Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations

Daniel S. Brown [*1] **Wonjoon Goo** [*1] **Prabhat Nagarajan** [2] **Scott Niekum** [1]

Abstract

A critical flaw of existing inverse reinforcement learning (IRL) methods is their inability to significantly outperform the demonstrator. This is because IRL typically seeks a reward function that makes the demonstrator appear near-optimal, rather than inferring the underlying intentions of the demonstrator that may have been poorly executed in practice. In this paper, we introduce a novel reward-learning-from-observation algorithm, Trajectory-ranked Reward EXtrapolation (T-REX), that extrapolates beyond a set of (approximately) ranked demonstrations in order to infer high-quality reward functions from a set of potentially poor demonstrations. When combined with deep reinforcement learning, T-REX outperforms state-of-the-art imitation learning and IRL methods on multiple Atari and MuJoCo benchmark tasks and achieves performance that is often more than twice the performance of the best demonstration. We also demonstrate that T-REX is robust to ranking noise and can accurately extrapolate intention by simply watching a learner noisily improve at a task over time.

1. Introduction

Due to advantages such as computational speed, precise manipulation, and exact timing, computers and robots are often superior to humans at performing tasks with well-defined goals and objectives. However, it can be difficult, even for experts, to design reward functions and objectives that lead to desired behaviors when designing autonomous agents (Ng et al., 1999; Amodei et al., 2016). When goals or rewards are difficult for a human to specify, inverse reinforcement learning (IRL) (Abbeel & Ng, 2004) techniques can be applied to infer the intrinsic reward function of a user from demonstrations. Unfortunately, high-quality demonstrations are difficult to provide for many tasks—for instance, consider a non-expert user attempting to give kinesthetic demonstrations of a household chore to a robot. Even for relative experts, tasks such as high-frequency stock trading or playing complex video games can be difficult to perform optimally.

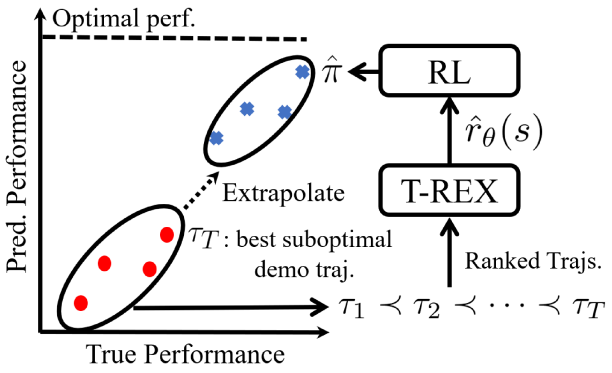


Figure 1. T-REX takes a sequence of ranked demonstrations and learns a reward function from these rankings that allows policy improvement over the demonstrator via reinforcement learning.

If a demonstrator is suboptimal, but their intentions can be ascertained, then a learning agent ought to be able to exceed the demonstrator's performance in principle. However, existing IRL algorithms fail to do this, typically searching for a reward function that makes the demonstrations appear near-optimal (Ramachandran & Amir, 2007; Ziebart et al., 2008; Finn et al., 2016; Henderson et al., 2018). Thus, when the demonstrator is suboptimal, IRL results in suboptimal behavior as well. Imitation learning approaches (Argall et al., 2009) that mimic behavior directly without reward inference, such as behavioral cloning (Torabi et al., 2018a), also suffer from the same shortcoming.

To overcome this critical flaw in current imitation learning methods, we propose a novel IRL algorithm, Trajectory-ranked Reward EXtrapolation (T-REX)[1] that utilizes ranked demonstrations to extrapolate a user's underlying intent be-

[*]Equal contribution [1]Department of Computer Science, University of Texas at Austin, USA [2]Preferred Networks, Japan. Correspondence to: Daniel S. Brown <dsbrown@cs.utexas.edu>, Wonjoon Goo <wonjoon@cs.utexas.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]Code available at https://github.com/hiwonjoon/ICML2019-TREX

yond the best demonstration, even when all demonstrations are highly suboptimal. This, in turn, enables a reinforcement learning agent to exceed the performance of the demonstrator by learning to optimize this extrapolated reward function. Specifically, we use ranked demonstrations to learn a state-based reward function that assigns greater total return to higher-ranked trajectories. Thus, while standard inverse reinforcement learning approaches seek a reward function that *justifies* the demonstrations, we instead seek a reward function that *explains* the ranking over demonstrations, allowing for potentially better-than-demonstrator performance.

Utilizing ranking in this way has several advantages. First, rather than imitating suboptimal demonstrations, it allows us to identify features that are correlated with rankings, in a manner that can be extrapolated beyond the demonstrations. Although the learned reward function could potentially overfit to the provided rankings, we demonstrate empirically that it extrapolates well, successfully predicting returns of trajectories that are significantly better than any observed demonstration, likely due to the powerful regularizing effect of having many pairwise ranking constraints between trajectories. For example, the degenerate all-zero reward function (the agent always receives a reward of 0) makes any given set of demonstrations appear optimal. However, such a reward function is eliminated from consideration by any pair of (non-equally) ranked demonstrations. Second, when learning features directly from high-dimensional data, this regularizing effect can also help to prevent overfitting to the small fraction of state space visited by the demonstrator. By utilizing a set of suboptimal, but ranked demonstrations, we provide the neural network with diverse data from multiple areas of the state space, allowing an agent to better learn both what to do and what not to do in a variety of situations.

We evaluate T-REX on a variety of standard Atari and MuJoCo benchmark tasks. Our experiments show that T-REX can extrapolate well, achieving performance that is often more than twice as high as the best-performing demonstration, as well as outperforming state-of-the-art imitation learning algorithms. We also show that T-REX performs well even in the presence of significant ranking noise, and provide results showing that T-REX can learn good policies simply by observing a novice demonstrator that noisily improves over time.

2. Related Work

The goal of our work is to achieve improvements over a suboptimal demonstrator in high-dimensional reinforcement learning tasks without requiring a hand-specified reward function or supervision during policy learning. While there is a large body of research on learning from demonstrations (Argall et al., 2009; Gao et al., 2012; Osa et al., 2018; Arora & Doshi, 2018), most work assumes access to action labels, while we learn only from observations. Additionally, little work has addressed the problem of learning from ranked demonstrations, especially when they are significantly suboptimal. To the best of our knowledge, our work is the first to show better-than-demonstrator performance in high-dimensional tasks such as Atari, without requiring active human supervision or access to ground-truth rewards.

2.1. Learning from demonstrations

Early work on learning from demonstration focused on behavioral cloning (Pomerleau, 1991), in which the goal is to learn a policy that imitates the actions taken by the demonstrator; however, without substantial human feedback and correction, this method is known to have large generalization error (Ross et al., 2011). Recent deep learning approaches to imitation learning (Ho & Ermon, 2016) have used Generative Adversarial Networks (Goodfellow et al., 2014) to model the distribution of actions taken by the demonstrator.

Rather than directly learn to mimic the demonstrator, inverse reinforcement learning (IRL) (Gao et al., 2012; Arora & Doshi, 2018) seeks to find a reward function that models the intention of the demonstrator, thereby allowing generalization to states that were unvisited during demonstration. Given such a reward function, reinforcement learning (Sutton & Barto, 1998) techniques can be applied to learn an optimal policy. Maximum entropy IRL seeks to find a reward function that makes the demonstrations appear near-optimal, while further disambiguating inference by also maximizing the entropy of the resulting policy (Ziebart et al., 2008; Boularias et al., 2011; Wulfmeier et al., 2015; Finn et al., 2016). While maximum entropy approaches are robust to limited and occasional suboptimality in the demonstrations, they still fundamentally seek a reward function that justifies the demonstrations, resulting in performance that is explicitly tied to the performance of the demonstrator.

Syed & Schapire (2008) proved that, given prior knowledge about which features contribute positively or negatively to the true reward, an apprenticeship policy can be found that is guaranteed to outperform the demonstrator. However, their approach requires hand-crafted, linear features, knowledge of the true signs of the rewards features, and also requires repeatedly solving a Markov decision process (MDP). Our proposed method uses deep learning and ranked demonstrations to automatically learn complex features that are positively and negatively correlated with performance, and is able to generate a policy that can outperform the demonstrator via the solution to a single RL problem.

Our work can be seen as a form of preference-based policy learning (Akrour et al., 2011) and preference-based IRL (PBIRL) (Wirth et al., 2016; Sugiyama et al., 2012) which both seek to optimize a policy based on preference rankings over demonstrations. However, existing approaches only

consider reward functions that are linear in hand-crafted features and have not studied extrapolation capabilities. For a more complete overview survey of preference-based reinforcement learning, see the survey by Wirth et al. (2017). Other methods (Burchfiel et al., 2016; El Asri et al., 2016) have proposed the use of quantitatively scored trajectories as opposed to qualitative pairwise preferences over demonstrations. However, none of the aforementioned methods have been applied to the types of high-dimensional deep inverse reinforcement learning tasks considered in this paper.

2.2. Learning from observation

Recently there has been a shift towards learning from observations, in which the actions taken by the demonstrator are unknown. Torabi et al. (2018a) propose a state-of-the-art model-based approach to perform behavioral cloning from observation. Sermanet et al. (2018) and Liu et al. (2018) propose methods to learn directly from a large corpus of videos containing multiple view points of the same task. Yu et al. (2018) and Goo & Niekum (2019) propose meta-learning-from-observation approaches that can learn from a single demonstration, but require training on a wide variety of similar tasks. Henderson et al. (2018) and Torabi et al. (2018b) extend Generative Adversarial Imitation Learning (Ho & Ermon, 2016) to remove the need for action labels. However, inverse reinforcement learning methods based on Generative Adversarial Networks (Goodfellow et al., 2014) are notoriously difficult to train and have been shown to fail to scale to high-dimensional imitation learning tasks such as Atari (Tucker et al., 2018).

