Function Modification

In order to account for *a priori* information on the consistency of the human's feedback with the true optimality of an action in a given state, we model the ground truth optimality

| P | DPS | TAMER | DQN | DPS vs DQN | TAMER vs DQN | DPS vs TAMER | ANOVA |
|---|---|---|---|---|---|---|---|
| 1 | **15763951.1** | 14200672.6 | 13547898.6 | $\mathbf{p < 0.005}$ | $p = 0.074244$ | $\mathbf{p < 0.005}$ | $\mathbf{p < 0.001}$ |
| 0.75 | **14139224.6** | 13625897.1 | 13547898.6 | $p = 0.068316$ | $p = 0.899995$ | $p = 0.129681$ | $p = 0.054321$ |
| 0.6 | 13334046.4 | 12713258.9 | **13547898.6** | $p = 0.697596$ | $\mathbf{p < 0.01}$ | $p = 0.064097$ | $\mathbf{p < 0.01}$ |

Table 1: The mean, averaged over 100 random hyperparameter settings, area under the curve for each algorithm at the indicated level of feedback consistency, with $p$-values for the ANOVA and post-hoc Tukey HSD tests. TAMER refers to DQN-TAMER.

as a latent random variable and derive the likelihood that our estimate of the probability of optimality produced the ground truth at that point. This allows us to construct a loss function that accounts for the noise in the human's feedback.

We model the feedback $h$ as a random variable that is consistent with the true optimality with probability $C$. Here $y$ is the probability that the action we are concerned with is optimal in the state given. From our formulation of the problem, we can derive the likelihood of $y$ given $h$.

$$L(y|h) = yC^h(1-C)^{1-h} + (1-y)C^{1-h}(1-C)^h \quad (2)$$

As with standard binary cross entropy loss, we can take the negative of the log-likelihood to use as our loss function for optimization.

$$\mathcal{L}(y,h) = -\log\left(L(y|h)\right) \quad (3)$$

We can see that the derived likelihood and loss function we derived has some properties that we would expect it to have: it reduces to normal cross entropy loss as $C \rightarrow 1$, it reduces to cross entropy loss with the labels flipped as $C \rightarrow 0$, and matches the original Policy Shaping in the discrete case. The first two can be seen just by taking the limit of $\mathcal{L}(y,h)$ as $C$ goes to 1 or 0 respectively. The third comes from the fact that we can also interpret $L(y|h)$ as $P(h|y)$. We can then compute $P(h|y=1)$ and $P(h|y=0)$. From that point the same derivation for the probability of optimality of an action used in discrete Policy Shaping applies.

### 3.3 Integration with Reinforcement Learning

Deep Policy Shaping is meant to shape the exploration of any off-policy RL algorithm. For all the experiments we utilize a Deep Q Network (DQN) for our RL component [Hasselt *et al.*, 2016] and Boltzmann exploration for our exploration function. The hyperparameters for the DQN also remain consistent throughout our experiments with a learning rate of $0.0001$, target network update every 2500 time steps, and the Boltzmann exploration temperature decaying with time according to the following equation:

$$T(t) = 0.05 + (100 - 0.05)e^{\frac{-t}{25000}} \quad (4)$$

Even with the consistency modification to the loss function, our neural network still risks overfitting the potentially noisy feedback. To avoid adverse effects, we reduce the effect that the distribution has on the outcome over time by clamping the outputs of the distribution to within the range $[0.5-\beta, 0.5+\beta]$ where $\beta(t) = 0.5 * \alpha^t$ and $\alpha$ is a hyperparameter from $(0,1)$ which controls the rate of reduction. Like discrete Policy Shaping, DPS combines the distribution over actions created by the RL algorithm with the estimate of the optimality distribution by multiplying them together [Griffith *et al.*, 2013]:

$$\pi \propto \pi_R \times \pi_F \quad (5)$$

Where $\pi_R$ is the policy distribution computed from the Q-values and exploration function of the DQN agent and $\pi_F$ are the probabilities of optimality for each of the actions computed from the feedback network. Since the only element of the RL algorithm that DPS uses is the action distribution, it can be applied to any off-policy RL algorithm that produces an action distribution. Once the distributions are multiplied together and normalized, the action is sampled from the resulting distribution.

