FRESH: Interactive Reward Shaping in High-Dimensional State Spaces using Human Feedback

Baicen Xiao
University of Washington
Seattle, WA, USA
bcxiao@uw.edu

Qifan Lu
University of Washington
Seattle, WA, USA
lqf96@uw.edu

Bhaskar Ramasubramanian
University of Washington
Seattle, WA, USA
bhaskarr@uw.edu

Andrew Clark
Worcester Polytechnic Institute
Worcester, MA, USA
aclark@wpi.edu

Linda Bushnell
University of Washington
Seattle, WA, USA
lb2@uw.edu

Radha Poovendran
University of Washington
Seattle, WA, USA
rp3@uw.edu

ABSTRACT

Reinforcement learning has been successful in training autonomous agents to accomplish goals in complex environments. Although this has been adapted to multiple settings, including robotics and computer games, human players often find it easier to obtain higher rewards in some environments than reinforcement learning algorithms. This is especially true of high-dimensional state spaces where the reward obtained by the agent is sparse or extremely delayed. In this paper, we seek to effectively integrate feedback signals supplied by a human operator with deep reinforcement learning algorithms in high-dimensional state spaces. We call this ***FRESH (Feedback-based REward SHaping)***. During training, a human operator is presented with trajectories from a replay buffer and then provides feedback on states and actions in the trajectory. In order to generalize feedback signals provided by the human operator to previously unseen states and actions at test-time, we use a *feedback neural network*. We use an ensemble of neural networks with a shared network architecture to represent model uncertainty and the confidence of the neural network in its output. The output of the feedback neural network is converted to a shaping reward that is augmented to the reward provided by the environment. We evaluate our approach on the *Bowling* and *Skiing* Atari games in the arcade learning environment. Although human experts have achieved high scores in these environments, state-of-the-art deep learning algorithms perform poorly. We observe that *FRESH* achieves much higher scores than state-of-the-art deep learning algorithms in both environments. *FRESH* also achieves a **21.4%** higher score than a human expert in *Bowling* and does as well as an expert in *Skiing*.

KEYWORDS

feedback-based reward shaping; deep reinforcement learning; human feedback; ensemble of neural networks

ACM Reference Format:
Baicen Xiao, Qifan Lu, Bhaskar Ramasubramanian, Andrew Clark, Linda Bushnell, and Radha Poovendran. 2020. FRESH: Interactive Reward Shaping in High-Dimensional State Spaces using Human Feedback. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning (RL) is a framework that enables an agent to explore an environment in order to maximize its expected long-term reward, where the reward signal is supplied by the environment. A major attraction of this approach is that the agent can learn to complete tasks even when a model of the environment (i.e. transition probabilities between states) is unknown. With adequate computational resources, model-free RL algorithms have been successfully applied to many domains, including board and computer games [29, 37] and robotics [14, 26, 36]. However, it has been noted that the sample complexity of these algorithms is typically very high, which limits their use on real-world systems [14, 29]. Model-based RL, on the other hand, aims to learn a model of the system, and then optimize a policy given this model [4, 13, 18, 25]. This approach has the benefit of reducing sample complexity, but the scope of the policy is limited by assumptions made on the model.

Although RL algorithms have shown impressive results in certain environments [29, 37], humans are usually much more efficient in terms of the number of actions required to obtain higher rewards. This has especially been observed in environments with high-dimensional state spaces, like video games, where states are raw pixels of images or snapshots of videos. In these settings, any prior knowledge that a human might have about the environment, and their ability to learn from the environment, is crucial in determining success. The authors of [33] showed that a human player is easily able to play and win games in setups where the reward structure is sparse or significantly delayed (for e.g. receiving a reward only for successfully completing one level of a game), while deep RL algorithms struggle. The role of a human operator in providing a *shaping* reward signal to aid the learning process of the RL agent in such environments has not been addressed in prior work.

In this paper, we seek to effectively integrate human feedback with deep RL algorithms in high-dimensional state spaces. We term this ***FRESH***, for **Feedback-based REward SHaping**. We assume that there is a human operator who can provide feedback on actions taken by the RL agent. During training, the operator is presented with trajectories (sequences of states and actions) from a replay buffer and indicates whether an action at a state in the trajectory is *good* or *bad*. The operator is additionally able to provide this feedback on the states themselves. They can also indicate that they are not sure if an action is good or bad (*cannot tell*). If they feel

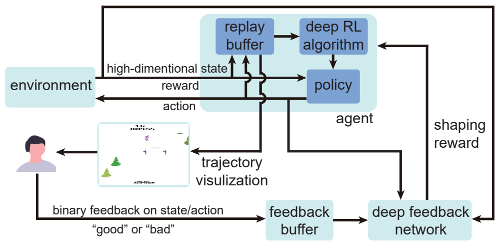

Figure 1: Schematic for *FRESH (Feedback-based REward SHaping)*. A human operator is presented with trajectories of game-play from the replay buffer. At each state in the trajectory, the operator indicates whether the action taken in that state is *good* or *bad*. At some states, the operator also indicates whether the state is *good* or *bad*. This feedback is stored in a feedback buffer. A deep feedback neural network is used to allow the deep reinforcement learning algorithm to generalize feedback signals obtained during training to unseen states and actions at test-time. The output of the feedback network is converted to a shaping reward, which is augmented to the reward provided by the environment.

that sufficient number of feedback signals have been given on a trajectory, they can choose to *skip* the remainder of that trajectory. The motivation for this type of feedback is that it is relatively easier for a human to understand whether a state and the consequence of an action taken at a state will be good or bad. A schematic of our setup is shown in Figure 1.

We wish to clarify that this approach is different from that used in Deep-TAMER [45], which used human-feedback, but did not use the environment reward, and instead adopted a supervised learning framework during training. There, a human operator associated a numerical value indicative of how good the agent's behavior was, in the opinion of the operator. In comparison, in *FRESH*, the operator only needs to provide a qualitative indication of whether an action in a state is good or bad, and we convert this to a shaping reward that is integrated with the reward from the environment.

