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 time-stamps 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.

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 mag-

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 time-stamp when they were produced by the PPO algorithm learning to perform the task.

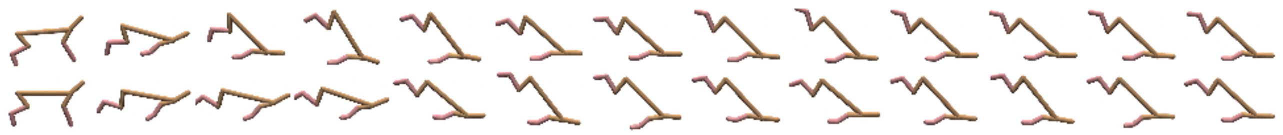
	HalfCheetah			Hopper			Ant	
	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2
Best Demo Performance	12.52 (1.04)	44.98 (0.60)	89.87 (8.15)	3.70 (0.01)	5.40 (0.12)	7.95 (1.64)	1.56 (1.28)	**54.64** (22.09)
T-REX (ours)	46.90 (1.89)	**61.56** (10.96)	143.40 (3.84)	**15.13** (3.21)	10.10 (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)	

nitude 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.

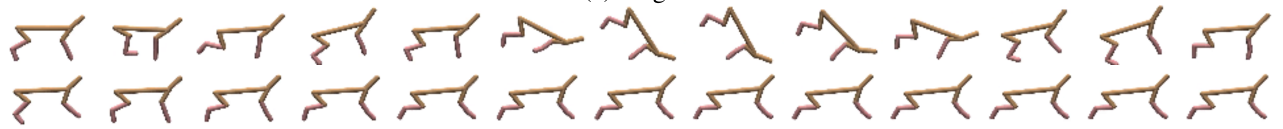
F. Human Demonstrations and Rankings

F.1. Human demonstrations

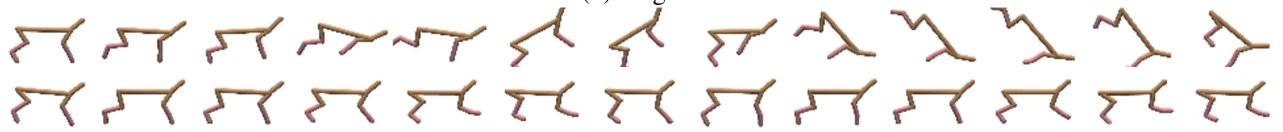
We used the Atari Grand Challenge data set (Kurin et al., 2017) to collect actual human demonstrations for five Atari games. We used the ground truth returns in the Atari Grand Challenge data set to rank demonstrations. To generate demonstrations we removed duplicate demonstrations (human demonstrations 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.



(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.

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 had 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 pass a 3x3 all-zero mask over each of the frames with a stride of 1. 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 change in reward for each pixel is used to generate an attention heatmap. We used the trajectories shown in the extrapolation plots shown 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 the 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 also requires action labels on the demonstrations, whereas T-REX learns purely from observation.

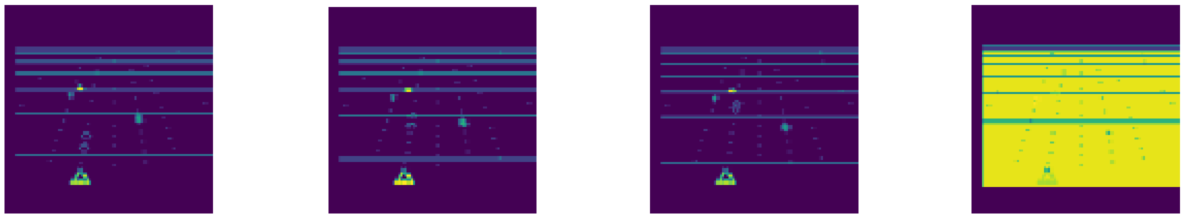
	Best Demonstration Received		Average Algorithm Performance	
Game	DQfD+A	T-REX	DQfD+A	T-REX
Beam Rider	19,844	1,188	4,100	*3,335.7
Breakout	79	33	*85	*221.3
Enduro	803	84	*1200	*586.8
Hero	99,320	13,235	35,000	0.0
Montezuma's Revenge	34,900	-	3,000	-
Pong	0	-6	*19	*-2.0
Private Eye	74,456	-	52,000	-
Q*bert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