### 3.4 Experimental Design

We use OpenAI [Brockman *et al.*, 2016] and Mujoco's [Todorov *et al.*, 2012] Reacher-v2 environment to evaluate DPS, as shown in Figure 2. Since our neural network design limits us to discrete action spaces, we discretize the state space to consist of 5 actions: idle, center joint clockwise, center joint counter clockwise, elbow joint clockwise, and elbow joint counter clockwise. Aside from idle, these actions all apply an equal amount of torque to their respective joint, accelerating or decelerating the arm in their respective directions. We removed the penalty for high torque actions, as done by [Christiano *et al.*, 2017], since all actions except idle used the same amount of torque, and torque is difficult for users to interpret when providing feedback. We also use a sparse reward signal, with the robot receiving rewards of 1 when it is within 0.015 of the goal, and rewards of zero otherwise. Each episode ends after 50 actions. We compare to DQN-TAMER [Arakawa *et al.*, 2018], the closest method to DPS to our knowledge, and a DQN.

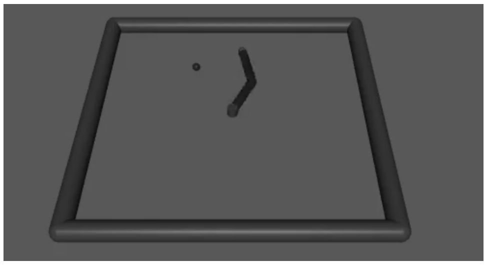


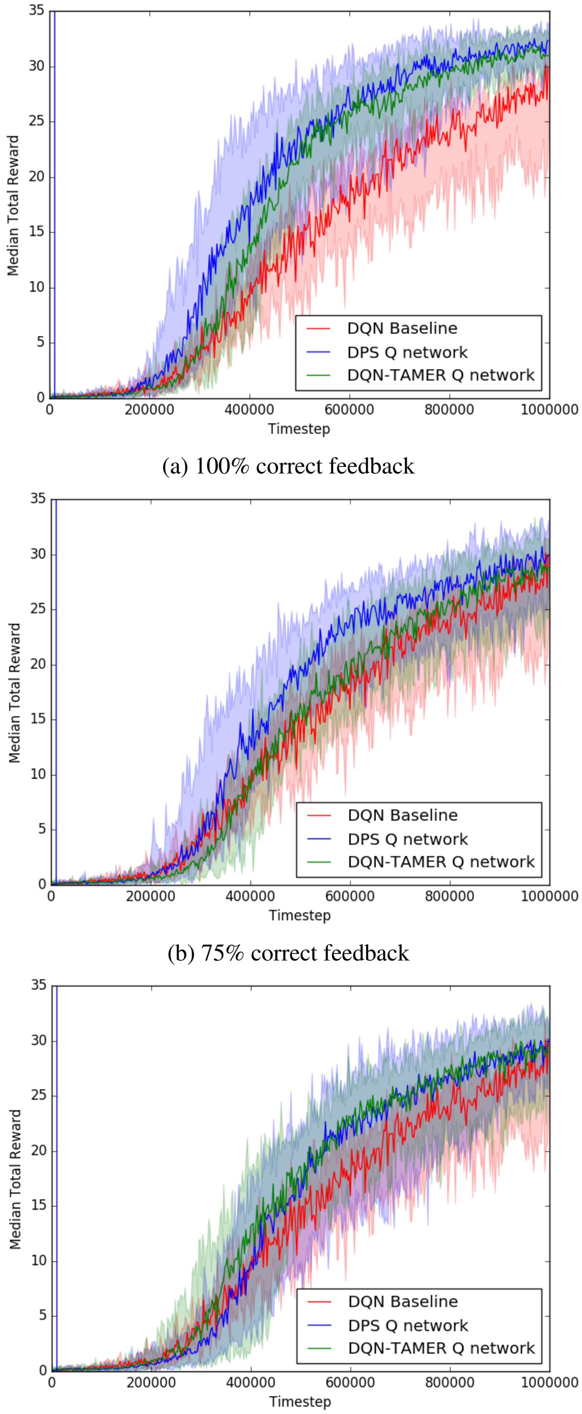
Figure 2: Reacher-v2 Environment [Todorov *et al.*, 2012]

To simulate human feedback, we use a pretrained DQN as an oracle. This uses the assumption that a user will give feedback as a label of action optimality. This oracle gives positive feedback when an action has a Q-value within 2.5% of the highest Q-value in the state. In the case of DPS, positive feedback is given as label 1, and negative feedback is given as label 0, whereas for DQN-TAMER negative feedback is labeled as -1. While DQN-TAMER can take scalar feedback rather than just binary, we give only 1 and -1 as feedback for consistency. The oracle gives feedback to the agent and the feedback networks are trained every 10 time steps starting at 0. Feedback completely stops at 10000 time steps, at which point the feedback network training reduces to once every 50 time steps.