Due to the size of the state space when operating in high dimensional state spaces, it will not be uncommon for the agent to encounter states at test time that it would have never seen during training. Moreover, the number of feedback signals that can be provided by a human is limited. Therefore, we need to be able to generalize from the feedback in order to determine whether a state encountered at test time is 'similar' to some state seen previously. Neural networks (NNs) will allow us to generalize from feedback. Since an NN yields an output for any input given to it, we require a mechanism to reject outputs that could lead the RL agent towards 'bad' states. One way to achieve this is by quantifying a measure of confidence of the network in its output. This way, by setting an appropriate threshold, outputs of the network that have a confidence score below this threshold can be rejected. Only those outputs with a confidence score above the threshold will be retained and used during the training phase. The authors of [44] used the output of a softmax function as a measure of confidence. However, a shortcoming of this approach is that a model could be uncertain in its predictions even with a high softmax output [10].

Bayesian neural networks [30] have been used to represent the uncertainty in the output of an NN. Two techniques that offset the high costs incurred during inference are bootstrapped ensemble NNs [24, 32] and dropout as Bayesian approximation [10]. The bootstrap principle is to approximate the distribution of a population by a sample distribution [7]. This property allows us to use an ensemble of NNs to effectively represent uncertainty in the model where each network in the ensemble produces a value indicative of its confidence in its output. In the dropout approach, an equivalence was established between *dropout* training in deep NNs and Bayesian inference in Gaussian processes. We prefer the bootstrapped ensemble over dropout in this paper because of its lower time complexity during inference.

1.1 Contributions

We use an ensemble of neural networks with a shared architecture to effectively represent human feedback and predict uncertainty in the model. The human feedback signal is then used as a shaping reward. We evaluate our method on the *Bowling* and *Skiing* Atari games in the Arcade Learning Environment [3]. We choose these games because many state-of-the-art deep reinforcement learning (DRL) algorithms are unable to achieve human-level performance. We observe that *FRESH* is able to outperform state-of-the-art DRL algorithms in both environments. In the *Bowling* game, *FRESH* obtains a higher average score ($\sim$ **180**) than a human expert ($\sim$ **150**) [29]. In the *Skiing* game, *FRESH* performs as well as a human expert.

1.2 Outline of Paper

The paper is organized as follows: Section 2 gives a brief introduction to reinforcement learning. Our approach is outlined in Section 3 and we present detailed results of our experiments in Section 4. Section 5 summarizes related work and we conclude in Section 6.

2 REINFORCEMENT LEARNING

A Markov decision process (MDP) [34] is a tuple $(S, A, \mathbb{T}, \rho_0, R)$. S is the set of states, A the set of actions. $\mathbb{T} : S \times A \times S \to [0, 1]$ encodes $\mathbb{P}(s_{t+1}|s_t, a_t)$, the probability of transition to s_{t+1}, given current state s_t and action a_t. ρ_0 is a probability distribution over the initial states. $R : S \times A \times S \to \mathbb{R}$ denotes the reward that the agent receives when transitioning to state s_{t+1} from s_t while taking action a_t.

The goal for an RL agent [39] is to interact with its environment (that is modeled as an MDP) and learn a *policy* π, in order to maximize $J := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})]$. Here, γ is a discounting factor, and the expectation is taken over the trajectory $\tau = (s_0, a_0, r_0, s_1, \dots)$ induced by policy π. If $\pi : S \to A$, the policy is *deterministic*. On the other hand, a randomized policy returns a probability distribution over the set of actions, and is denoted $\pi : S \times A \to [0, 1]$. It should be noted that the agent does not have direct access to the transition probabilities or the reward.

The value of a state-action pair (s, a) following policy π is represented by the *Q-function*, $Q^\pi(s, a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})|s_0 = s, a_0 = a]$. The Q-function allows us to calculate the state value $V^\pi(s) = \mathbb{E}_{a\sim\pi}[Q^\pi(s, a)]$. The advantage of a particular action a, over other actions at a state s is defined by $A^\pi(s, a) := Q^\pi(s, a) - V^\pi(s)$. The overall policy can be determined by (dynamically) updating Q-values, using Q-learning [46]. Exploration strategies in the environment at the early stages of learning include ϵ-greedy (choose action that maximizes Q-value with high probability, and a random action with low probability) and Boltzmann exploration (choose action according to a probabilistic model).

3 FEEDBACK-BASED REWARD SHAPING

The use of human feedback in high-dimensional state spaces raises the following questions:

Q1: How can feedback signals provided by a human operator be effectively used in high-dimensional state spaces?

Q2: How can this feedback be meaningfully integrated with deep reinforcement learning (DRL) algorithms?

In this section, we present ***FRESH***, a feedback-based reward shaping approach towards answering these questions. We assume that an RL agent has to learn to maximize its reward in an environment that has a high-dimensional state space. An example of such an environment is an Atari game in the Arcade Learning Environment. 'States' in this environment are collections of pixels (images). Although deep NNs have been successful in outperforming a human expert player in many Atari games [29], there are some games where the expert is still able to perform better than state-of-the-art DRL algorithms. This paper studies two examples of the latter (Bowling and Skiing), and we demonstrate that *FRESH* performs at least as well as a human expert player in these environments.

3.1 Binary-valued Human Feedback

We assume that a human operator is able to provide binary-valued feedback on actions and states. We do not require the operator to be an expert. However, we assume that this operator will have the ability to understand game-play in the environment after explanation by an expert. Feedback given on an action is a (local) interpretation of the quality of the action at a particular state, independent of how 'good' or 'bad' the state is. In comparison, feedback provided on a state is a (more global) interpretation on whether the current observed state is good or bad in terms of the rewards that can be obtained. The reason we use this type of feedback signal is that in many settings, rather than assigning a numeric value, it is relatively easier for a human to interpret whether a state or an action taken in a particular state is simply 'good' or 'bad'. Specifically, the human provides feedback on actions implicitly according to a hidden function $h_{s,a}(\cdot,\cdot) : S \times A \to \{0,1\}$, and feedback on states according to a hidden function $h_s(\cdot) : S \to \{0,1\}$. In each case, 0/1 denote *bad/ good.* The human operator can additionally provide feedback signals that indicate *not sure* (if they are not certain whether an action in a state is 'good' or 'bad') and *skip* (if the operator feels that sufficient number of feedback signals have been provided for a trajectory). These latter two signals are not used to train the agent, but they allow the human operator to significantly reduce the amount of time spent in the training phase. We note that both $h_{s,a}$ and h_s can be time-varying, implying that the feedback can change for the same state or state-action pair as the agent training process proceeds. Moreover, the human operator might also change their expectations of the agents' performance over time, which justifies the time-varying nature of the feedback signal.