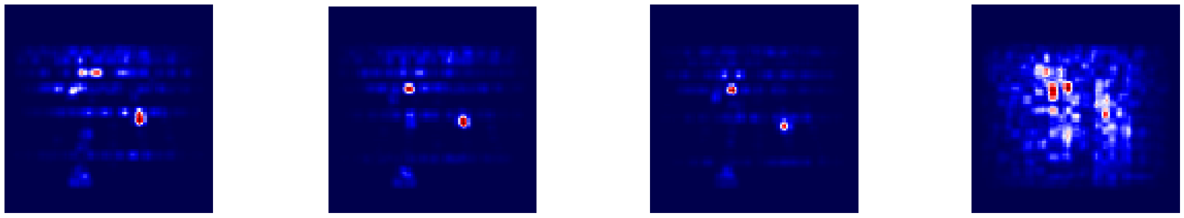
	Novice Human		
Game	Best	Average	T-REX
Montezuma's Revenge	**2,600**	1,275.0	0.0
Ms Pacman	**1,360**	818.3	550.7
Q*bert	875	439.6	**6,869.2**
Space Invaders	470	290.0	**1,092.0**
Video Pinball	4,210	2,864.3	**20,000.2**

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.

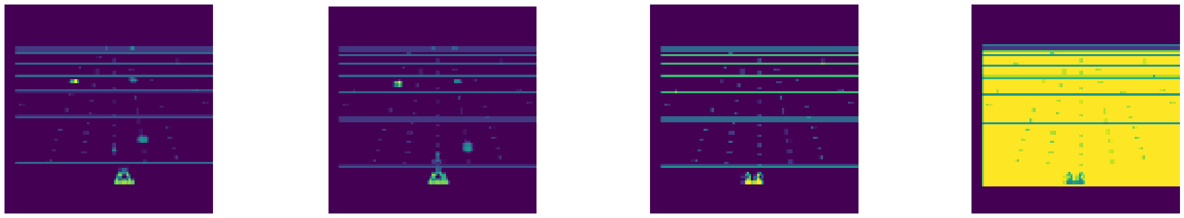
	Human-Ranked Demonstrations				
Game	Best	Average	Ranking Accuracy	Num. Labels	T-REX avg. perf.
Beam Rider	1,332	686.0	63.0%	54	**3,457.2**
Breakout	32	14.5	88.1%	59	**253.2**
Enduro	**84**	39.8	58.6%	58	0.03
Hero	**13,235**	6742	77.6%	58	2.5
Pong	**-6**	-15.6	79.6%	54	-13.0
Q*bert	800	627	75.9%	58	**66,082**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



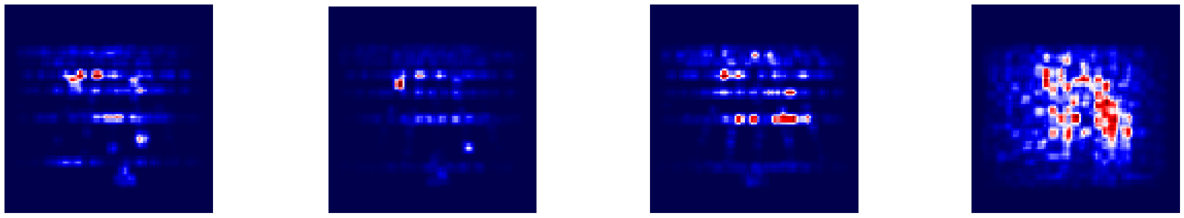
(a) Beam Rider observation with maximum predicted reward



