

---

# Trial without Error: Towards Safe Reinforcement Learning via Human Intervention

---

**William Saunders**  
University of Oxford

**Girish Sastry**  
University of Oxford

**Andreas Stuhlmüller**  
Stanford University

**Owain Evans**  
University of Oxford

## Abstract

AI systems are increasingly applied to complex tasks that involve interaction with humans. During training, such systems are potentially dangerous, as they haven't yet learned to avoid actions that could cause serious harm. How can an AI system explore and learn without making a *single* mistake that harms humans or otherwise causes serious damage? For model-free reinforcement learning, having a human "in the loop" and ready to intervene is currently the only way to prevent all catastrophes. We formalize human intervention and show how to reduce the human labor for intervention by training a supervised learner to imitate the intervention decisions. We evaluate our scheme on deep RL agents playing Atari games. While human oversight can prevent all catastrophes in Pong and Space Invaders, and can be automated robustly after four hours of human supervision, small variations in the quality of human supervision affect success in the game of Road Runner (where the agent has a perverse incentive to fool the supervisor). We extrapolate to more complex environments and show that our simple approach would not scale up (due to the infeasible amount of human labor required). We outline extensions of our approach that are necessary if we are to train model-free agents safely.

[Link to videos](#) that illustrate our approach on Atari games.

## 1 Introduction

### 1.1 Motivation

AI systems are increasingly applied to complex tasks that involve interaction with humans. During training, such systems are potentially dangerous, as they haven't yet learned to avoid actions that would cause serious harm. How can an AI system explore and learn without making a *single* mistake that harms humans, destroys property, or damages the environment?

A crucial safeguard against this danger is *human intervention*. Self-driving cars are overseen by human drivers, who take control when they predict the AI system will perform badly. These overseers frequently intervene, especially in self-driving systems at an early stage of development [11]. We use the same safeguard for human learners, who are overseen by a licensed driver.

Many AI systems pose no *physical* danger to humans. Yet web-based systems can still cause unintended harm. Microsoft's chatbot Tay reproduced thousands of offensive tweets before being taken down [29]. Facebook's algorithms for sharing news stories inadvertently provided a platform for malicious and false stories and disinformation during the US 2016 election [3]. If human operators had monitored these systems in real-time (as with self-driving cars), the bad outcomes could have been avoided.

Human oversight is currently the only means of avoiding *all* accidents in complex real-world domains. How does human intervention for safety fit together with Deep Learning and Reinforcement Learning, which are likely to be key components of future applied AI systems? We investigate a scheme for

human intervention in RL systems in the Atari domain, document serious scalability challenges, and outline potential remedies.

## 1.2 Contributions

We provide a formal scheme (HIRL) for applying human oversight to an RL agent. The scheme makes it easy to train a supervised learner to imitate the human’s intervention policy and take over from the human. (Automating human oversight is crucial since it’s infeasible for a human to watch over an RL agent for 100 million timesteps.) While the human oversees a *particular* RL agent, the supervised learner can be used as a safety-harness for different agents.<sup>1</sup>

The goal of HIRL is to enable an RL agent to learn a task in the real world without a single catastrophe. We investigated the scalability of HIRL by applying it to Atari games, which are challenging toy environments for current AI [20]. We applied HIRL to Deep RL agents playing three games (Pong, Space Invaders, Road Runner). For the first 4.5 hours of training, a human watched every frame and intervened to prevent the agent taking catastrophic actions. The supervised learner (which had been trained to imitate the human) was then able to prevent all catastrophes in Pong and Space Invaders without preventing the agent from learning the game. In Road Runner, where the class of catastrophes was more diverse and complex, HIRL reduced catastrophes by a factor of 50 but did not eliminate them.

Our results show that it is possible for Deep RL agents to learn to play some Atari games from scratch without causing a single catastrophe. We also describe some key challenges for scaling HIRL. When faced with an adversarial RL agent (as in Road Runner), the supervised learner needs to be robust to adversarial distribution shift [2]. Additional techniques are needed to bring down the amount of human labor required. Using a formula for the quantity of human labor required by HIRL, we show that slightly different Atari environments would require 10 or 100 times the human labor. We outline a range of techniques that would bring down human labor costs.

## 2 HIRL: A Scheme for Safe RL via Human Intervention

### 2.1 Motivation for HIRL

Can RL agents learn *safely* in real-world environments? The existing literature contains a variety of definitions of “safe RL” [12]. In this paper, we say an RL agent is safe if it never takes “catastrophic actions” during training. We define “catastrophic actions” as actions that the human overseer deems unacceptable under any circumstances (even at the start of training). So we avoid formalizing “catastrophes” and let the human supervisor specify them (as in [15]). The overseer will typically distinguish *sub-optimal* actions from catastrophic actions. It is tolerable for a car to drive slowly during learning; but hitting pedestrians is catastrophic and must be avoided from the very start of training.

Reinforcement learning alone is insufficient to achieve this kind of safety. The fundamental problem is that RL learns by trial and *error*. Without prior knowledge, a model-free RL agent will not avoid a catastrophic action unless it has tried the action (or a similar action) and learned from the negative experience.<sup>2</sup>

This problem could potentially be side-stepped by training in simulation [10]. The agent explores dangerous actions in simulation and transfers this knowledge to the real world [8]. To work reliably, this would require advances in transfer learning and in simulation. Yet simulating humans accurately is infeasible for many tasks<sup>3</sup> and tasks involving human interaction are the most safety-critical.

Imitation learning can be used to learn a safe initial policy from human demonstrations [16]. While the initial policy will be much safer than random initialization, any deviation between the human and

---

<sup>1</sup>The human does not need to spend time providing safety interventions every time they try a new agent architecture. This makes possible a typical workflow in which researchers explore different algorithms (e.g. DQN vs. A3C) for a task.