We assume that at each time, in a state s_t, there is exactly one action a_t that is the 'best' in that state. That is, taking this action will lead to the agent receiving a higher (accumulated) reward than taking any other action. We formulate a binary classification problem, and use maximum likelihood estimation (MLE) to determine the best action in a state. We denote the predicted probability that action a_i is the best in state s by $f^{a_i}(s)$. Furthermore, $\sum_{i=1}^{|A|} f^{a_i}(s) = 1$. To implement our maximum likelihood estimator, we use cross-entropy loss with both positive and negative labels:

$$\mathcal{L}(f^{a_i}(s), h_{s,a_i}) = -h_{s,a_i} \log f^{a_i}(s) - (1 - h_{s,a_i}) \log \sum_{a \neq a_i} f^a(s),$$

where h_{s,a_i} is the binary-valued human feedback associated to state-action pair (s, a_i).

We formulate an analogous classification problem for feedback on states. Let $g(s)$ denote the predicted probability that state s is *good.* The loss function used to evaluate this prediction is:

$$\mathcal{L}(g(s), h_s) = -h_s \log g(s) - (1 - h_s) \log(1 - g(s)),$$

3.2 Generalizing Human Feedback

A challenge in high-dimensional state spaces that is not encountered in the tabular setting is that the agent might observe a lot of states at test-time that it might have never seen during training. Moreover, since the number of feedback signals that a human can provide is often limited, it is important to be able to *generalize* from the feedback in order to determine whether a state encountered at test time is similar to some state seen during training. In order to fully exploit the feedback signals given by a human, we leverage deep neural networks (DNNs). DNNs have been shown to have the ability to use feedback on states and actions and generalize it to other states and actions that have not been previously seen, but 'similar' to those already known [11]. We term the neural networks used to abstract the feedback provided by a human operator as *feedback neural networks (FNNs)* for the rest of this paper.

Using NNs to generalize human feedback presents another challenge. NNs typically produce an output for any input, and do not have a measure of confidence. This could lead to a scenario when the network produces an arbitrary output to an input state that it has never seen during training, and this could lead the RL agent to undesired/ incorrect states. This necessitates development of a mechanism that allows the agent to reject the output of FNNs.

A class of NNs called *Bayesian neural networks* [30] are able to produce both an output, and a value indicating the confidence of the NN in the output. However, this process is computationally expensive. To obtain these confidence values, approximations of Bayesians NNs using dropout [10] and ensemble of NNs for predicting model uncertainty [24, 32] have been proposed. We use the bootstrap ensemble NN architecture in this paper due to its lower time complexity during inference when compared to dropout.

In order to achieve a further reduction in complexity, we use a shared network architecture as shown in Figure 2. In this architecture, the entire network has $K_s + K_a$ heads, with K_s bootstrapped heads for learning feedback on states and K_a bootstrapped heads for learning feedback on actions, along with a shared network. Each head is randomly initialized and trained on a bootstrapped subset of feedback data, that is sampled with replacement from the entire feedback data. All heads share the same preceding layers, which allows the entire network to be trained more efficiently. We collect

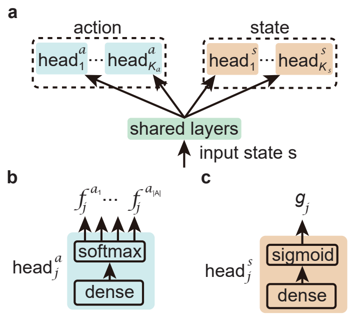
a
action
state
head$_1^a$ ··· head$_{K_a}^a$
head$_1^s$ ··· head$_{K_s}^s$
shared layers
input state s
b
c
$f_j^{a_1}$ ··· $f_j^{a_{|A|}}$
softmax
dense
head$_j^a$
g_j
sigmoid
dense
head$_j^s$

Figure 2: Figure a shows the multi-head architecture with shared layers for the feedback neural network that we use in this paper. Figures b and c show architectures of parts of the networks corresponding to learning feedback on actions and feedback on states respectively.

the prediction from each head and use the mean of these predictions as the final prediction.

Let $\mathbf{f}_j = \left(f_j^{a_1}(s), f_j^{a_2}(s), \ldots, f_j^{a_{|A|}}(s)\right)$ denote the output of the j-th action head, where $f_j^{a_i}(s)$ is the probability that action a_i is the best in state s, and let $g_j(s)$ denote the output of the j-th state head. In order to predict the human feedback on actions, we define:

$$\text{pred}_{\text{action}}(s) = \underset{i\in\{1,\ldots,|A|\}}{\arg\max} \left\{\frac{1}{K_a}\sum_{j=1}^{K_a} f_j^{a_i}(s)\right\},$$

and to predict human feedback on states, we define:

$$\text{pred}_{\text{state}}(s) = \begin{cases} 1, & \text{if } \frac{1}{K_s}\sum_{j=1}^{K_s} g_j(s) > 0.5, \\ 0, & \text{otherwise.} \end{cases}$$

We calculate the empirical standard deviation of individual prediction by different heads and use it as a proxy for the confidence value. Let $p_j^a(s) = \underset{i\in\{1,\ldots,|A|\}}{\arg\max} f_j^{a_i}(s)$ denote the prediction of the j-th action head. Then, the confidence value $c_a(s)$ for predicting human feedback on actions can be obtained by computing the empirical standard deviation:

$$c_a(s) = 1 - \sqrt{\frac{1}{K_a - 1}\sum_{j=1}^{K_a}\left(p_j^a(s) - \frac{\sum_{j=1}^{K_a} p_j^a(s)}{K_a}\right)^2}$$

The (proxy for the) confidence $c_s(s)$ of predicting the human feedback on states can be computed similarly.