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

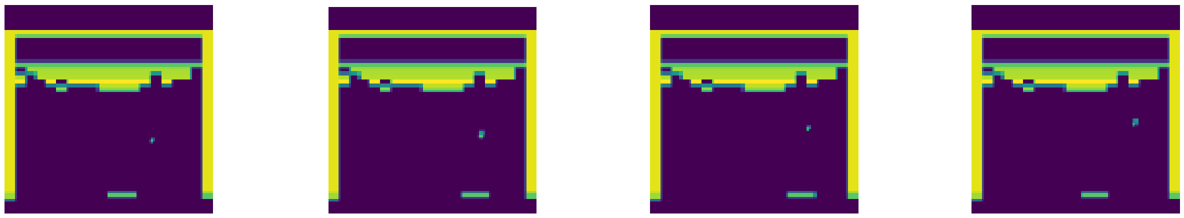


(c) Beam Rider observation with minimum predicted reward

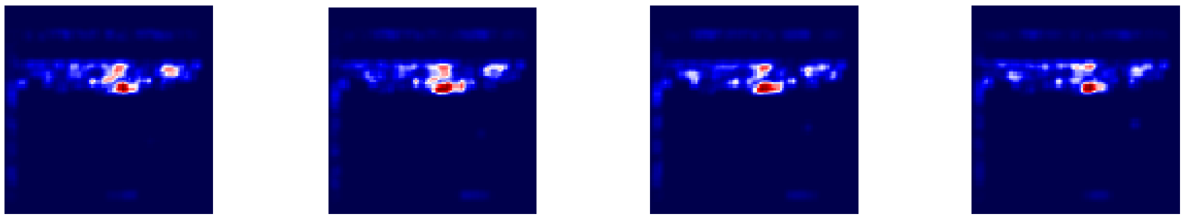


(d) Beam Rider reward model attention on minimum predicted reward

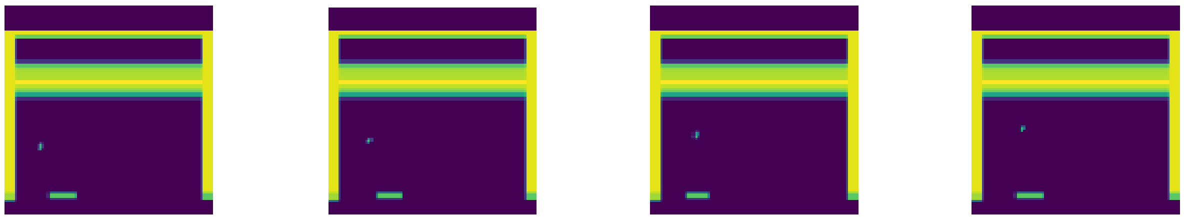
Figure 2. Maximum and minimum predicted observations and corresponding attention maps for Beam Rider. The observation with the maximum predicted reward shows successfully destroying an enemy ship, with the network paying attention to the oncoming enemy ships and the shot that was fired to destroy the 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.



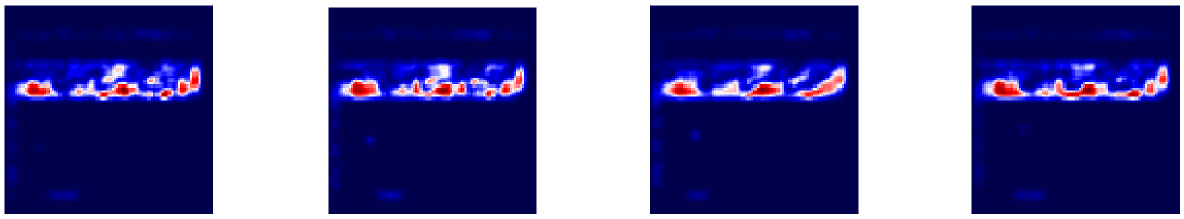
(a) Breakout observation with maximum predicted reward