We evaluate both DPS and DQN-TAMER against a DQN in two ways. First, we compare the performance of the best-performing hyperparameters over a search of ten hyperparameter settings, chosen using a random search [Bergstra and Bengio, 2012]. Each hyperparameter setting is judged by averaging the area under curve over 10 executions. We sample the hyperparameters exponentially from a range decided by prior manual exploration: for range $[x, y]$ we draw a value uniformly from $[\log(x), \log(y)]$ and exponentiate it. We sample the learning rate for the feedback networks from $[0.05, 0.00005]$, the $l_2$ regularization of the feedback networks from $[0.01, 0.00001]$, and the base for attenuation $\alpha$ from $[0.99997, 0.999997]$. For DQN-TAMER, we set $\alpha_h(t) = \alpha^t$ and $\alpha_q(t) = 1 - \alpha_h(t)$, the weighting used in [Knox and Stone, 2010]. Similar to DPS, the $\alpha_h$ and $\alpha_q$ hyperparameters for DQN-TAMER control how the influence of human feedback decays over the course of the learning process ($\alpha_h$), and how weighted the Q-function is ($\alpha_q$).

Second, we compare the average performance of each algorithm over 100 hyperparameter settings chosen through random search, each run for one execution. We do this as in practice, HRL algorithms such as DPS and DQN-TAMER would likely see little to no hyperparameter optimization due to the significant cost of obtaining human feedback.

For both of these experiments, we repeat over varied correctness of feedback. We apply noise to all states and actions uniformly. Each feedback is flipped with some probability $1 - P$, $P \in [0.6, 0.75, 1.0]$. For each experiment, we set the consistency hyperparameter for DPS, $C = P$. DQN-TAMER does not define $\alpha_h(t)$ and $\alpha_q(t)$ in regards to feedback accuracy, so for all values of $P$ we still sample $\alpha$ exponentially from the range $[0.99997, 0.999997]$ and define $\alpha_h(t)$ and $\alpha_q(t)$ the same way with regard to $\alpha$. As $P$ decreases, changing either the range $\alpha$ is drawn from or defining $\alpha_h(t)$ and $\alpha_q(t)$ differently may improve performance. Future work will include running experiments to determine the best value of $\alpha$ and definition of $\alpha_h(t)$ and $\alpha_q(t)$ for each correctness setting. However, this would involve hyperparameter search over a substantially larger space which, due to the substantial cost of human feedback, makes tuning for the potential increased performance expensive in practice. In comparison, DPS provides its user a way to integrate *a priori* knowledge of feedback correctness without increasing the size of the hyperparameter search space.



(a) 100% correct feedback

(b) 75% correct feedback

(c) 60% correct feedback

Figure 3: Comparison between best performing hyperparameters.

# 4 Results and Discussion

## 4.1 Best Performing Hyperparameters

In all graphs, the y-axis is the mean cumulative reward at each time step, and the shaded area shows the range over all 10 runs. All significance values are calculated using a one way ANOVA with post-hoc Tukey HSD test. For $P = 1$, shown in Figure 3a, the ANOVA produced a significant $p$-value of 0.013. DPS has an area of 18236783.75 under the curve, whereas DQN-TAMER and the DQN have areas of 16932102.5 and 13547898.6 respectively. DPS and DQN-TAMER both significantly outperform the DQN ($p < 0.005$ for both). DPS gives a 7.7% increase over DQN-TAMER, although this is not significant ($p = 0.141$).

For $P = 0.75$, shown in Figure 3b, the ANOVA produced an insignificant $p$-value of 0.054. DPS has an area of 15639707.5 under the curve, whereas DQN-TAMER and the DQN have areas of 13688177.5 and 13547898.6 respectively. DPS significantly outperforms the DQN ($p < 0.005$). DQN-TAMER also outperforms the DQN, but not significantly ($p = 0.9$). DPS gives a 14.2% significant increase over DQN-TAMER ($p < 0.05$).

For $P = 0.6$, shown in Figure 3c, the ANOVA produced a significant $p$-value of 0.013. DPS has an area of 14396098.75 under the curve, whereas DQN-TAMER and the DQN have areas of 15023776.25 and 13547898.6 respectively. DPS outperforms the DQN but not significantly ($p = 0.2408$), while DQN-TAMER significantly outperforms the DQN ($p < 0.05$). DPS has a slight 4.2% decrease from DQN-TAMER, but this result is not significant ($p = 0.6354$).

These results show that with tuned hyperparameters, DPS can match or outperform the DQN and DQN-TAMER. These improvements increase when the correctness is decreased to 75%, but decrease when at 60%. This may be because DPS is able to use $C$ to leverage the useful information when feedback is not entirely correct, but 60% correctness is close to 50%, which gives no additional information over a DQN.