3.3 Reward Shaping

The reward signal supplied by the environment can often be sparse and/ or significantly delayed, although it can *accurately* define desired goals for the agent. In order to make the learning procedure more efficient, the human feedback can be introduced through reward shaping. Once an approximation of the human feedback model is available, we can transfer the model output to more frequent and timely reward signals, although this additional reward may sometimes be incorrect due to an error made by human operator providing the feedback.

In order to make use of the feedback given by the human operator, and not limit the agent's performance when this operator makes an error, we present a way to combine FNNs with the environment reward. Let s_t, a_t, and s_{t+1} respectively denote the state and action at time t, and the next state after taking action a_t in s_t. The feedback received on actions is transferred to an additional reward r_a as:

$$r_a(s_t, a_t, s_{t+1}) = \begin{cases} 1, & \text{if } a_t = \text{pred}_{\text{action}}(s_t) \text{ and } c_a(s_t) > 1 - \beta_a \\ 0, & \text{otherwise,} \end{cases}$$

where β_a is a pre-assigned constant threshold. If the action taken by the agent is consistent with the best action predicted by the FNN and the confidence of FNN in the prediction $c_a(s_t)$ is higher than $1 - \beta_a$, then an additional reward is provided. Similarly, feedback received on states can be transferred to an additional reward r_s as:

$$r_s(s_t, a_t, s_{t+1}) = \begin{cases} 1, & \text{if } \text{pred}_{\text{state}}(s_{t+1}) = 1 \text{ and } c_s(s_{t+1}) > 1 - \beta_s \\ 0, & \text{otherwise,} \end{cases}$$

If r_e denotes the environment reward, the *shaped reward* $r(s_t, a_t, s_{t+1})$ is then:

$$r(s_t, a_t, s_{t+1}) = r_e(s_t, a_t, s_{t+1}) + \lambda_a r_a(s_t, a_t, s_{t+1}) + \lambda_s r_s(s_t, a_t, s_{t+1}), \quad (1)$$

where λ_a and λ_s are the weights, balancing the importance of the three rewards. Although λ_a and λ_s can decay over time, during our experiments we observed that keeping them constant works well.

3.4 Algorithm

We evaluate *FRESH* when it is used with deep RL algorithms that use a replay buffer for learning, e.g. DQN. The procedure for collecting the human feedback is summarized in Algorithm 1. First, we sample a trajectory τ from the replay buffer B_q (line 1). Any choice of sampling method can be used in this step. For example, the trajectory with highest or lowest reward may be given higher priority at different training stages. This sampled trajectory is then presented to the human operator in order to receive feedback. The speed at which τ is displayed to the human can be much slower than actual game play. For example, when the states are represented by images, we can apply a lower frame rate so that it will be easier for the operator to assess states and actions in the trajectory. After feedback on a state f_s or feedback on an action f_a is provided, a mask $\mathbf{m}_s \in \mathbb{Z}^{K_s}$ or $\mathbf{m}_a \in \mathbb{Z}^{K_a}$ will be sampled from a masking

Algorithm 1 HumanFeedbackCollection

```
Input: Replay buffer $B_q$ storing trajectory experience and buffer
    $B_f$ storing human feedback. Masking distributions $M_s, M_a$
 1: Sample a trajectory $\tau$ from $B_q$ and visualize $\tau$ for feedback.
 2: for $(s_t, a_t, r_t, s_{t+1}) \in \tau$ do
 3:     if new feedback on state $f_s$ is available then
 4:         sample $\mathbf{m}_s \sim M_s$ and store $(s_{t+1}, f_s, \mathbf{m}_s)$in $B_f$
 5:     end if
 6:     if new feedback on action $f_a$ is available then
 7:         sample $\mathbf{m}_a \sim M_a$ and store $(s_t, a_t, f_a, \mathbf{m}_a)$ in $B_f$
 8:     end if
 9: end for
10: return $B_f$
```

distribution. This will indicate which heads this feedback should be used to train. For example, the components of the mask can be independently drawn from a Bernoulli distribution (double or nothing bootstrap) or from an exponential distribution (Bayesian non-parametric posterior of a Dirichlet process) [32]. We note that feedback will not be provided on all states or actions, since the human operator might refuse to provide an assessment if they are not sure of the quality of the state/action. The feedback together with the mask will be stored in the feedback buffer (line 3-8).

Algorithm 2 describes *FRESH*. The feedback buffer is initialized by providing feedback on trajectories from random play to train the feedback networks FNN (lines 1-7). The FNNs are updated using stochastic gradient descent (SGD). For training the value networks Q, we use a DQN-based algorithm [42, 43], but the reward function is changed to Equation (1) (lines 9-12). The human operator is asked to provide feedback every N_c episodes (lines 13-15). If N_f new feedback signals are available, the FNN is re-tuned (lines 16-19).

Algorithm 2 *FRESH* for DQN

```
Input: Value networks $Q$. Feedback networks FNN. Masking distributions $M_s, M_a$. Replay buffer $B_q$ storing experience for training DQN and buffer $B_f$ storing human feedback for training FNN. Thresholds $\beta_a$ and $\beta_s$. Weights $\lambda_a$ and $\lambda_s$. Feedback collection frequency $N_c$ and FNN update frequency $N_f$. Initial number of trajectories and feedback $n_i$ and $m_i$.
1:  repeat
2:      Sample trajectories based on random play and store trajectories in $B_q$.
3:  until Collect $n_i$ trajectories in $B_q$
4:  repeat
5:      $B_f = HumanFeedbackCollection(B_q, B_f, M_s, M_a)$
6:  until collect $m_i$ feedback in $B_f$
7:  sample batches from $B_f$ and update FNN using SGD
8:  for Episode i=1, . . . do
9:      repeat
10:         execute action $a = \arg\max_a Q(s, a)$, observe reward $r$ and next state $s'$ and store $(s, a, r, s')$ in $B_q$
11:         update $Q$ using DQN algorithm but change reward function to Equation (1)
12:     until end of episode
13:     if i % $N_c$ == 0 then
14:         $B_f = HumanFeedbackCollection(B_q, B_f, M_s, M_a)$
15:     end if
16:     if Number of new feedback $new_f > N_f$ then
17:         sample batches from $B_f$ and update FNN using SGD
18:         $new_f \leftarrow 0$
19:     end if
20: end for
```

4 EXPERIMENTAL EVALUATION

We evaluate *FRESH* on two Atari games in the Arcade Learning Environment [3]. Figure 3 shows screen shots of game play in the *Bowling* and *Skiing* environments. Although human experts can achieve high scores with relative ease in these games, it has been extremely difficult for state-of-the-art deep reinforcement learning (DRL) algorithms to match this. We observe that the performance on these games using *FRESH* compares favorably with that of a human expert, and is significantly improved over other DRL algorithms.