(b) Breakout reward model attention on maximum predicted reward

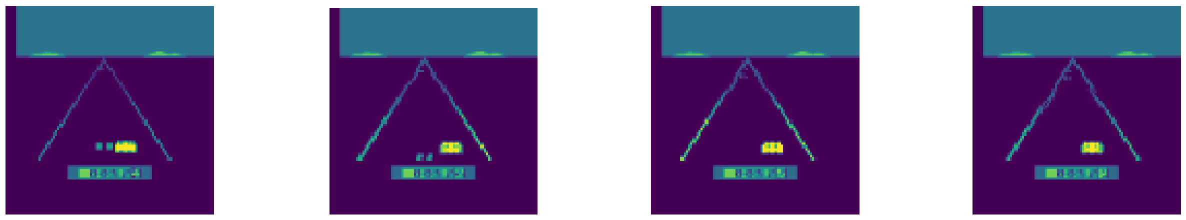


(c) Breakout observation with minimum predicted reward

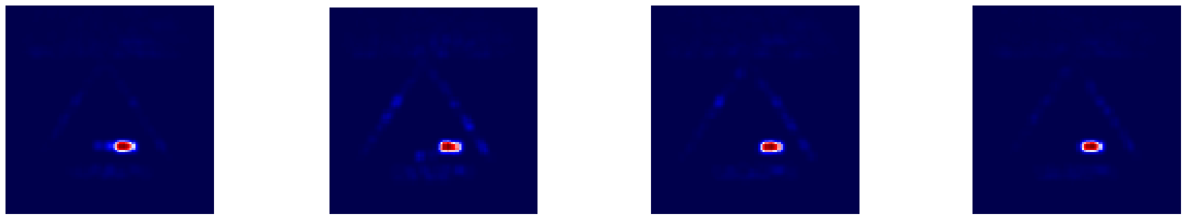


(d) Breakout reward model attention on minimum predicted reward

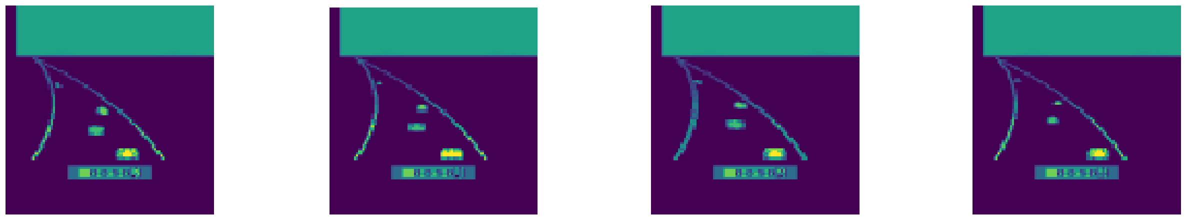
Figure 3. Maximum and minimum predicted observations and corresponding attention maps for Breakout. The observation with maximum predicted reward shows many of the bricks destroyed with the ball on its way to hit another brick. The network has learned to put most of the reward weight on the remaining 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.



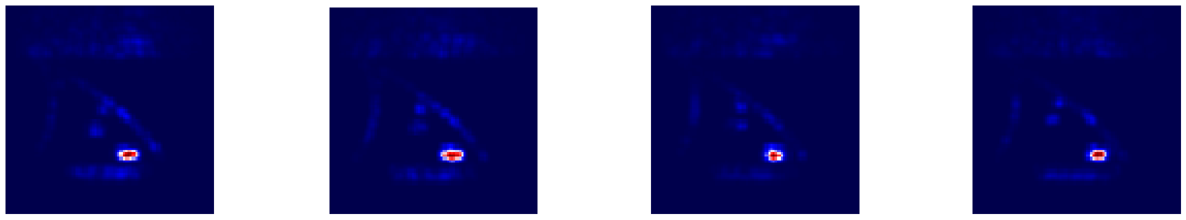
(a) Enduro observation with maximum predicted reward