2.3. Learning from suboptimal demonstrations

Very little work has tried to learn good policies from highly suboptimal demonstrations. Grollman & Billard (2011) propose a method that learns from failed demonstrations where a human attempts, but is unable, to perform a task; however, demonstrations must be labeled as failures and manually clustered into two sets of demonstrations: those that overshoot and those that undershoot the goal. Shiarlis et al. (2016) demonstrate that if successful and failed demonstrations are labeled and the reward function is a linear combination of known features, then maximum entropy IRL can be used to optimize a policy to match the expected feature counts of successful demonstrations while not matching the feature counts of failed demonstrations. Zheng et al. (2014) and Choi et al. (2019) propose methods that are robust to small numbers of unlabeled suboptimal demonstrations, but require a majority of expert demonstrations in order to correctly identify which demonstrations are anomalous.

In reinforcement learning, it is common to initialize a policy from suboptimal demonstrations and then improve this policy using the ground truth reward signal (Kober & Peters, 2009; Taylor et al., 2011; Hester et al., 2017; Gao et al., 2018). However, it is often still difficult to perform significantly better than the demonstrator (Hester et al., 2017) and designing reward functions for reinforcement learning can be extremely difficult for non-experts and can easily lead to unintended behaviors (Ng et al., 1999; Amodei et al., 2016).

2.4. Reward learning for video games

Most deep learning-based methods for reward learning require access to demonstrator actions and do not scale to high-dimensional tasks such as video games (e.g. Atari) (Ho & Ermon, 2016; Finn et al., 2016; Fu et al., 2017; Qureshi & Yip, 2018). Tucker et al. (2018) tested state-of-the-art IRL methods on the Atari domain and showed that they are unsuccessful, even with near-optimal demonstrations and extensive parameter tuning.

Our work builds on the work of Christiano et al. (2017), who proposed an algorithm that learns to play Atari games via pairwise preferences over trajectories that are actively collected during policy learning. However, this approach requires obtaining thousands of labels through constant human supervision during policy learning. In contrast, our method only requires an initial set of (approximately) ranked demonstrations as input and can learn a better-than-demonstrator policy without any supervision during policy learning. Ibarz et al. (2018) combine deep Q-learning from demonstrations (DQfD) (Hester et al., 2017) and active preference learning (Christiano et al., 2017) to learn to play Atari games using both demonstrations and active queries. However, Ibarz et al. (2018) require access to the demonstrator's actions in order to optimize an action-based, large-margin loss (Hester et al., 2017) and to optimize the state-action Q-value function using (s, a, s')-tuples from the demonstrations. Additionally, the large-margin loss encourages Q-values that make the demonstrator's actions better than alternative actions, resulting in performance that is often significantly worse than the demonstrator despite using thousands of active queries during policy learning.

Aytar et al. (2018) use video demonstrations of experts to learn good policies for the Atari domains of Montezuma's Revenge, Pitfall, and Private Eye. Their method first learns a state-embedding and then selects a set of checkpoints from a demonstration. During policy learning, the agent is rewarded only when it reaches these checkpoints. This approach relies on high-performance demonstrations, which their method is unable to outperform. Furthermore, while Aytar et al. (2018) do learn a reward function purely from observations, their method is inherently different from ours in that their learned reward function is designed to only imitate the demonstrations, rather than extrapolate beyond the capabilities of the demonstrator.

To the best of our knowledge, our work is the first to sig-

nificantly outperform a demonstrator without using ground truth rewards or active preference queries. Furthermore, our approach does not require demonstrator actions and is able to learn a reward function that matches the demonstrator's intention without any environmental interactions—given rankings, reward learning becomes a binary classification problem and does not require access to an MDP.

3. Problem Definition

We model the environment as a Markov decision process (MDP) consisting of a set of states $\mathcal{S}$, actions $\mathcal{A}$, transition probabilities P, reward function $r : \mathcal{S} \to \mathbb{R}$, and discount factor γ (Puterman, 2014). A policy π is a mapping from states to probabilities over actions, $\pi(a|s) \in [0, 1]$. Given a policy and an MDP, the expected discounted return of the policy is given by $J(\pi) = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t | \pi]$.

In this work we are concerned with the problem of inverse reinforcement learning from observation, where we do not have access to the reward function of the MDP nor the actions taken by the demonstrator. Given a sequence of m ranked trajectories τ_t for $t = 1, \ldots, m$, where $\tau_i \prec \tau_j$ if $i < j$, we wish to find a parameterized reward function $\hat{r}_\theta$ that approximates the true reward function r that the demonstrator is attempting to optimize. Given $\hat{r}_\theta$, we then seek to optimize a policy $\hat{\pi}$ that can outperform the demonstrations.

We only assume access to a qualitative ranking over demonstrations. Thus, we only require the demonstrator to have an internal goal or intrinsic reward. The demonstrator can rank trajectories using any method, such as giving pairwise preferences over demonstrations or by rating each demonstration on a scale. Note that even if the relative scores of the demonstrations are used for ranking, it is still necessary to infer why some trajectories are better than others, which is what our proposed method does.

4. Method

We now describe Trajectory-ranked Reward EXtrapolation (T-REX), an algorithm for using ranked suboptimal demonstrations to extrapolate a user's underlying intent beyond the best demonstration. Given a sequence of m demonstrations ranked from worst to best, $\tau_1, \ldots, \tau_m$, T-REX has two steps: (1) reward inference and (2) policy optimization.

Given the ranked demonstrations, T-REX performs reward inference by approximating the reward at state s using a neural network, $\hat{r}_\theta(s)$, such that $\sum_{s \in \tau_i} \hat{r}_\theta(s) < \sum_{s \in \tau_j} \hat{r}_\theta(s)$ when $\tau_i \prec \tau_j$. The parameterized reward function $\hat{r}_\theta$ can be trained with ranked demonstrations using the generalized loss function:

$$\mathcal{L}(\theta) = \mathbf{E}_{\tau_i, \tau_j \sim \Pi} \Big[\xi \Big(\mathrm{P}\big(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)\big), \tau_i \prec \tau_j \Big) \Big], \quad (1)$$

where Π is a distribution over demonstrations, ξ is a binary classification loss function, $\hat{J}$ is a (discounted) return defined by a parameterized reward function $\hat{r}_\theta$, and $\prec$ is an indication of the preference between the demonstrated trajectories.

We represent the probability P as a softmax-normalized distribution and we instantiate ξ using a cross entropy loss:

$$\mathrm{P}\big(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)\big) \approx \frac{\exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}{\exp \sum_{s \in \tau_i} \hat{r}_\theta(s) + \exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}, \quad (2)$$

$$\mathcal{L}(\theta) = - \sum_{\tau_i \prec \tau_j} \log \frac{\exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}{\exp \sum_{s \in \tau_i} \hat{r}_\theta(s) + \exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}. \quad (3)$$

This loss function trains a classifier that can predict whether one trajectory is preferable to another based on the predicted returns of each trajectory. This form of loss function follows from the classic Bradley-Terry and Luce-Shephard models of preferences (Bradley & Terry, 1952; Luce, 2012) and has been shown to be effective for training neural networks from preferences (Christiano et al., 2017; Ibarz et al., 2018).

To increase the number of training examples, T-REX trains on partial trajectory pairs rather than full trajectory pairs. This results in noisy preference labels that are only weakly supervised; however, using data augmentation to obtain pairwise preferences over many partial trajectories allows T-REX to learn expressive neural network reward functions from only a small number of ranked demonstrations. During training we randomly select pairs of trajectories, τ_i and τ_j. We then randomly select partial trajectories $\tilde{\tau}_i$ and $\tilde{\tau}_j$ of length L. For each partial trajectory, we take each observation and perform a forward pass through the network $\hat{r}_\theta$ and sum the predicted rewards to compute the cumulative return. We then use the predicted cumulative returns as the logit values in the cross-entropy loss with the label corresponding to the higher ranked demonstration.

Given the learned reward function $\hat{r}_\theta(s)$, T-REX then seeks to optimize a policy $\hat{\pi}$ with better-than-demonstrator performance through reinforcement learning using $\hat{r}_\theta$.

5. Experiments and Results

5.1. Mujoco

We first evaluated our proposed method on three robotic locomotion tasks using the Mujoco simulator (Todorov et al., 2012) within OpenAI Gym (Brockman et al., 2016), namely HalfCheetah, Hopper, and Ant. In all three tasks, the goal of the robot agent is to move forward as fast as possible without falling to the ground.

5.1.1. DEMONSTRATIONS

To generate demonstrations, we trained a Proximal Policy Optimization (PPO) (Schulman et al., 2017) agent with the ground-truth reward for 500 training steps (64,000 simulation steps) and checkpointed its policy after every 5 training steps. For each checkpoint, we generated a trajectory of length 1,000. This provides us with different demonstrations of varying quality which are then ranked based on the ground-truth returns. To evaluate the effect of different levels of suboptimality, we divided the trajectories into different overlapping stages. We used 3 stages for HalfCheetah and Hopper. For HalfCheetah, we used the worst 9, 12, and 24 trajectories, respectively. For Hopper, we used the worst 9, 12, and 18 trajectories. For Ant, we used two stages consisting of the worst 12 and 40 trajectories. We used the PPO implementation from OpenAI Baselines (Dhariwal et al., 2017) with the given default hyperparameters.

5.1.2. EXPERIMENTAL SETUP

We trained the reward network using 5,000 random pairs of partial trajectories of length 50, with preference labels based on the trajectory rankings, not the ground-truth return of the partial trajectories. To prevent overfitting, we represented the reward function using an ensemble of five deep neural networks, trained separately with different random pairs. Each network has 3 fully connected layers of 256 units with ReLU nonlinearities. We train the reward network using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 1e-4 and a minibatch size of 64 for 10,000 timesteps.