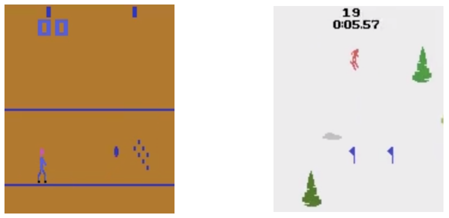

Figure 3: Snapshots from game-play in the Atari games of Bowling (left) and Skiing (right). In Bowling, the goal for the player is to roll the ball and knock down as many pins as possible. In Skiing, the goal for the player is to reach the bottom of a valley as soon as possible, and at the same time, pass through as many gates as possible while avoiding obstacles.

4.1 Game Description and Experiment Setup

The Bowling game comprises four actions, *no-action, up, down, fire*. A game lasts 10 rounds and the player (agent) gets two chances in each round to roll the bowling ball to knock down as many pins as possible. The game begins with the player choosing a position to release (*fire*) the ball by moving vertically using actions *up* or *down*. After releasing the ball, the player gets one chance to steer the direction of the ball by taking actions *up* or *down*. The reward structure of the game makes it difficult for state-of-the-art deep reinforcement learning (DRL) algorithms to obtain a high reward. In particular, the player does not receive an immediate reward if all pins are knocked down in one turn. Instead, this reward will be supplied at the end of the next turn, together with the reward for that turn. This makes it difficult for DRL algorithms to correctly provide credit for actions by simply looking at the reward.

The Skiing game consists of three actions, *no-action, left, right*. The player controls their direction of motion in order to avoid obstacles (trees and moguls) and pass through the gates. The goal for the player is to reach the bottom of the valley as soon as possible, and in the process, pass through as many gates as possible. This game is difficult for DRL algorithms to play since the reward indicating the number of gates passed through is supplied only at the end the game, making the credit assignment task difficult.

In all our experiments, we use the same neural network architecture and hyper-parameters for DQN as [43] does. Additionally, we apply double Q-learning [42] to avoid overestimation of action values. The shared network of our feedback neural network (FNN) has the same architecture as the convolutional layers of DQN, and each head of the FNN adopts three fully-connected layers with batch-normalization. Each state is a tensor stacked by 4 gray-scale images, obtained by converting 4 consecutive colored video game frames. The regions of the frames showing the game score are removed when the frame is an input to the FNN. We clip the environment reward using the same approach as [29] does, and set $\lambda_a = 0.2$ and $\lambda_s = 0.1$ across the experiments. In the early stages of training the FNN, when sufficient diverse feedback signals are

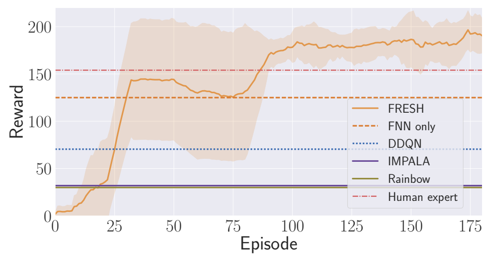

Figure 4: *FRESH* outperforms state-of-the-art deep reinforcement learning algorithms, and also outperforms a human expert player in Bowling. The shaded region indicates the variance of the reward.

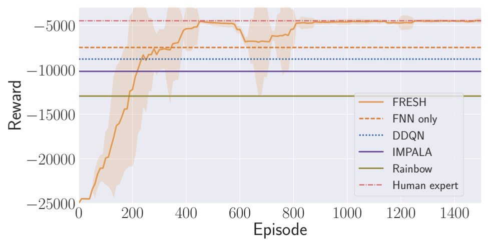

Figure 5: *FRESH* outperforms state-of-the-art deep reinforcement learning algorithms, and performs as well as a human expert player in Skiing. The shaded region indicates the variance of the reward.

not available, the agent might be distracted from the true goal and run into a *cycling* problem, as indicated in [31]. To alleviate this problem, when $|s_t - s_{t-1}|$ is smaller than a known threshold value (which is indicative of the agent being stuck in a local state), we give the agent a penalty to offset the reward supplied by the FNN. This works well in our experiments. Bernoulli distribution is used as the masking distribution. To evaluate Algorithm 2, a human operator provides feedback using a computer with an user interface to visualize trajectories and receiving feedback. While providing feedback, the human trainer is allowed to say *not sure* for states and actions for which the human is not sure about the quality. In the following, we first evaluate the performance of *FRESH* in both Bowling and Skiing and then present a detailed study on the effect of the different components of the FNN in Bowling.

4.2 *FRESH* for Bowling and Skiing

We first collect $n_i = 100$ trajectories using random play, which are sorted based on the accumulated reward. The trajectory with the highest reward is given the highest priority to obtain human feedback. We collect a total of $m_i = 500$ feedback signals for Bowling and $m_i = 5000$ signals for Skiing from trajectories of random play. While training the DQN, we collect feedback signals every $N_c = 30$ episodes. When $N_f = 300$ new feedback signals is available, we update the FNN using all the data in the feedback buffer. In our experiments, we observed that the ratio of the feedback signal *good* over *bad* is around 0.2. In this section, we fix the number of heads $K_a = K_s = 10$ and thresholds $(\beta_a = 1.0, \beta_s = 0.02)$.