(b) Enduro reward model attention on maximum predicted reward

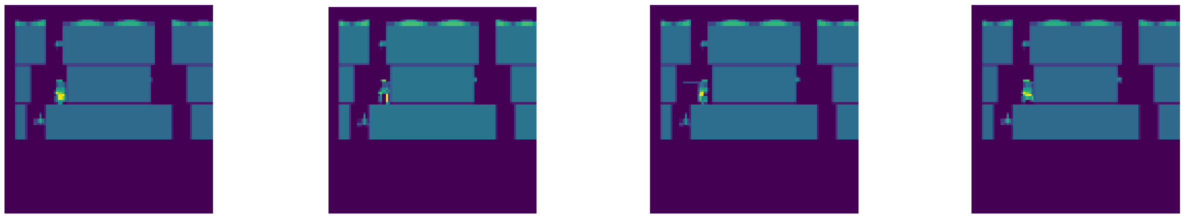


(c) Enduro observation with minimum predicted reward

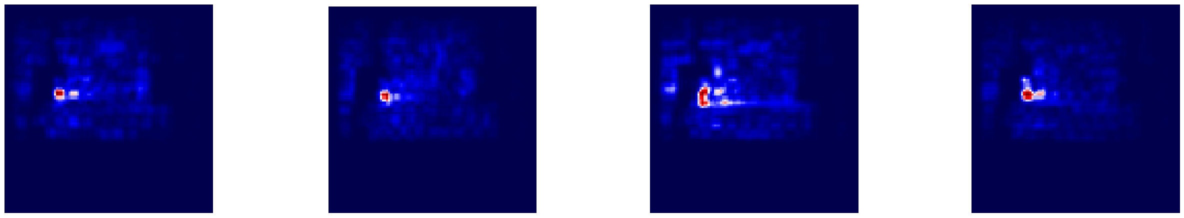


(d) Enduro reward model attention on minimum predicted reward

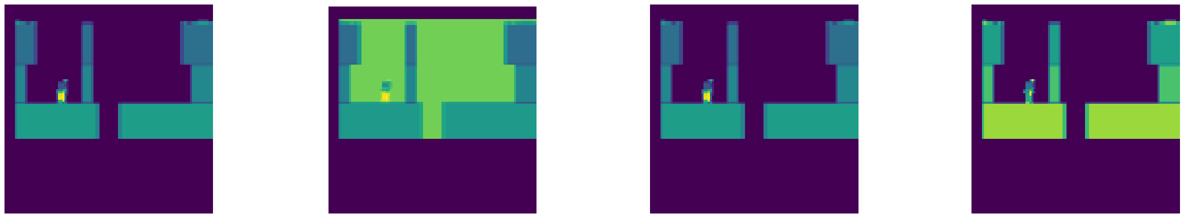
Figure 4. Maximum and minimum predicted observations and corresponding attention maps for Enduro. The observation with maximum predicted reward shows the car passing to the right of another car. The network has learned to put attention on the controlled car as well as the sides of the road with some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



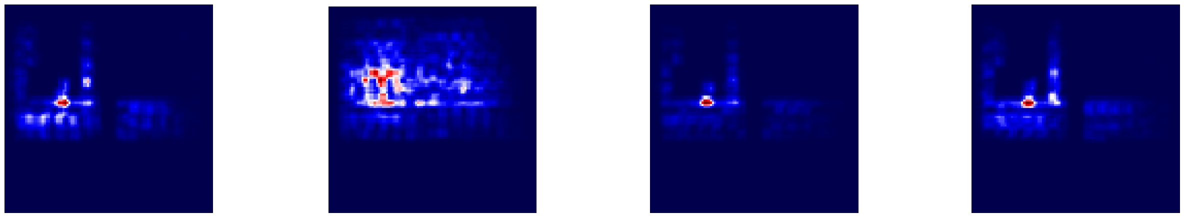
(a) Hero observation with maximum predicted reward



(b) Hero reward model attention on maximum predicted reward

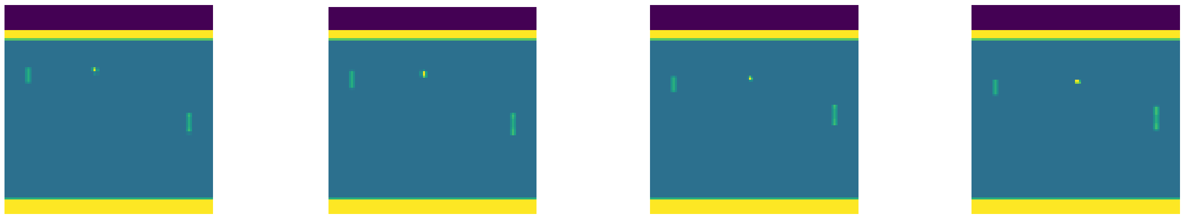


(c) Hero observation with minimum predicted reward

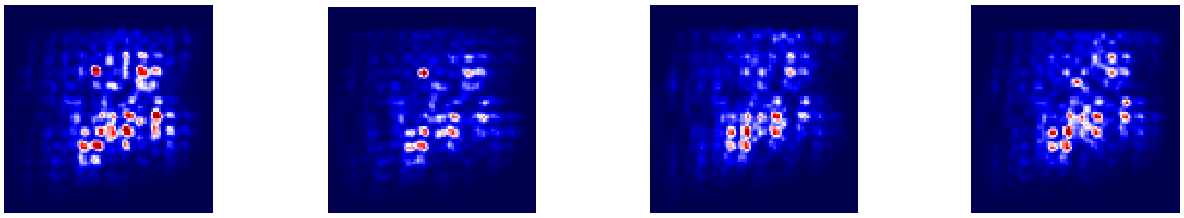


(d) Hero reward model attention on minimum predicted reward

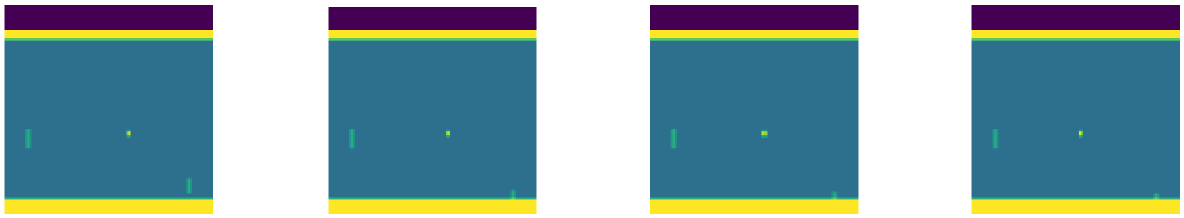
Figure 5. Maximum and minimum predicted observations and corresponding attention maps for Hero. The observation with maximum predicted reward is difficult to interpret, but shows the network attending to the controllable character and the shape of the surrounding maze. The observation with minimum predicted 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.