To evaluate the quality of our learned reward, we then trained a policy to maximize the inferred reward function via PPO. The outputs of each the five reward networks in our ensemble, $\hat{r}(s)$, are normalized by their standard deviation to compensate for any scale differences amongst the models. The reinforcement learning agent receives the average of the ensemble as the reward, plus the control penalty used in OpenAI Gym (Brockman et al., 2016). This control penalty represents a standard safety prior over reward functions for robotics tasks, namely to minimize joint torques. We found that optimizing a policy based solely on this control penalty does not lead to forward locomotion, thus learning a reward function from demonstrations is still necessary.

5.1.3. RESULTS

Learned Policy Performance We measured the performance of the policy learned by T-REX under the ground-truth reward function. We also compared against Behavior Cloning from Observations (BCO) (Torabi et al., 2018a), a state-of-the-art learning-from-observation method, and Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016), a state-of-the-art inverse reinforcement learning algorithm. BCO trains a policy via supervised learning, and has been shown to be competitive with state-of-the-art IRL (Ho & Ermon, 2016) on MuJoCo tasks without requiring action labels, making it one of the strongest baselines when learning from observations. We trained BCO using only the best demonstration among the available suboptimal demonstrations. We trained GAIL with all of the demonstrations. GAIL uses demonstrator actions, while T-REX and BCO do not.

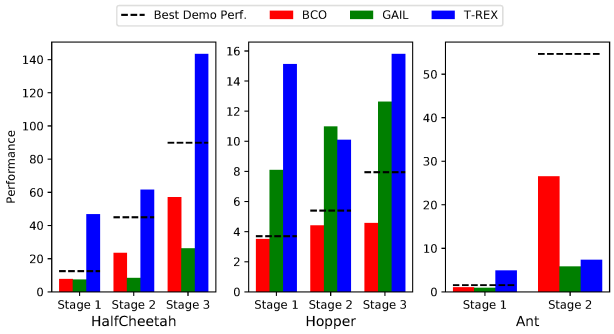
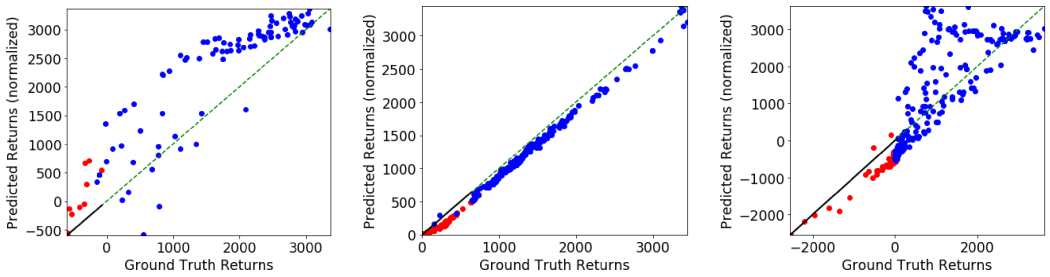


Figure 2. The results on three robotic locomotion tasks when given suboptimal demonstrations. For each stage and task, the best performance given suboptimal demonstrations is shown for T-REX (ours), BCO (Torabi et al., 2018a), and GAIL (Ho & Ermon, 2016). The dashed line shows the performance of the best demonstration.

We compared against three different levels of suboptimality (Stage 1, 2, and 3), corresponding to increasingly better demonstrations. The results are shown in Figure 2 (see the appendix for full details). The policies learned by T-REX perform significantly better than the provided suboptimal trajectories in all the stages of HalfCheetah and Hopper. This provides evidence that T-REX can discover reward functions that extrapolate beyond the performance of the demonstrator. T-REX also outperforms BCO and GAIL on all tasks and stages except for Stage 2 for Hopper and Ant. BCO and GAIL usually fail to perform better than the average demonstration performance because they explicitly seek to imitate the demonstrator rather than infer the demonstrator's intention.

Reward Extrapolation We next investigated the ability of T-REX to accurately extrapolate beyond the demonstrator. To do so, we compared ground-truth return and T-REX-inferred return across trajectories from a range of performance qualities, including trajectories much better than the best demonstration given to T-REX. The extrapolation of the reward function learned by T-REX is shown in Figure 3. The plots in Figure 3 give insight into the performance of T-REX. When T-REX learns a reward function that has a strong positive correlation with the ground-truth reward function, then it is able to surpass the performance of the suboptimal demonstrations. However, in Ant the correlation is not as strong, resulting in worse-than-demonstrator



(a) HalfCheetah (b) Hopper (c) Ant

Figure 3. Extrapolation plots for T-REX on MuJoCo Stage 1 demonstrations. Red points correspond to demonstrations and blue points correspond to trajectories not given as demonstrations. The solid line represents the performance range of the demonstrator, and the dashed line represents extrapolation beyond the demonstrator's performance. The x-axis is the ground-truth return and the y-axis is the predicted return from our learned reward function. Predicted returns are normalized to have the same scale as the ground-truth returns.

performance in Stage 2.

5.2. Atari

5.2.1. Demonstrations

We next evaluated T-REX on eight Atari games shown in Table 1. To obtain a variety of suboptimal demonstrations, we generated 12 full-episode trajectories using PPO policies checkpointed every 50 training updates for all games except for Seaquest and Enduro. For Seaquest, we used every 5th training update due to the ability of PPO to quickly find a good policy. For Enduro, we used every 50th training update starting from step 3,100 since PPO obtained 0 return until after 3,000 steps. We used the OpenAI Baselines implementation of PPO with the default hyperparameters.

5.2.2. Experimental Setup

We used an architecture for reward learning similar to the one proposed in (Ibarz et al., 2018), with four convolutional layers with sizes 7x7, 5x5, 3x3, and 3x3, with strides 3, 2, 1, and 1. Each convolutional layer used 16 filters and LeakyReLU non-linearities. We then used a fully connected layer with 64 hidden units and a single scalar output. We fed in stacks of 4 frames with pixel values normalized between 0 and 1 and masked the game score and number of lives.

For all games except Enduro, we subsampled 6,000 trajectory pairs between 50 and 100 observations long. We optimized the reward functions using Adam with a learning rate of 5e-5 for 30,000 steps. Given two full trajectories τ_i and τ_j such that $\tau_i \prec \tau_j$, we first randomly sample a subtrajectory from τ_i. Let t_i be the starting timestep for this subtrajectory. We then sample an equal length subtrajectory from τ_j such that $t_i \leq t_j$, where t_j is the starting time step of the subtrajectory from τ_j. We found that this resulted in better performance than comparing randomly chosen subtrajectories, likely due to the fact that (1) it eliminates pairings that compare a later part of a worse trajectory with an earlier part of a better trajectory and (2) it encourages reward functions that are monotonically increasing as progress is made in the game. For Enduro, training on short partial trajectories was not sufficient to score any points and instead we used 2,000 pairs of down-sampled full trajectories (see appendix for details).

We optimized a policy by training a PPO agent on the learned reward function. To reduce reward scaling issues, we normalized predicted rewards by feeding the output of $\hat{r}_\theta(s)$ through a sigmoid function before passing it to PPO. We trained PPO on the learned reward function for 50 million frames to obtain our final policy. We also compare against Behavioral Cloning from Observation (BCO) (Torabi et al., 2018a) and the state-of-the-art Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016). Note that we give action labels to GAIL, but not to BCO or T-REX. We tuned the hyperparameters for GAIL to maximize performance when using expert demonstrations on Breakout and Pong. We gave the same demonstrations to both BCO and T-REX; however, we found that GAIL was very sensitive to poor demonstrations so we trained GAIL on 10 demonstrations using the policy checkpoint that generated the best demonstration given to T-REX.

5.2.3. Results

Learned Policy Performance The average performance of T-REX under the ground-truth reward function and the best and average performance of the demonstrator are shown in Table 1. Table 1 shows that T-REX outperformed both BCO and GAIL in 7 out of 8 games. More importantly, T-REX outperformed the best demonstration in 7 out of 8 games. On four games (Beam Rider, Breakout, Enduro, and Q*bert) T-REX achieved score that is more than double the score of the best demonstration. In comparison, BCO performed worse than the average performance of

Table 1. Comparison of T-REX with a state-of-the-art behavioral cloning algorithm (BCO) (Torabi et al., 2018a) and state-of-the-art IRL algorithm (GAIL) (Ho & Ermon, 2016). Performance is evaluated on the ground-truth reward. T-REX achieves better-than-demonstrator performance on 7 out of 8 games and surpasses the BCO and GAIL baselines on 7 out of 8 games. Results are the best average performance over 3 random seeds with 30 trials per seed.

	Ranked Demonstrations		LfD Algorithm Performance		
Game	Best	Average	T-REX	BCO	GAIL
Beam Rider	1,332	686.0	**3,335.7**	568	355.5
Breakout	32	14.5	**221.3**	13	0.28
Enduro	84	39.8	**586.8**	8	0.28
Hero	**13,235**	6,742.0	0	2,167	0
Pong	-6	-15.6	**-2.0**	-21	-21
Q*bert	800	627	**32,345.8**	150	0
Seaquest	600	373.3	**747.3**	0	0
Space Invaders	600	332.9	**1,032.5**	88	370.2

the demonstrator in all games, and GAIL only performed better than the average demonstration on Space Invaders. Despite using better training data, GAIL was unable to learn good policies on any of the Atari tasks. These results are consistent with those of Tucker et al. (2018) that show that current GAN-based IRL methods do not perform well on Atari. In the appendix, we compare our results against prior work (Ibarz et al., 2018) that uses demonstrations plus active feedback during policy training to learn control policies for the Atari domain.