Figures 4 and 5 show the performance of *FRESH* in *Bowling* and *Skiing*. Each curve is an average over five runs with different human feedback and initialization. Shaded regions indicate the standard deviation. We compare the performance of *FRESH* (Algorithm 2) with Double-DQN [42], IMPALA [8], Rainbow [16], and Human expert. We use the best final performance reported in the literature for Double-DQN, IMPALA and Rainbow. The human expert performance data is from [29]. Since the trained FNN is able to output an estimated best action for every state, it can also be used as a policy. Therefore, we report the performance of using FNN alone as well.

FRESH allows the agent to learn good policies in both environments. In contrast, state-of-the-art DRL algorithms fail even after training for > 100 million frames. *FRESH* lets the agent learn policies that allow it to obtain higher scores than a human expert in *Bowling*. This indicates that *FRESH* can not only guide the learning process but also that the final performance is not limited by the quality of feedback. In *Skiing*, *FRESH* outperforms the policy induced by the FNN and achieve a similar score as a human expert.

We observe that *FRESH* allows the agent to achieve an average score of **187** in *Bowling*. This is **21.4**% higher than the average score obtained by a human expert in this environment. The highest score obtained by our algorithm in *Bowling* is > **200**. State-of-the-art deep RL algorithms perform poorly in comparison- DDQN obtains an average score of 70.5, while the other methods report an even lower score. Deep-TAMER [45], on the other hand, is able to achieve an average score of around 180, and a high-score of around 200. In *Skiing*, *FRESH* is able to obtain an average score of −**4400**, which is equal to that obtained by an human expert player. The deep RL algorithms perform poorly in this environment also. The authors of [45] do not report a score in the Skiing environment.

4.3 Discussion: Bowling

In order to better understand the performance of our algorithm, we carry out several modifications:

1. *Type of feedback*: We train the FNN using feedback on actions only, feedback on states only, or feedback on both actions and states. The number of heads of the FNN is fixed at $K_a = K_s = 10$, and the thresholds are fixed to $(\beta_a = 1.0, \beta_s = 0.02)$.
2. *Number of heads*: We modify the number of heads of the FNN. We consider the cases: $K_a = K_s = 1$ (no ensemble), $K_a = K_s = 5$, $K_a = K_s = 10$ and $K_a = K_s = 20$. When $K_a, K_s > 1$, thresholds are fixed to $\beta_a = 1.0$ and $\beta_s = 0.02$. Feedback on both states and actions is used to train the FNN.
3. *Thresholds*: We fix $K_a = K_s = 10$ and use feedback on both states and actions to train the FNN, but vary the thresholds. We compare: no threshold, $(\beta_a = 1.0, \beta_s = 0.02)$, $(\beta_a = 1.0, \beta_s = 0.2)$, $(\beta_a = 0.5, \beta_s = 0.02)$.

Due to the difficulty of obtaining large amounts of human interaction data, we reuse feedback data collected from the experiments of the previous section and train FNN at the start of DQN training.

Figure 6 compares the effect of using different types of feedback for training the FNN. We observe that feedback provided on both states and actions, or only on actions allows the agent to achieve super-human performance. Providing feedback on actions alone is only slightly worse in terms of average reward obtained and

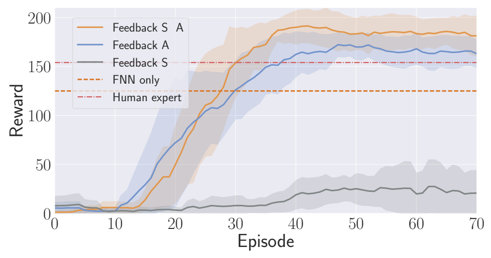

Figure 6: Effect of using different types of feedback in Bowling. The highest scores are achieved when both, feedback on states and actions are used. Providing feedback on states alone is not sufficient to guide the agent's learning process. The shaded region indicates the variance of the reward.

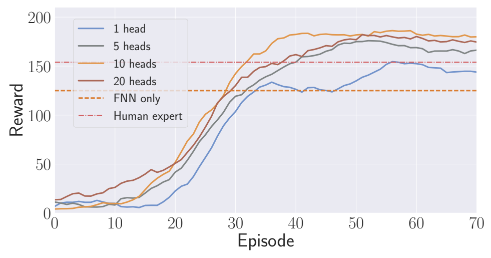

Figure 7: Effect of using different number of heads for the feedback neural network (FNN) in Bowling. When no ensemble is used (1 head), the performance of the FNN is worse than a human expert. Using 10 heads for the FNN yields the best results. Too many heads may make the training process more difficult, and too few heads could mean that the FNN does not have the confidence to discard less useful outputs.

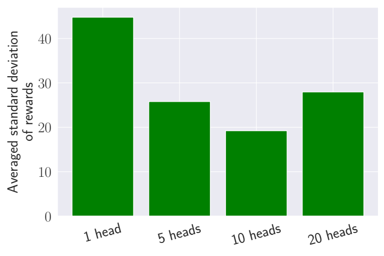

Figure 8: Averaged standard deviation of rewards in Bowling for different number of FNN heads. Using 10 heads has the lowest average standard deviation. Having fewer heads could result in the FNN retaining less useful outputs, while having more heads makes training more difficult.

number of episodes required to perform better than a human expert. In comparison, if only feedback on states is used to train the FNN, the agent is not able to obtain a high reward. This indicates that feedback on states alone is not sufficient to efficiently guide the agent's learning process. A signal that incorporates feedback on actions taken at a state is better suited for this environment.

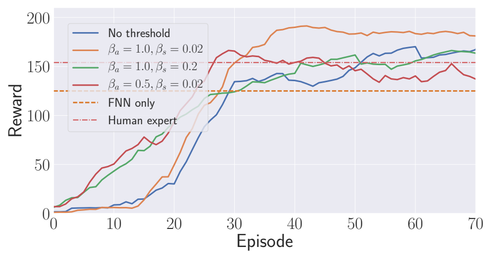

Figure 9: Effect of using different thresholds in Bowling. $\beta_a = 1.0$ and $\beta_s = 0.02$ results in the highest reward. $\beta_a = 1.0$ retains about 97% of the FNN output, and $\beta_s = 0.02$ retains about 85% of the FNN output.