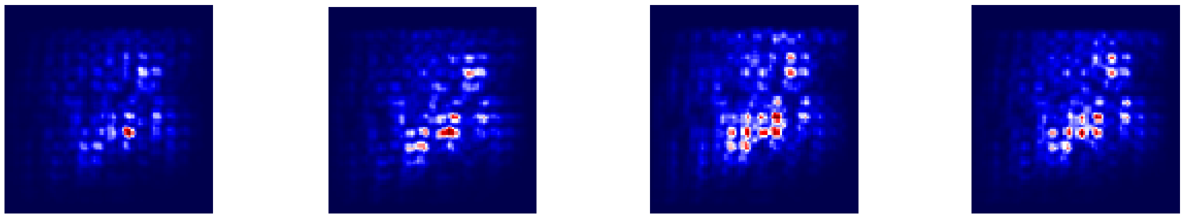
(a) Pong observation with maximum predicted reward



(b) Pong reward model attention on maximum predicted reward

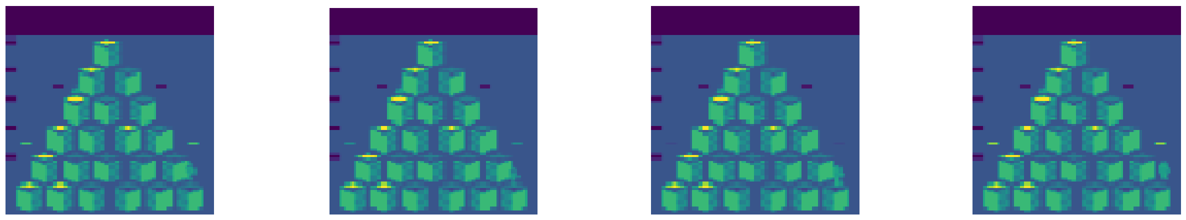


(c) Pong observation with minimum predicted reward

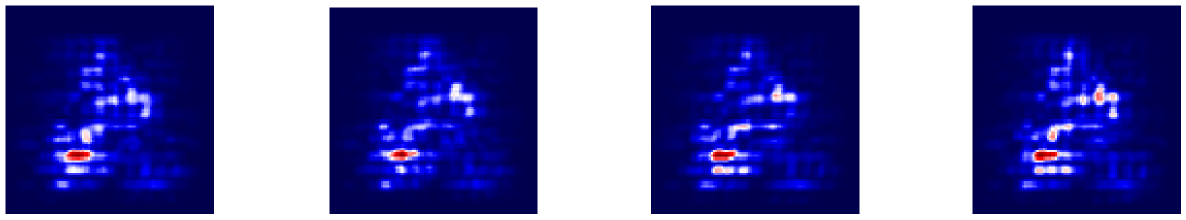


(d) Pong reward model attention on minimum predicted reward

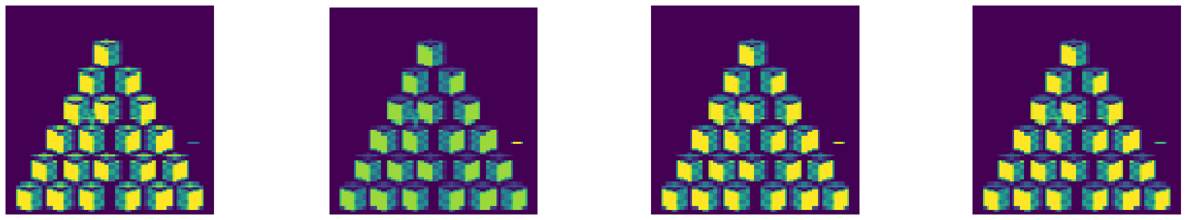
Figure 6. Maximum and minimum predicted observations and corresponding attention maps for Pong. It is unclear what the network is paying attention to or whether there are any interesting semantics in the observations with minimum and maximum predicted reward. It appears that the network has latched onto certain possible ball locations that may be indicative of a good or bad rally.