Reward Extrapolation We also examined the extrapolation of the reward function learned using T-REX. Results are shown in Figure 4. We observed accurate extrapolation for Beam Rider, Breakout, Enduro, Seaquest, and Space Invaders—five games where we are able to significantly outperform the demonstrator. The learned rewards for Pong, Q*bert, and Hero show less correlation. On Pong, T-REX overfits to the suboptimal demonstrations and ends up preferring longer games which do not result in a significant win or loss. T-REX is unable to score any points on Hero, likely due to poor extrapolation and the higher complexity of the game. Surprisingly, the learned reward function for Q*bert shows poor extrapolation, yet T-REX is able to outperform the demonstrator. We analyzed the resulting policy for Q*bert and found that PPO learns a repeatable way to score points by inducing Coily to jump off the edge, and is able to consistently achieve high scores without actually clearing any levels. This behavior was not seen in the demonstrations. In the appendix, we plot the maximum and minimum predicted observations from the trajectories used to create Figure 4 along with attention maps for the learned reward functions.

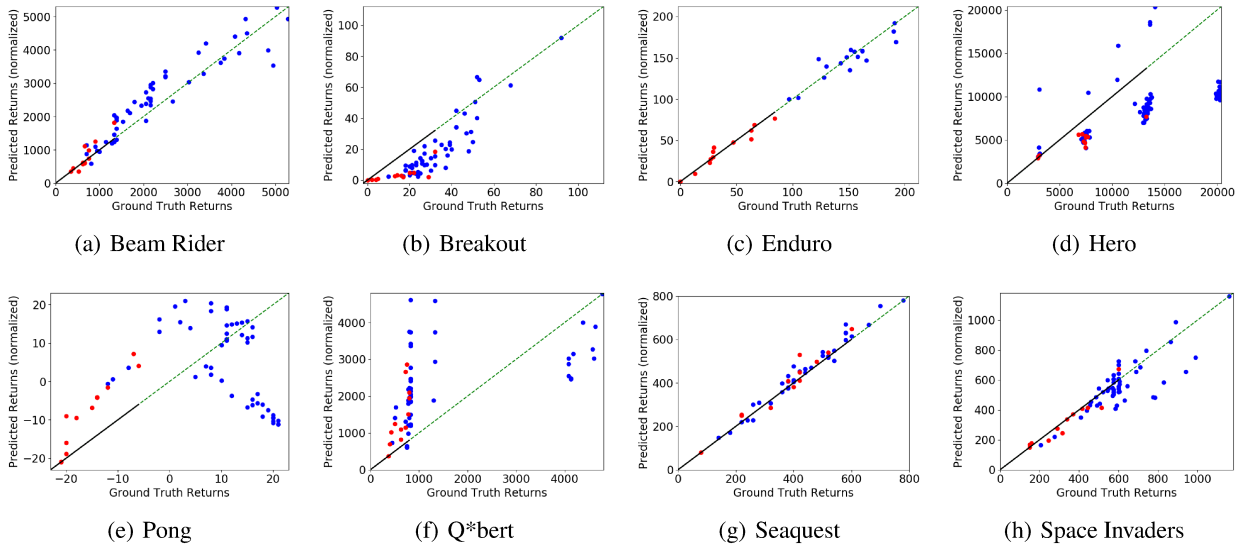
5.2.4. Human Demonstrations

The above results used synthetic demonstrations generated from an RL agent. We also tested T-REX when given ground-truth rankings over human demonstrations. We used novice human demonstrations from the Atari Grand Challenge Dataset (Kurin et al., 2017) for five Atari tasks. T-REX was able to significantly outperform the best human demonstration in Q*bert, Space Invaders, and Video Pinball, but was unable to outperform the human in Montezuma's Revenge and Ms Pacman (see the appendix for details).

5.3. Robustness to Noisy Rankings

All experiments described thus far have had access to ground-truth rankings. To explore the effects of noisy rankings we first examined the stage 1 Hopper task. We synthetically generated ranking noise by starting with a list of trajectories sorted by ground-truth returns and randomly swapping adjacent trajectories. By varying the number of swaps, we were able to generate different noise levels. Given n trajectories in a ranked list provides $\binom{n}{2}$ pairwise preferences over trajectories. The noise level is measured as a total order correctness: the fraction of trajectory pairs whose pairwise ranking after random swapping matches the original ground-truth pairwise preferences. The results of this experiment, averaged over 9 runs per noise level, are shown in Figure 5. We found that T-REX is relatively robust to noise of up to around 15% pairwise errors.

To examine the effect of noisy human rankings, we used the synthetic PPO demonstrations that were used in the previous Atari experiments and used Amazon Mechanical Turk to collect human rankings. We presented videos of the demonstrations in pairs along with a brief text description of the goal of the game and asked workers to select which demonstration had better performance, with an option for selecting "Not Sure". We collected six labels per demonstration pair and used the most-common label as the label for training the reward function. We removed from the training data any pairings where there was a tie for the most-common label or where "Not Sure" was the most common label. We found



(a) Beam Rider (b) Breakout (c) Enduro (d) Hero

(e) Pong (f) Q*bert (g) Seaquest (h) Space Invaders

Figure 4. Extrapolation plots for Atari games. We compare ground truth returns over demonstrations to the predicted returns using T-REX (normalized to be in the same range as the ground truth returns). The black solid line represents the performance range of the demonstrator. The green dashed line represents extrapolation beyond the range of the demonstrator's performance.

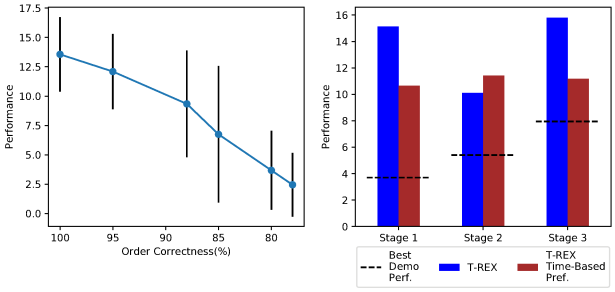


Figure 5. (left): The performance of T-REX for different amounts of pairwise ranking noise in the Hopper domain. T-REX shows graceful degradation as ranking noise increases. The reward function is trained on stage-1 Hopper demonstrations. The graph shows the mean across 9 trials and 95% confidence interval. (right): T-REX results with time-based rankings in the Hopper domain.

that despite this preprocessing step, human labels added a significant amount of noise and resulted in pair-wise rankings with accuracy between 63% and 88% when compared to ground-truth labels. However, despite significant ranking noise, T-REX outperformed the demonstrator on 5 of the 8 Atari games (see the appendix for full details).

5.3.1. Learning from Time-Based Rankings

Finally, we tested whether T-REX has the potential to work without explicit rankings. We took the same demonstrations used for the Mujoco tasks, and rather than sorting them based on ground-truth rankings, we used the order in which they were generated by PPO to produce a ranked list of trajectories, ordered by timestamp from earliest to latest. This provides ranked demonstrations without any need for demonstrator labels, and enables us to test whether simply observing an agent learn over time allows us to extrapolate intention by assuming that later trajectories are preferable to trajectories produced earlier in learning. The results for Hopper are shown in Figure 5 and other task results are shown in the appendix. We found that T-REX is able to infer a meaningful reward function even when noisy, time-based rankings are provided. All the trained policies produced comparable results on most stages to the ground-truth rankings, and those policies outperform BCO and GAIL on all tasks and stages except for Ant Stage 2.

6. Conclusion

In this paper, we introduced T-REX, a reward learning technique for high-dimensional tasks that can learn to extrapolate intent from suboptimal ranked demonstrations. To the best of our knowledge, this is the first IRL algorithm that is able to significantly outperform the demonstrator without additional external knowledge (e.g. signs of feature contributions to reward) and that scales to high-dimensional Atari games. When combined with deep reinforcement learning, we showed that this approach achieves better-than-demonstrator performance as well as outperforming state-of-the-art behavioral cloning and IRL methods. We also demonstrated that T-REX is robust to modest amounts of ranking noise, and can learn from automatically generated labels, obtained by watching a learner noisily improve at a task over time.

Acknowledgments

This work has taken place in the Personal Autonomous Robotics Lab (PeARL) at The University of Texas at Austin. PeARL research is supported in part by the NSF (IIS-1724157, IIS-1638107, IIS-1617639, IIS-1749204) and ONR(N00014-18-2243).

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st international conference on Machine learning*, 2004.

Akrour, R., Schoenauer, M., and Sebag, M. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 12–27. Springer, 2011.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

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

Arora, S. and Doshi, P. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.

Aytar, Y., Pfaff, T., Budden, D., Paine, T. L., Wang, Z., and de Freitas, N. Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*, 2018.

Boularias, A., Kober, J., and Peters, J. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 182–189, 2011.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39 (3/4):324–345, 1952.

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

Burchfiel, B., Tomasi, C., and Parr, R. Distance minimization for reward learning from scored trajectories. In *AAAI*, pp. 3330–3336, 2016.

Choi, S., Lee, K., and Oh, S. Robust learning from demonstrations with mixed qualities using leveraged gaussian processes. *IEEE Transactions on Robotics*, 2019.

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

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. https://github.com/openai/baselines, 2017.

El Asri, L., Piot, B., Geist, M., Laroche, R., and Pietquin, O. Score-based inverse reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 457–465. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, 2016.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Gao, Y., Peters, J., Tsourdos, A., Zhifei, S., and Meng Joo, E. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3): 293–311, 2012.

Gao, Y., Lin, J., Yu, F., Levine, S., Darrell, T., et al. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

Goo, W. and Niekum, S. One-shot learning of multi-step tasks from observation via activity localization in auxiliary video. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Greydanus, S., Koul, A., Dodge, J., and Fern, A. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pp. 1787–1796, 2018.

Grollman, D. H. and Billard, A. Donut as i do: Learning from failed demonstrations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3804–3809. IEEE, 2011.

Henderson, P., Chang, W.-D., Bacon, P.-L., Meger, D., Pineau, J., and Precup, D. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in atari. In *Advances in Neural Information Processing Systems*, pp. 8022–8034, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kober, J. and Peters, J. R. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pp. 849–856, 2009.