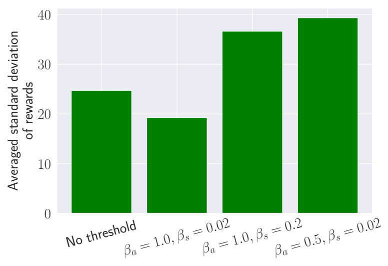

Figure 10: Averaged standard deviation of obtained rewards in Bowling for different threshold values. $\beta_a = 1.0$, $\beta_s = 0.02$ results in the lowest average standard deviation. Standard deviations for other threshold values is higher due to the FNN rejecting or accepting too many outputs of the FNN.

In Figures 7 and 8, we show the effect of varying the number of heads of the FNN on the learning process. When no ensemble is used, the average reward obtained is lower than that received by a human expert, and the (mean of the) standard deviation of the reward is high. This is an indication that in the absence of an ensemble, the agent could obtain a higher reward than a human expert player in some cases, but the large variance could also indicate that the performance is unstable. We observe that $K_a = K_s = 10$ yields the best performance in terms of both variance (lowest) and averaged reward (highest). Our conjecture is that having too many heads can make the training process more difficult while having too few heads might not be able to provide a good confidence value in order to discard some less useful outputs of the FNN.

Figures 9 and 10 show the effect of using different thresholds to eliminate some outputs of the FNN. *FRESH* performs the best in terms of the average reward obtained, and variance of this reward when ($\beta_a = 1.0, \beta_s = 0.02$). In this case, setting $\beta_a = 1.0$ retains about 97% of the FNN outputs, and $\beta_s = 0.02$ retains about 85% of the FNN output. We observe that for some other threshold values, the performance is degraded due to rejecting or accepting too many outputs of the FNN (for example, $\beta_a = 0.5$ rejects around 15% and $\beta_s = 0.2$ keeps around 97% of the FNN outputs).

5 RELATED WORK

The role of feedback provided by a human to an agent in RL settings has been studied by multiple researchers. A survey of recent research in using human guidance for deep RL tasks is presented in [50]. We summarize related work in some of these techniques that are most relevant to this paper in this section.

Reward shaping modifies rewards supplied by the environment to accelerate the learning process of the agent [5, 17, 41]. A framework called TAMER (Training an Agent Manually via Evaluative Reinforcement) that enabled *shaping* (interactively training an agent via an external signal provided by a human) was presented in [21]. This work was extended to enable human feedback to augment an RL agent that learned using an MDP reward signal in [22, 23]. The outcome was that using this human feedback achieved a significant reduction in sample complexity. More recently, the authors of [45] proposed Deep-TAMER, an architecture that extends the TAMER framework to environments with high-dimensional state spaces. In Deep-TAMER, the policy is given by a neural network that is trained via supervised learning using data from feedback provided by a human operator (while not considering the reward given by the environment). Another extension of TAMER, DQN-TAMER [1], modeled additional characteristics of human observers like inferring human reward from facial expressions. Signals provided by the human operator in TAMER was a numerical value indicative of how good the agent's behavior was, in the opinion of the operator.

Potential-based reward shaping (PBRS) methods used 'potential functions' to accelerate the learning process, while preserving the identity of optimal policies [31, 47, 49]. The potential function was designed to encode 'rules' of the environment of the RL agent. However, potential functions will typically need to be pre-specified. This has restricted the use of PBRS to tabular/ low-dimensional state spaces [15, 38]. The cycling problem (repeatedly visiting certain states) mention in Section 4.1 can be resolved by PBRS. This suggests that transforming the FNN output to a potential-based reward is an interesting direction of future research.

Preference-based RL [6] was used to communicate complex goals to allow systems to interact with real-world environments in a meaningful way. Although this approach required a human observer to compare trajectories and provide feedback during training, it alleviated the need for expert observers, since non-experts can easily compare and distinguish between 'good' and 'bad' trajectories. The human preferences were translated into a scalar reward, which was then used as a reward signal to an RL algorithm. This allowed the RL agent to directly learn from expert preferences. However, this approach is limited by assumptions on the existence of a (total) order among the set of trajectories. A survey of preference-based RL methods is presented in [48] for the interested reader.

Consequences of the type and quality of human feedback that could result in forgetting of learned behavior was presented in COACH [28], which used policy-dependent human feedback to learn an optimal policy. The human feedback was interpreted as an unbiased estimate of the advantage function (i.e., the incremental value derived by an action in comparison to the current policy). The authors demonstrated that the role of human feedback is diminished as the agent learns the desired behavior. An extension to domains with large state spaces, Deep-COACH, was presented in [2].

Another policy-based method to provide human feedback is *policy shaping* [12]. Here, feedback is a label on the optimality of an action, rather than a reward signal added to the reward from the environment. In this setup, the RL agent, in addition to receiving a reward from the environment, obtained an indication of whether the most recent action was correct or incorrect, and this label was used to infer the human's belief of the optimal policy in the current state. Further, it could be the case that there might be inconsistencies between the intended communication of the human and the information received by the agent, which results in the construction of a Bayes optimal feedback policy. Extensions to this work considered the effect of human attention [9]- which allows a robot (the RL agent) to learn desired behaviors faster by encouraging exploration of the environment in the presence of human supervision and exploiting actions for which positive feedback has been provided in states previously observed by the human supervisor- and settings where the robot can request feedback in states where it is not sure of the feedback received previously [20]. A similar approach that studied the interpretation of feedback strategies adopted by a human trainer as a probabilistic model was presented in [27], and this resulted in a strategy-aware Bayesian learning (SABL) algorithm.

Demonstrations provided by a human operator were used to synthesize a 'baseline policy' in Human-Agent Transfer (HAT) [40]. This baseline policy was then used to guide learning during the RL procedure. CHAT [44] extended HAT to consider possible errors made while summarizing demonstrations, and used this uncertainty to improve performance. Instead of providing demonstrations, the authors of [35] presented DAGGER, an iterative imitation learning method which required a domain expert be available to provide correct actions during the entire learning process. A subsequent paper [19] presented HG-DAGGER that predicted the performance of the agent using a threshold that modeled uncertainty.