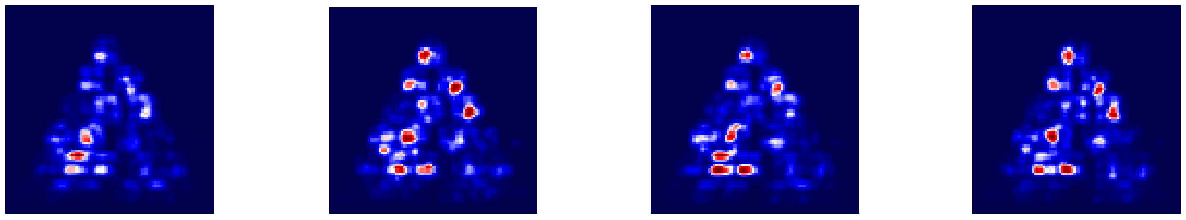
(a) Q*bert observation with maximum predicted reward



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

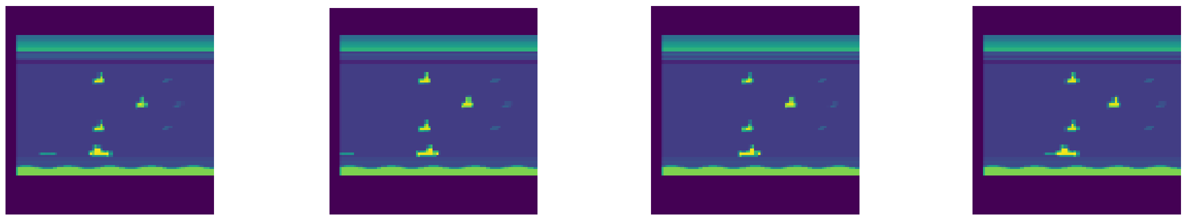


(c) Q*bert observation with minimum predicted reward

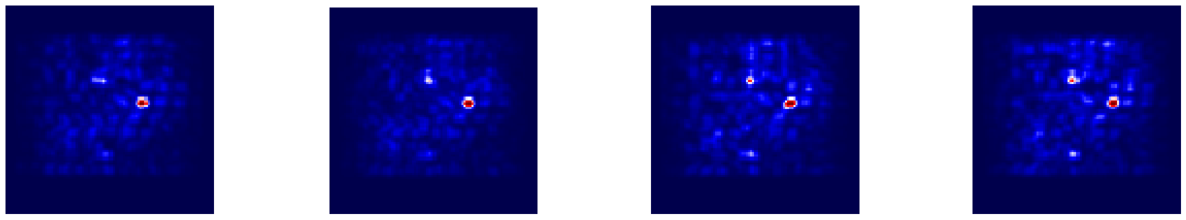


(d) Q*bert reward model attention on minimum predicted reward

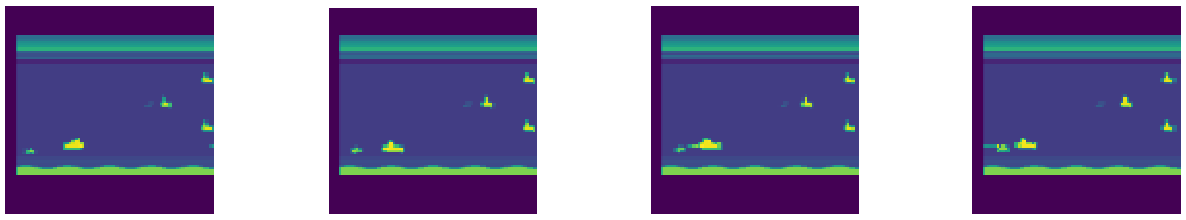
Figure 7. Maximum and minimum predicted observations and corresponding attention maps for Q*bert. The observation for the maximum predicted reward shows an observation from the second level of the game where stairs change color from yellow to blue. The observation for the minimum predicted reward is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



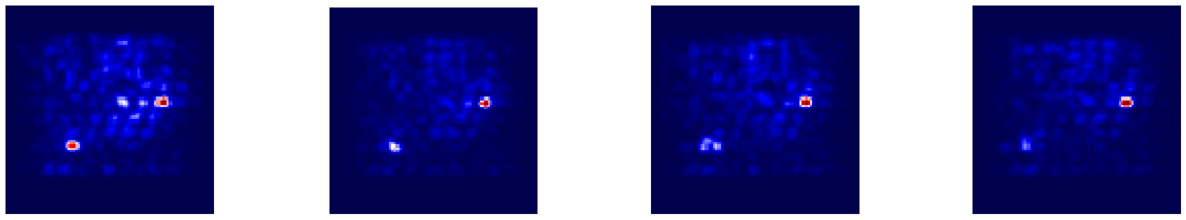
(a) Seaquest observation with maximum predicted reward