Kurin, V., Nowozin, S., Hofmann, K., Beyer, L., and Leibe, B. The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*, 2017.

Liu, Y., Gupta, A., Abbeel, P., and Levine, S. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125. IEEE, 2018.

Luce, R. D. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

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

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

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

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Qureshi, A. H. and Yip, M. C. Adversarial imitation via variational inverse reinforcement learning. *arXiv preprint arXiv:1809.06404*, 2018.

Ramachandran, D. and Amir, E. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, pp. 2586–2591, 2007.

Ross, S., Gordon, G., and Bagnell, D. 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*, pp. 627–635, 2011.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Shiarlis, K., Messias, J., and Whiteson, S. Inverse reinforcement learning from failure. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1060–1068. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

Sugiyama, H., Meguro, T., and Minami, Y. Preference-learning based inverse reinforcement learning for dialog control. In *INTERSPEECH*, pp. 222–225, 2012.

Sutton, R. S. and Barto, A. G. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Syed, U. and Schapire, R. E. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pp. 1449–1456, 2008.

Taylor, M. E., Suay, H. B., and Chernova, S. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 617–624. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, July 2018a.

Torabi, F., Warnell, G., and Stone, P. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018b.

Tucker, A., Gleave, A., and Russell, S. Inverse reinforcement learning for video games. In *Proceedings of the Workshop on Deep Reinforcement Learning at NeurIPS*, 2018.

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

Wirth, C., Akrour, R., Neumann, G., and Fürnkranz, J. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017. URL http://jmlr.org/papers/v18/16-634.html.

Wulfmeier, M., Ondruska, P., and Posner, I. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., and Levine, S. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

Zheng, J., Liu, S., and Ni, L. M. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2198–2205, 2014.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

A. Code and Videos

Code as well as supplemental videos are available at https://github.com/hiwonjoon/ICML2019-TREX.

B. T-REX Results on the MuJoCo Domain

B.1. Policy performance

Table 1 shows the full results for the MuJoCo experiments. The T-REX (time-ordered) row shows the resulting performance of T-REX when demonstrations come from observing a learning agent and are ranked based on timestamps rather than using explicit preference rankings.

B.2. Policy visualization

We visualized the T-REX-learned policy for HalfCheetah in Figure 1. Visualizing the demonstrations from different stages shows the specific way the policy evolves over time; an agent learns to crawl first and then begins to attempt to walk in an upright position. The T-REX policy learned from the highly suboptimal Stage 1 demonstrations results in a similar-style crawling gait; however, T-REX captures some of the intent behind the demonstration and is able to optimize a gait that resembles the demonstrator but with increased speed, resulting in a better-than-demonstrator policy. Similarly, given demonstrations from Stage 2, which are still highly suboptimal, T-REX learns a policy that resembles the gait of the best demonstration, but is able to optimize and partially stabilize this gait. Finally, given demonstrations from Stage 3, which are still suboptimal, T-REX is able to learn a near-optimal gait.

C. Behavioral Cloning from Observation

To build the inverse transition models used by BCO (Torabi et al., 2018a) we used 20,000 steps of a random policy to collect transitions with labeled states. We used the Adam optimizer with learning rate 0.0001 and L2 regularization of 0.0001. We used the DQN architecture (Mnih et al., 2015) for the classification network, using the same architecture to predict actions given state transitions as well as predict actions given states. When predicting $P(a|s_t, s_{t+1})$, we concatenate the state vectors obtaining an 8x84x84 input consisting of two 4x84x84 frames representing s_t and s_{t+1}.

We give both T-REX and BCO the full set of demonstrations. We tried to improve the performance of BCO by running behavioral cloning only on the best $X\%$ of the demonstrations, but were unable to find a parameter setting that performed better than $X = 100$, likely due to a lack of training data when using very few demonstrations.

D. Atari reward learning details

We used the OpenAI Baselines implementation of PPO with default hyperparameters. We ran all of our experiments on an NVIDIA TITAN V GPU. We used 9 parallel workers when running PPO.

When learning and predicting rewards, we mask the score and number of lives left for all games. We did this to avoid having the network learn to only look at the score and recognize, say, the number of significant digits, etc. We additionally masked the sector number and number of enemy ships left on Beam Rider. We masked the bottom half of the dashboard for Enduro to mask the position of the car in the race. We masked the number of divers found and the oxygen meter for Seaquest. We masked the power level and inventory for Hero.

To train the reward network for Enduro, we randomly downsampled full trajectories. To create a training set we repeatedly randomly select two full demonstrations, then randomly cropped between 0 and 5 of the initial frames from each trajectory and then downsampled both trajectories by only keeping every xth frame where x is randomly chosen between 3 and 6. We selected 2,000 randomly downsampled demonstrations and trained the reward network for 10,000 steps of Adam with a learning rate of 5e-5.

Table 1. The results on three robotic locomotion tasks when given suboptimal demonstrations. For each stage and task, the best performance given suboptimal demonstrations is shown on the top row, and the best achievable performance (i.e. performance achieved by a PPO agent) under the ground-truth reward is shown on the bottom row. The mean and standard deviation are based on 25 trials (obtained by running PPO five times and for each run of PPO performing five policy rollouts). The first row of T-REX results show the performance when demonstrations are ranked using the ground-truth returns. The second row of T-REX shows results for learning from observing a learning agent (time-ordered). The demonstrations are ranked based on the timestamp when they were produced by the PPO algorithm learning to perform the task.

	HalfCheetah			Hopper			Ant	
	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2
Best Demo Performance	12.52 (1.04)	44.98 (0.60)	89.87 (8.15)	3.70 (0.01)	5.40 (0.12)	7.95 (1.64)	1.56 (1.28)	**54.64** (22.09)
T-REX (ours)	46.90 (1.89)	**61.56** (10.96)	143.40 (3.84)	**15.13** (3.21)	10.10 (1.68)	**15.80** (0.37)	4.93 (2.86)	7.34 (2.50)
T-REX (time-ordered)	**51.39** (4.52)	54.90 (2.29)	**154.67** (57.43)	10.66 (3.76)	**11.41** (0.56)	11.17 (0.60)	**5.55** (5.86)	1.28 (0.28)
BCO	7.71 (8.35)	23.59 (8.33)	57.13 (19.14)	3.52 (0.14)	4.41 (1.45)	4.58 (1.07)	1.06 (1.79)	26.56 (12.96)
GAIL	7.39 (4.12)	8.42 (3.43)	26.28 (12.73)	8.09 (3.25)	10.99 (2.35)	12.63 (3.66)	0.95 (2.06)	5.84 (4.08)
Best w/ GT Reward	199.11 (9.08)			15.94 (1.47)			182.23 (8.98)	

(a) Stage 1

(b) Stage 2

(c) Stage 3

Figure 1. HalfCheetah policy visualization. For each subplot, (top) is the best given demonstration policy in a stage, and (bottom) is the trained policy with a T-REX reward function.

E. Comparison to active reward learning

In this section, we examine the ability of prior work on active preference learning to exceed the performance of the demonstrator. In Table 2, we denote the results that surpass the best demonstration with an asterisk (*). DQfD+A only surpasses the demonstrator in 3 out of 9 games tested, even with thousands of active queries. Note that DQfD+A extends the original DQfD algorithm (Hester et al., 2017), which uses demonstrations combined with RL on ground-truth rewards, yet is only able to surpass the best demonstration in 14 out of 41 Atari games. In contrast, we are able to leverage only 12 ranked demos to achieve better-than-demonstrator performance on 7 out of 8 games tested, without requiring access to true rewards or access to thousands of active queries from an oracle.

Ibarz et al. (2018) combine Deep Q-learning from demonstrations and active preference queries (DQfD+A). DQfD+A uses demonstrations consisting of (s_t, a_t, s_{t+1})-tuples to initialize a policy using DQfD (Hester et al., 2017). The algorithm then uses the active preference learning algorithm of Christiano et al. (2017) to refine the inferred reward function and initial policy learned from demonstrations. The first two columns of Table 2 compare the demonstration quality given to DQfD+A and T-REX. While our results make use of more demonstrations (12 for T-REX versus 4–7 for DQfD+A), our demonstrations are typically orders of magnitude worse than the demonstrations used by DQfD+A: on average the demonstrations given to DQfD+A are 38 times better than those used by T-REX. However, despite this large gap in the performance of the demonstrations, T-REX surpasses the performance of DQfD+A on Q*Bert, and Seaquest. We achieve these results using 12 ranked demonstrations. This requires only 66 comparisons $(n \cdot (n-1)/2)$ by the demonstrator. In comparison, the DQfD+A results used 3,400 preference labels obtained during policy training using ground-truth rewards.

F. Human Demonstrations and Rankings

F.1. Human demonstrations

We used the Atari Grand Challenge data set (Kurin et al., 2017) to collect actual human demonstrations for five Atari games. We used the ground truth returns in the Atari Grand Challenge data set to rank demonstrations. To generate demonstrations we removed duplicate demonstrations (human demonstrations that achieved the same score). We then sorted the remaining demonstrations based on ground truth return and selected 12 of these demonstrations to form our training set. We ran T-REX using the same hyperparameters as described above.

The resulting performance of T-REX is shown in Table 3. T-REX is able to outperform the best human demonstration on Q*bert, Space Invaders, and Video Pinball; however, it is not able to learn a good control policy for Montezuma's Revenge or Ms Pacman. These games require maze navigation and balancing different objectives, such as collecting objects and avoiding enemies. This matches our results in the main text that show that T-REX is unable to learn a policy for playing Hero, a similar maze navigation task with multiple objectives such as blowing up walls, rescuing people, and destroying enemies. Extending T-REX to work in these types of settings is an interesting area of future work.