### 4.2 Average Hyperparameter Performance

All results are shown in Table 1, with significance values calculated using a one way ANOVA with post-hoc Tukey HSD test. With $P = 1$, DPS significantly outperforms DQN-TAMER on average. Both DPS and DQN-TAMER outperform the DQN alone, DPS significantly so. With the feedback correctness reduced to $P = 0.75$, DPS outperforms both DQN-TAMER and the DQN but not significantly so. DQN-TAMER also does not significantly outperform the DQN. With $P = 0.6$, DQN-TAMER lowers the performance of the agent more on average than DPS does, in fact the decrease in performance from the DQN alone to DQN-TAMER is a significant one. DPS has a higher AUC than DQN-TAMER, but this difference is not significant. These results suggest that DPS matches or outperforms DQN-TAMER over many hyperparameter settings. This result is useful when the hyperparameter settings cannot be tuned.

## 5 Conclusion and Future Work

In this work, we introduce the DPS algorithm and show that it outperforms or matches a baseline DQN and DQN-TAMER on a simulated task over various settings of feedback correctness, particularly when averaged over multiple hyperparameter settings, which can rarely be tuned prior to training. Using DPS will enable robots to learn from people with varying levels of feedback quality.

A high level drawback in our approach is the amount of data necessary to train neural networks. There are some things approaches that can be done to mitigate this cost, such as the autoencoder pretraining [Warnell *et al.*, 2018] and applying a single piece of feedback to multiple states [Warnell *et al.*, 2018; Arumugam *et al.*, 2019]. Applications of these techniques to DPS could be studied in the future. Another potential method of reducing the total feedback needed is adding active learning to DPS which could also be explored in future work.

Although the design of our oracle was based off of insights about how humans provide feedback [Thomaz and Breazeal, 2008], it is still just an approximation of human behavior. Future work will evaluate DPS with feedback from humans and in comparison to other HRL algorithms.

## 6 Acknowledgements

This work is supported by Intel, the Office of Naval Research awards N000141612835 and N000141612785, the National Science Foundation awards 1564080 and 1724157, and NSF-GRFP Grant DGE-1610403.

## References

[Arakawa *et al.*, 2018] Riku Arakawa, Sosuke Kobayashi, Yuya Unno, Yuta Tsuboi, and Shin-ichi Maeda. Dqn-tamer: Human-in-the-loop reinforcement learning with intractable feedback. *arXiv preprint arXiv:1810.11748*, 2018.

[Arumugam *et al.*, 2019] Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L. Littman. Deep reinforcement learning from policy-dependent human feedback. *CoRR*, abs/1902.04257, 2019.

[Bergstra and Bengio, 2012] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, February 2012.

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

[Christiano *et al.*, 2017] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4299–4307. Curran Associates, Inc., 2017.

[Griffith *et al.*, 2013] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2625–2633. Curran Associates, Inc., 2013.

[Hasselt *et al.*, 2016] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2094–2100. AAAI Press, 2016.

[Knox and Stone, 2009] W. Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the Fifth International Conference on Knowledge Capture*, K-CAP '09, page 9–16, New York, NY, USA, 2009. Association for Computing Machinery.

[Knox and Stone, 2010] W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *AAMAS*, 2010.

[Kurenkov *et al.*, 2019] Andrey Kurenkov, Ajay Mandlekar, Roberto Martin-Martin, Silvio Savarese, and Animesh Garg. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. *arXiv preprint arXiv:1909.04121*, 2019.

[Lakshminarayanan *et al.*, 2016] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2016.

[Lee *et al.*, 2015] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks, 2015.

[Lin *et al.*, 2017] 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.

[MacGlashan *et al.*, 2017] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, David Roberts, Matthew E. Taylor, and Michael L. Littman. Interactive learning from policy-dependent human feedback, 2017.

[Mnih and Hinton, 2012] Volodymyr Mnih and Geoffrey Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th Annual International Conference on Machine Learning (ICML 2012)*, June 2012.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[Sridharan, 2011] Mohan Sridharan. Augmented reinforcement learning for interaction with non-expert humans in agent domains. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, volume 1, pages 424–429. IEEE, 2011.

[Thomaz and Breazeal, 2008] Andrea L. Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artif. Intell.*, 172(6–7):716–737, April 2008.

[Todorov *et al.*, 2012] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[Warnell *et al.*, 2018] Garrett Warnell, Nicholas R. Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *AAAI*, 2018.

[Xiao *et al.*, 2020] Baicen Xiao, Qifan Lu, Bhaskar Ramasubramanian, Andrew Clark, Linda Bushnell, and Radha Poovendran. Fresh: Interactive reward shaping in high-dimensional state spaces using human feedback, 2020.