(b) Seaquest reward model attention on maximum predicted reward

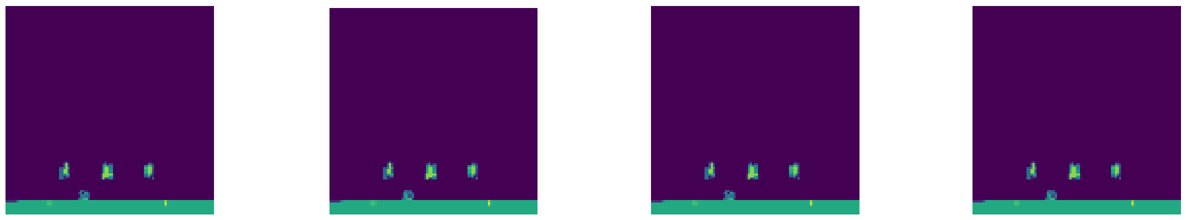


(c) Seaquest observation with minimum predicted reward

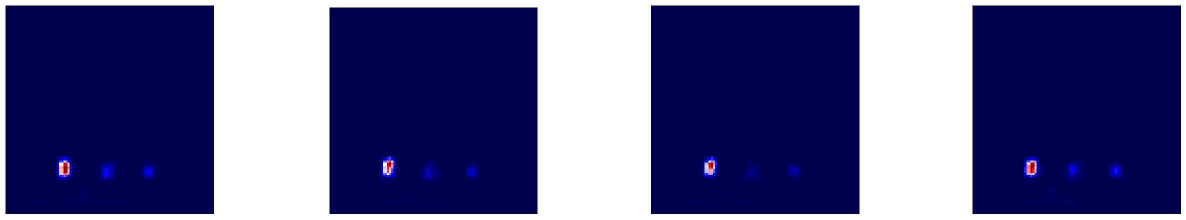


(d) Seaquest reward model attention on minimum predicted reward

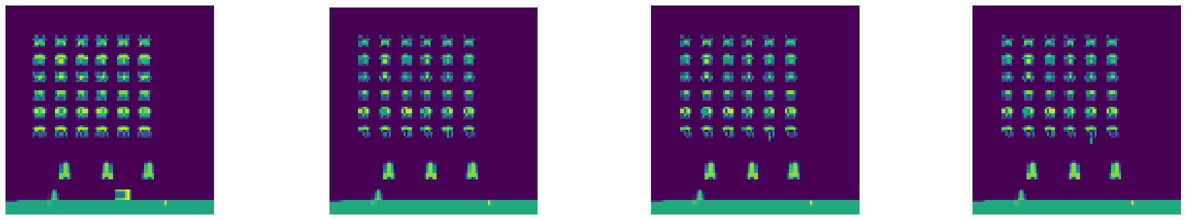
Figure 8. Maximum and minimum predicted observations and corresponding attention maps for Seaquest. The observation with maximum predicted reward shows the submarine in a relatively safe area with no immediate threats. The observation with minimum predicted reward shows an enemy that is about to hit the submarine and the submarine firing a shot. The attention map for the maximum predicted reward is focused on the enemies, since there are no immediate threats. On the other hand, the attention map for the minimum predicted reward focuses on the controlled submarine, but this attention decreases once the submarine has fired a shot at the approaching enemy.



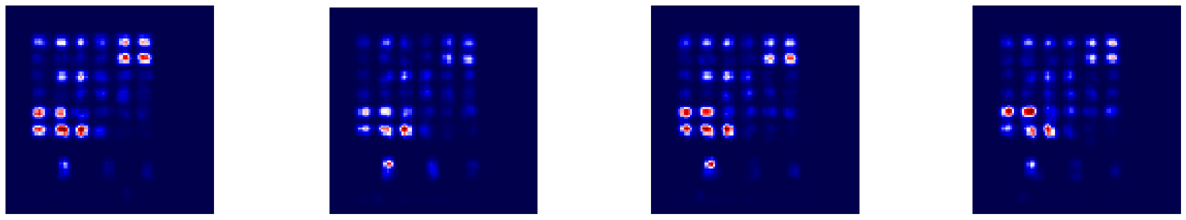
(a) Space Invaders observation with maximum predicted reward



(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 not attending 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.