F.2. Human rankings

To measure the effects of human ranking noise, we took the same 12 PPO demonstrations described above in the main text and had humans rank the demonstrations. We used Amazon Mechanical Turk and showed the workers two side-by-side demonstrations and asked them to classify whether video A or video B had better performance or whether they were unsure.

We took all 132 possible sequences of two videos across the 12 demonstrations and collected 6 labels for each pair of demonstrations. Because the workers are not actually giving the demonstrations and because some workers may exploit the task by simply selecting choices at random, we expect these labels to be a worst-case lower bound on the accuracy. To ameliorate the noise in the labels we take all 6 labels per pair and use the majority vote as the human label. If there is no majority or if the majority selects the "Not Sure" label, then we do not include this pair in our training data for T-REX.

The resulting accuracy and number of labels that had a majority preference are shown in Table 4. We ran T-REX using the same hyperparameters described in the main text. We ran PPO with 3 different seeds and report the performance of the best final policy averaged over 30 trials. We found that surprisingly, T-REX is able to optimize good policies for many of the games, despite noisy labels. However, we did find cases such as Enduro, where the labels were too noisy to allow successful policy learning.

G. Atari Reward Visualizations

We generated attention maps for the learned rewards for the Atari domains. We use the method proposed by Greydanus et al. (2018), which takes a stack of 4 frames and passes a 3x3 mask over each of the frames with a stride of 1. The mask is set to be the default background color for each game. For each masked 3x3 region, we compute the absolute difference in predicted reward when the 3x3 region is not masked and when it is masked. This allows us to measure the influence of different regions of the image on the predicted reward. The sum total of absolute changes in reward for each pixel is used to generate an attention heatmap. We used the trajectories shown in the extrapolation plots in Figure 4 of the main text and performed a search using the learned reward function to find the observations with minimum and maximum predicted reward. We show the minimum and maximum observations (stacks of four frames) along with the attention heatmaps across all four stacked frames for the learned reward functions in figures 2–9. The reward function visualizations suggest that our networks are learning relevant features of the reward function.

Table 2. Best demonstrations and average performance of learned policies for T-REX (ours) and DQfD with active preference learning (DQfD+A) (see Ibarz et al. (2018) Appendix A.2 and G). Results for T-REX are the best performance over 3 random seeds averaged over 30 trials. Results that exceed the best demonstration are marked with an asterisk (*). Note that T-REX requires at most only 66 pair-wise preference labels ($n(n-1)/2$ for $n = 12$ demonstrations), whereas DQfD+A uses between 4–7 demonstrations along with 3.4K labels queried during policy learning. DQfD+A requires action labels on the demonstrations, whereas T-REX learns from observation.

	Best Demonstration Received		Average Algorithm Performance	
Game	DQfD+A	T-REX	DQfD+A	T-REX
Beam Rider	19,844	1,188	4,100	*3,335.7
Breakout	79	33	*85	*221.3
Enduro	803	84	*1200	*586.8
Hero	99,320	13,235	35,000	0.0
Montezuma's Revenge	34,900	-	3,000	-
Pong	0	-6	*19	*-2.0
Private Eye	74,456	-	52,000	-
Q*bert	99,450	800	14,000	*32,345.8
Seaquest	101,120	600	500	*747.3
Space invaders	-	600	-	*1,032.5

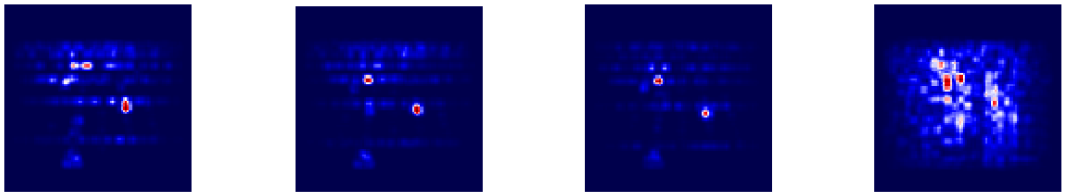
Table 3. T-REX performance with real novice human demonstrations collected from the Atari Grand Challenge Dataset (Kurin et al., 2017). Results are the best average performance over 3 random seeds with 30 trials per seed.

	Novice Human		
Game	Best	Average	T-REX
Montezuma's Revenge	**2,600**	1,275.0	0.0
Ms Pacman	**1,360**	818.3	550.7
Q*bert	875	439.6	**6,869.2**
Space Invaders	470	290.0	**1,092.0**
Video Pinball	4,210	2,864.3	**20,000.2**

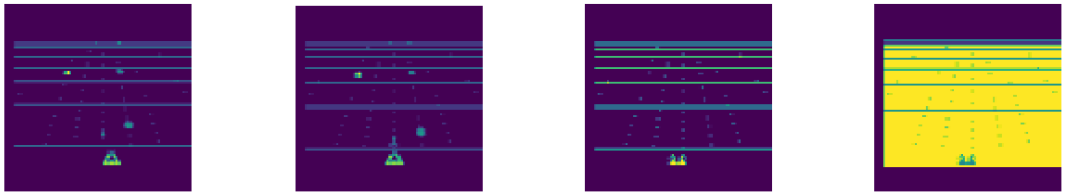
Table 4. Evaluation of T-REX on human rankings collected using Amazon Mechanical Turk. Results are the best average performance over 3 random seeds with 30 trials per seed.

	Human-Ranked Demonstrations				
Game	Best	Average	Ranking Accuracy	Num. Labels	T-REX avg. perf.
Beam Rider	1,332	686.0	63.0%	54	**3,457.2**
Breakout	32	14.5	88.1%	59	**253.2**
Enduro	**84**	39.8	58.6%	58	0.03
Hero	**13,235**	6742	77.6%	58	2.5
Pong	**-6**	-15.6	79.6%	54	-13.0
Q*bert	800	627	75.9%	58	**66,082**
Seaquest	600	373.3	80.4%	56	**655.3**
Space Invaders	600	332.9	84.7%	59	**1,005.3**

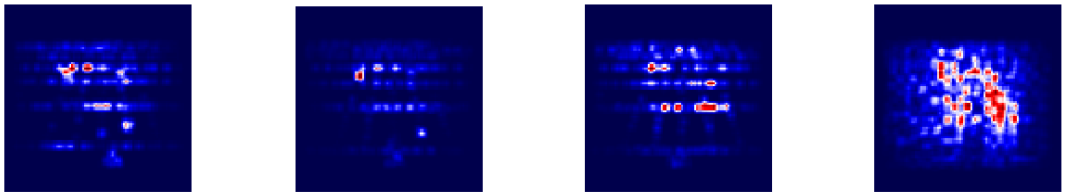
(a) Beam Rider observation with maximum predicted reward



(b) Beam Rider reward model attention on maximum predicted reward

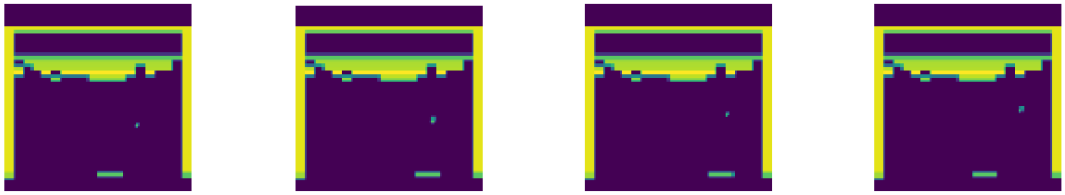


(c) Beam Rider observation with minimum predicted reward

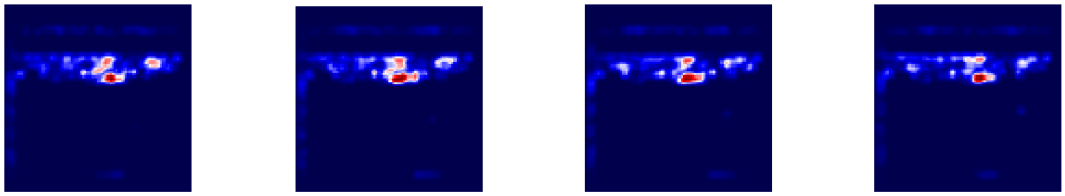


(d) Beam Rider reward model attention on minimum predicted reward

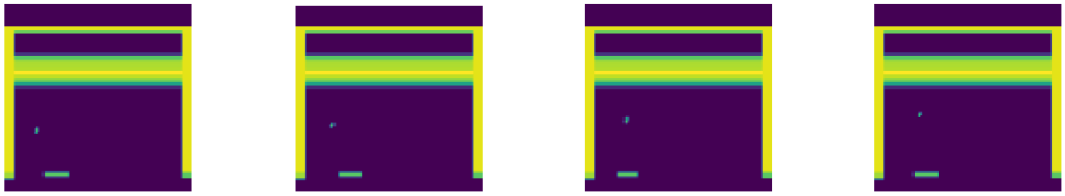
Figure 2. Maximum and minimum predicted observations and corresponding attention maps for Beam Rider. The observation with the maximum predicted reward shows successfully destroying an enemy ship, with the network paying attention to the oncoming enemy ships and the shot that was fired to destroy the enemy ship. The observation with minimum predicted reward shows an enemy shot that destroys the player's ship and causes the player to lose a life. The network attends most strongly to the enemy ships but also to the incoming shot.