6 CONCLUSION

We presented *FRESH*, a feedback-based reward shaping framework to effectively integrate human feedback with deep RL algorithms in high-dimensional state spaces. We used a feedback neural network to effectively generalize feedback signals provided by the human operator, and an ensemble of neural networks to represent the confidence of the neural network in its output. Our approach was evaluated on the *Bowling* and *Skiing* Atari games of the arcade learning environment. *FRESH* performed much better than state-of-the-art deep learning algorithms in these environments. In *Bowling*, *FRESH* obtained an average score of $\mathbf{187}$, which was $\mathbf{21.4\%}$ higher than the score obtained by a human expert [29]. The highest score obtained by *FRESH* in Bowling was $> \mathbf{200}$. In *Skiing*, *FRESH* obtained an average score of $\mathbf{-4400}$, which was equal to that obtained by a human expert player.

ACKNOWLEDGEMENT

This work was supported by the U.S. Army Research Office and the Office of Naval Research via Grants W911NF-16-1-0485 and N00014-17-S-B001 respectively.

REFERENCES

[1] Riku Arakawa, Sosuke Kobayashi, Yuya Unno, Yuta Tsuboi, and Shin-ichi Maeda. 2018. DQN-TAMER: Human-in-the-Loop Reinforcement Learning with Intractable Feedback. *arXiv preprint arXiv:1810.11748* (2018).

[2] Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L Littman. 2019. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257* (2019).

[3] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[4] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. 2017. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*. 908–918.

[5] Hyeong Soo Chang. 2006. Reinforcement learning with supervision by combining multiple learnings and expert advices. In *Proceedings of the American Control Conference*. 4159–4164.

[6] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*. 4299–4307.

[7] Bradley Efron and Robert Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC Press.

[8] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *International Conference on Machine Learning*. 1406–1415.

[9] Taylor Kessler Faulkner, Elaine Schaertl Short, and Andrea Lockerd Thomaz. 2018. Policy Shaping with Supervisory Attention Driven Exploration. In *International Conference on Intelligent Robots and Systems*. IEEE, 842–847.

[10] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[12] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. 2013. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*. 2625–2633.

[13] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. 2016. Continuous deep Q-learning with model-based acceleration. In *International Conference on Machine Learning*. 2829–2838.

[14] Roland Hafner and Martin Riedmiller. 2011. Reinforcement learning in feedback control. *Machine Learning* 84 (2011), 137–169.

[15] Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowé. 2015. Expressing Arbitrary Reward Functions as Potential-Based Advice. In *AAAI*. 2652–2658.

[16] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*.

[17] Charles Isbell, Christian R Shelton, Michael Kearns, Satinder Singh, and Peter Stone. 2001. A social reinforcement learning agent. In *Proceedings of the International Conference on Autonomous Agents*. 377–384.

[18] Rushikesh Kamalapurkar, Patrick Walters, and Warren E Dixon. 2016. Model-based reinforcement learning for approximate optimal regulation. *Automatica* 64 (2016), 94–104.

[19] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. 2019. HG-DAgger: Interactive Imitation Learning with Human Experts. In *International Conference on Robotics and Automation*. IEEE, 8077–8083.

[20] Taylor Kessler Faulkner, Reymundo A Gutierrez, Elaine Schaertl Short, Guy Hoffman, and Andrea L Thomaz. 2019. Active Attention-Modified Policy Shaping: Socially Interactive Agents Track. In *Autonomous Agents and MultiAgent Systems*. 728–736.

[21] W Bradley Knox and Peter Stone. 2009. Interactively shaping agents via human reinforcement: The TAMER framework. In *International Conference on Knowledge Capture*. 9–16.

[22] W Bradley Knox and Peter Stone. 2010. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Autonomous Agents and Multiagent Systems*. 5–12.

[23] W Bradley Knox and Peter Stone. 2012. Reinforcement learning from simultaneous human and MDP reward. In *Autonomous Agents and Multiagent Systems*. 475–482.

[24] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*. 6402–6413.

[25] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[26] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning and Representations*.

[27] Robert Loftin, Bei Peng, James MacGlashan, Michael L Littman, Matthew E Taylor, Jeff Huang, and David L Roberts. 2016. Learning behaviors via human-delivered discrete feedback: Modeling implicit feedback strategies to speed up learning. *Autonomous agents and multi-agent systems* 30, 1 (2016), 30–59.

[28] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. 2017. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning*. 2285–2294.

[29] 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. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[30] Radford M Neal. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.

[31] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*.

[32] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*.

[33] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*.

[34] Martin L Puterman. 2014. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

[35] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 627–635.

[36] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

[37] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016).

[38] Matthijs Snel and Shimon Whiteson. 2014. Learning potential functions and their representations for multi-task reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 28, 4 (2014), 637–681.

[39] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

[40] Matthew E Taylor, Halit Bener Suay, and Sonia Chernova. 2011. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 617–624.

[41] Andrea Lockerd Thomaz and Cynthia Breazeal. 2006. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*. 1000–1005.

[42] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-learning. In *AAAI*.

[43] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *International Conference on Machine Learning*. 1995–2003.

[44] Zhaodong Wang and Matthew E Taylor. 2017. Improving Reinforcement Learning with Confidence-Based Demonstrations.. In *IJCAI*. 3027–3033.

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

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

[47] Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. 2003. Principled methods for advising reinforcement learning agents. In *International Conference on Machine Learning*. 792–799.

[48] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. 2017. A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research* 18, 1 (2017), 4945–4990.

[49] Baicen Xiao, Bhaskar Ramasubramanian, Andrew Clark, Hannaneh Hajishirzi, Linda Bushnell, and Radha Poovendran. 2019. Potential-Based Advice for Stochastic Policy Learning. In *Proc. IEEE Conference on Decision and Control. Available: arXiv:1907.08823*.

[50] Ruohan Zhang, Faraz Torabi, Lin Guan, Dana H Ballard, and Peter Stone. 2019. Leveraging Human Guidance for Deep Reinforcement Learning Tasks. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 6339–6346.