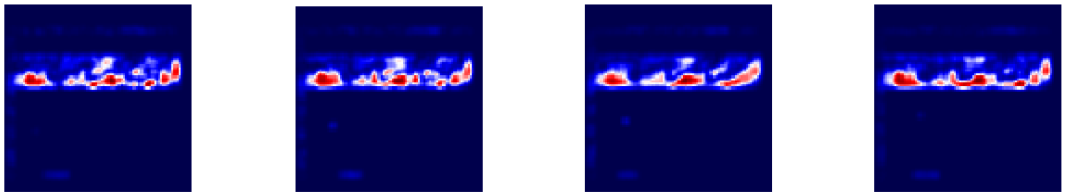
(a) Breakout observation with maximum predicted reward



(b) Breakout reward model attention on maximum predicted reward

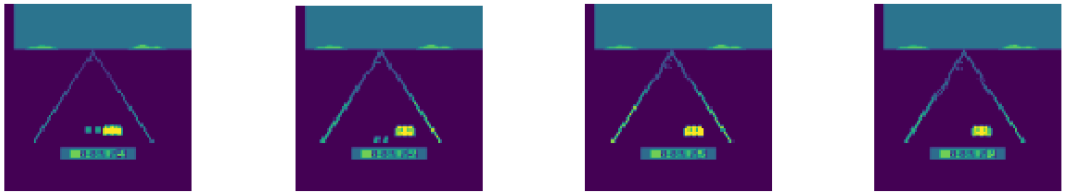


(c) Breakout observation with minimum predicted reward

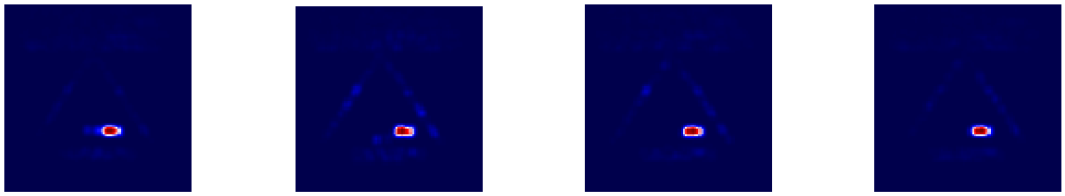


(d) Breakout reward model attention on minimum predicted reward

Figure 3. Maximum and minimum predicted observations and corresponding attention maps for Breakout. The observation with maximum predicted reward shows many of the bricks destroyed with the ball on its way to hit another brick. The network has learned to put most of the reward weight on the remaining bricks with some attention on the ball and paddle. The observation with minimum predicted reward is an observation where none of the bricks have been destroyed. The network attention is focused on the bottom layers of bricks.

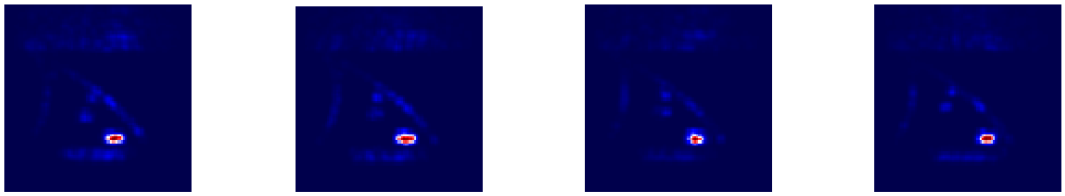


(a) Enduro observation with maximum predicted reward



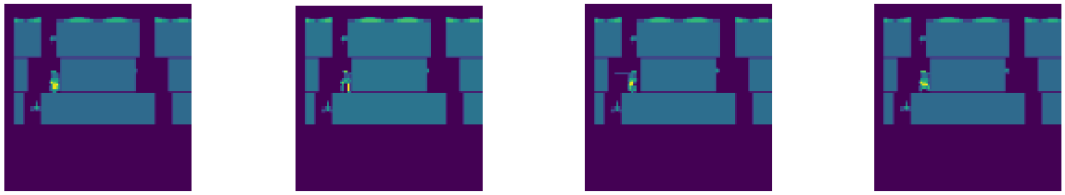
(b) Enduro reward model attention on maximum predicted reward

(c) Enduro observation with minimum predicted reward

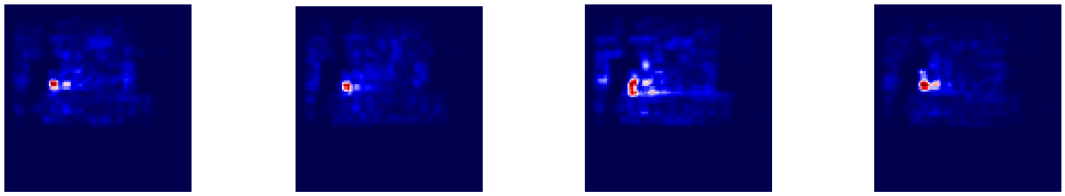


(d) Enduro reward model attention on minimum predicted reward

Figure 4. Maximum and minimum predicted observations and corresponding attention maps for Enduro. The observation with maximum predicted reward shows the car passing to the right of another car. The network has learned to put attention on the controlled car as well as the sides of the road with some attention on the car being passed and on the odometer. The observation with minimum predicted reward shows the controlled car falling behind other racers, with attention on the other cars, the odometer, and the controlled car.

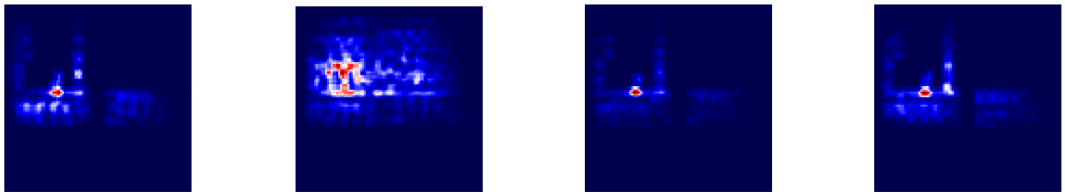


(a) Hero observation with maximum predicted reward



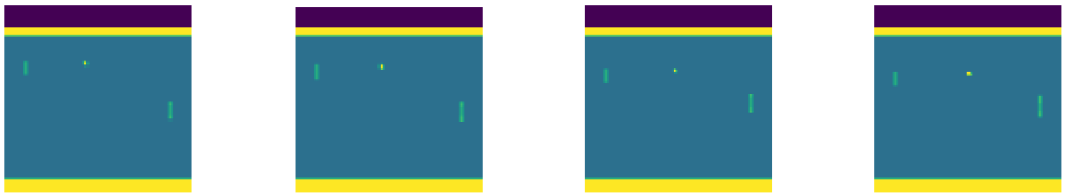
(b) Hero reward model attention on maximum predicted reward

(c) Hero observation with minimum predicted reward

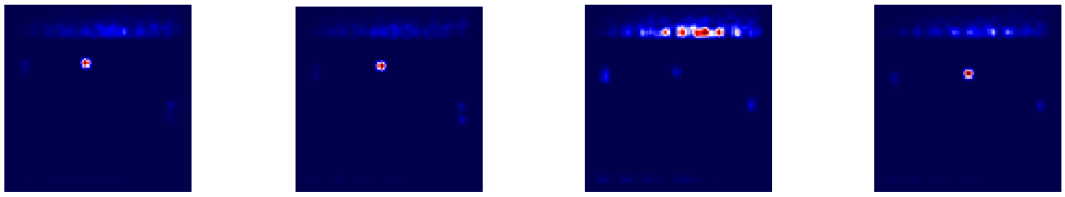


(d) Hero reward model attention on minimum predicted reward

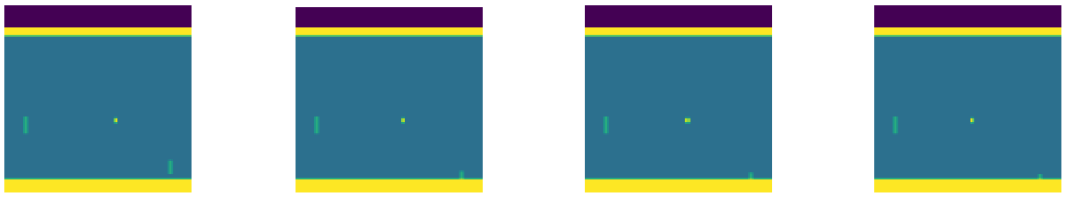
Figure 5. Maximum and minimum predicted observations and corresponding attention maps for Hero. The observation with maximum predicted reward is difficult to interpret, but shows the network attending to the controllable character and the shape of the surrounding maze. The observation with minimum predicted reward shows the agent setting off a bomb that kills the main character rather than the wall. The learned reward function attends to the controllable character, the explosion and the wall that was not destroyed.



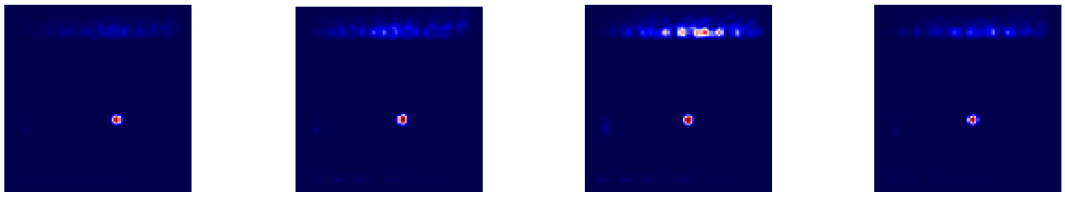
(a) Pong observation with maximum predicted reward



(b) Pong reward model attention on maximum predicted reward

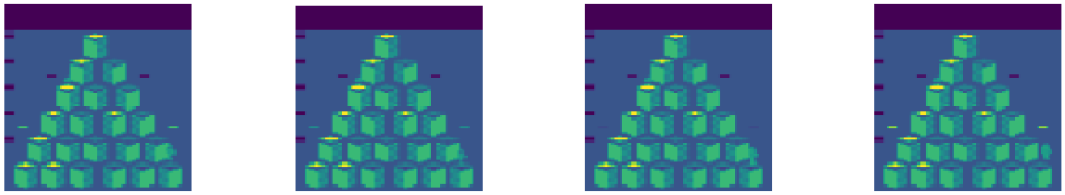


(c) Pong observation with minimum predicted reward

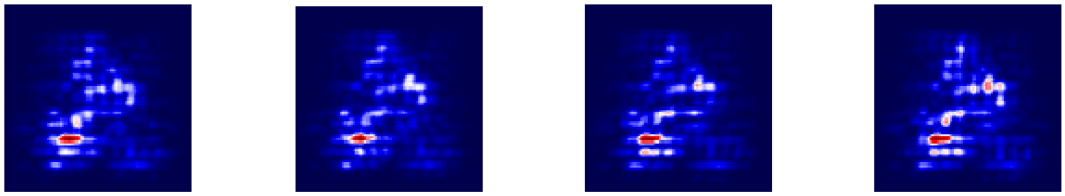


(d) Pong reward model attention on minimum predicted reward

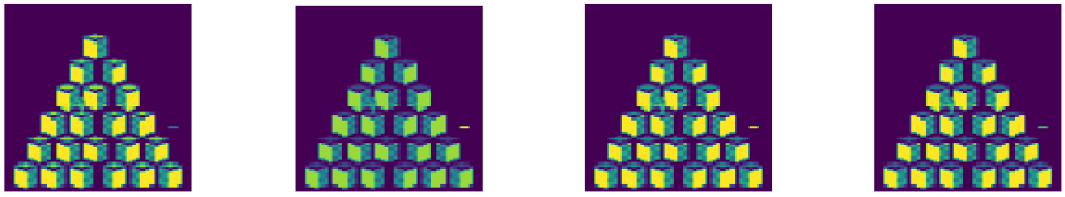
Figure 6. Maximum and minimum predicted observations and corresponding attention maps for Pong. The network mainly attends to the ball, with some attention on the paddles.



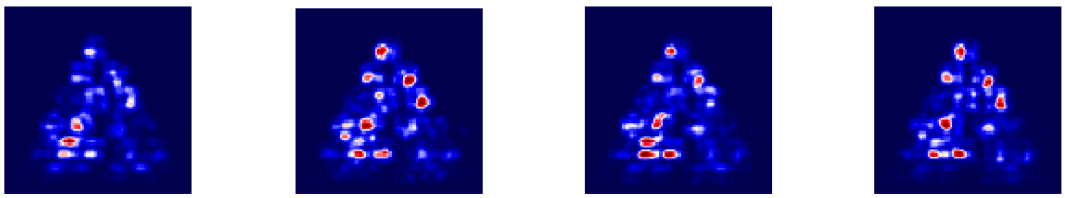
(a) Q*bert observation with maximum predicted reward



(b) Q*bert reward model attention on maximum predicted reward

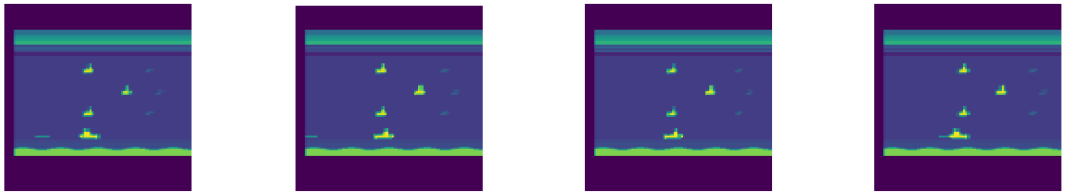


(c) Q*bert observation with minimum predicted reward

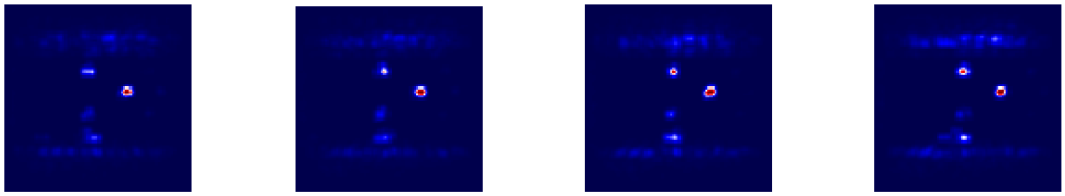


(d) Q*bert reward model attention on minimum predicted reward

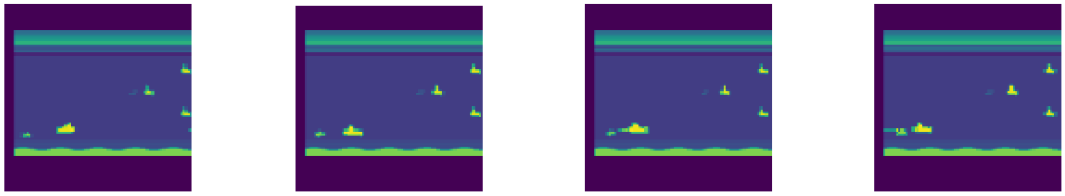
Figure 7. Maximum and minimum predicted observations and corresponding attention maps for Q*bert. The observation for the maximum predicted reward shows an observation from the second level of the game where stairs change color from yellow to blue. The observation for the minimum predicted reward is less interpretable. The network attention is focused on the different stairs, but it is difficult to attribute any semantics to the attention maps.



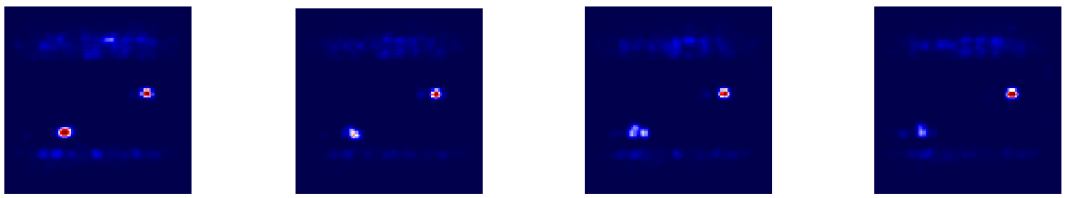
(a) Seaquest observation with maximum predicted reward



(b) Seaquest reward model attention on maximum predicted reward

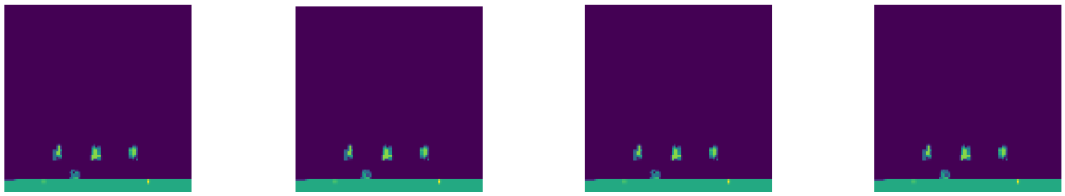


(c) Seaquest observation with minimum predicted reward

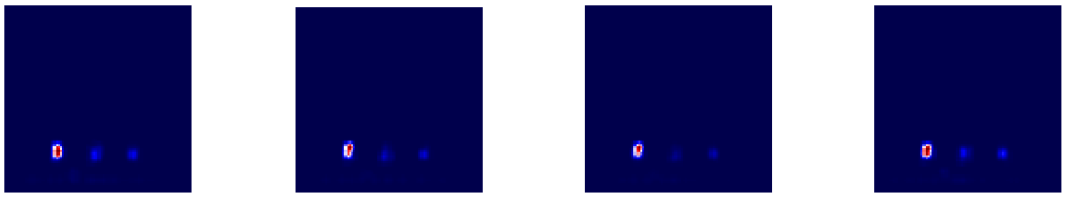


(d) Seaquest reward model attention on minimum predicted reward

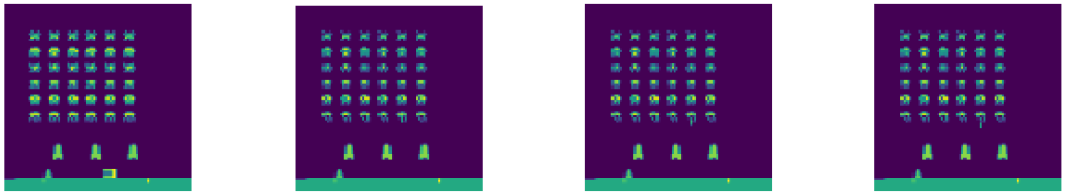
Figure 8. Maximum and minimum predicted observations and corresponding attention maps for Seaquest. The observation with maximum predicted reward shows the submarine in a relatively safe area with no immediate threats. The observation with minimum predicted reward shows an enemy that is about to hit the submarine—the submarine fires a shot, but misses. The attention maps show that the network focuses on the nearby enemies and also on the controlled submarine.



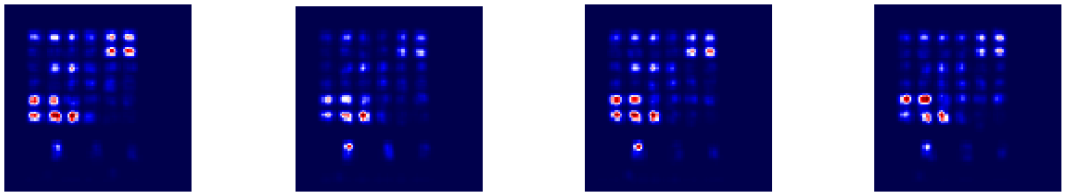
(a) Space Invaders observation with maximum predicted reward



(b) Space Invaders reward model attention on maximum predicted reward



(c) Space Invaders observation with minimum predicted reward



(d) Space Invaders reward model attention on minimum predicted reward

Figure 9. Maximum and minimum predicted observations and corresponding attention maps for Space Invaders. The observation with maximum predicted reward shows an observation where all the aliens have been successfully destroyed and the protective barriers are still intact. Note that the agent never observed a demonstration that successfully destroyed all the aliens. The attention map shows that the learned reward function is focused on the barriers, but does not attend to the location of the controlled ship. The observation with minimum predicted reward shows the very start of a game with all aliens still alive. The network attends to the aliens and barriers, with higher weight on the aliens and barrier closest to the space ship.