<sup>2</sup>Model-based agents have the same problem unless they have a very good prior on models. This paper focuses on the model-free case.

<sup>3</sup>It’s hard to simulate how a human would change their strategy in response to interaction with an AI system. This is no accident: simulating the strategic reasoning of humans would solve a major open problem in AI.

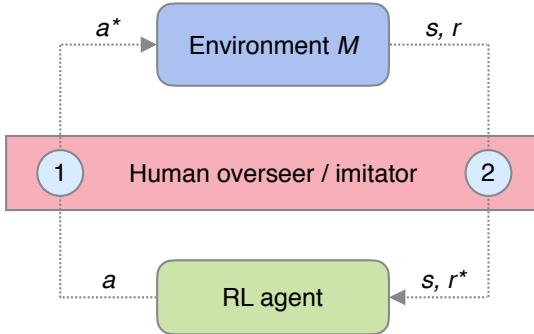


Figure 1: HIRL scheme. At (1) the human overseer (or Blocker imitating the human) can block/intercept unsafe actions  $a$  and replace them with safe actions  $a^*$ . At (2) the overseer can provide a negative reward  $r^*$  for the unsafe action (i.e. this is “reward shaping”).

the learned policy can result in unsafe actions, and subsequent fine-tuning of the policy using RL can introduce catastrophic behavior. So imitation learning is not sufficient on its own but could be valuable combined with HIRL. (Imitation learning is helpful for safe initialization when the human knows an easy-to-learn policy that performs well and steers well clear of dangerous regions of the state space.)

## 2.2 Formal Specification of HIRL

We model the RL agent’s environment as a Markov Decision Process (MDP). The *environment* is an MDP specified by a tuple  $M = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ , where  $\mathcal{S}$  is the state space,  $\mathcal{A}$  is the action space,  $\mathcal{T}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$  is the transition function,  $\mathcal{R}: \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$  is the reward function, and  $\gamma$  is the discount factor.

How can an RL agent learn while never taking a single catastrophic action? Our scheme, HIRL (Human Intervention RL), is simple. The human controls the interface between the RL agent and environment  $M$ , constantly watching over the agent and *blocking* any catastrophic actions before they happen. More precisely, at each timestep the human observes the current state  $s$  and the agent’s proposed action  $a$ . If  $(s, a)$  is catastrophic, the human sends a safe action  $a^*$  to the environment instead. The human also replaces the new reward  $r = \mathcal{R}(s, a^*)$  with a penalty  $r^*$  (Figure 1).

The period in which the human blocks the agent is called the “Human Oversight” phase of HIRL. During this phase, we store each state-action  $(s, a)$  and a binary label for whether or not the human blocked it. This dataset is used to train a “Blocker”, a classifier trained by supervised learning to imitate the human’s blocking decisions. The Human Oversight phase lasts until the Blocker performs well on a held-out subset of the training data. At this point, the human retires and the Blocker takes over for the rest of time. The Blocker *never* stops overseeing the agent, which prevents catastrophes even if the agent exhibits random exploration or catastrophic forgetting [19].

HIRL is *agnostic* as to the inner workings of the RL algorithm (building on our earlier work [1]). It works for Q-learning [21], for policy gradient algorithms like A3C [22] and for model-based RL [14]. Moreover, the Blocker that imitates the human overseer is *modular*. While trained on data from one agent, the Blocker can act as a safeguard for a completely different agent. However, since different RL agents may explore different parts of the state space, Blockers will not always transfer their safety properties.

## 3 The Time Cost of Human Intervention

When is it feasible to prevent catastrophes using HIRL? First, the Blocker (which imitates human interventions) needs to achieve near-perfect reliability in recognizing catastrophic actions, which may require a huge number of observations from the environment. Second, the environment must proceed slowly enough for the human to intervene. This rules out tasks like high-frequency trading or controlling vehicles at super-human speed. In environments where speed is a controllable parameter (e.g. computer tasks), slowing down the environment can make the RL agent’s learning too slow.



Figure 2: Pong. It's a catastrophe if the agent (green paddle) enters the Catastrophe Zone.



Figure 3: Road Runner. It's a catastrophe if Road Runner touches the Coyote.

### 3.1 Formula for Time Cost

To measure the feasibility of HIRL, we provide a formula for the time cost of human intervention. Our goal is that the probability of the RL agent ever taking a catastrophic action is vanishingly small. More precisely, suppose the RL agent runs for  $H$  episodes. These episodes include any mode of operation: both training and deployment (where the agent is no longer learning). Then the RL system is “catastrophe-safe” for  $H$  if the probability of catastrophe over the  $H$  episodes is close to zero (say  $10^{-9}$ ).

The human provides labels for state-action pairs  $(s, a)$  which are used to train the Blocker. The data-points have the form  $((s, a), +1)$  for *catastrophic* observations and  $((s, a), -1)$  for *safe* observations. The formula for the time-cost of human intervention (first in symbols, then in words) is:

$$C = t_{\text{human}} \times N_{\text{all}} \quad (1)$$

[ total time-cost = time per human label  $\times$  # observations to label ]

In this formula,  $t_{\text{human}}$ , is the average time it takes the human to label an observation. Since humans are intrinsically slow, we're stuck with a bound  $t_{\text{human}} > 10^{-1}$  seconds. So we focus on  $N_{\text{all}}$ , the number of observations the human has to label.

$N_{\text{all}}$  depends on how many datapoints are required to train the Blocker<sup>4</sup>. In order for the agent to be catastrophe-safe over  $H$  episodes, the Blocker must have a very low *false-negative* rate over duration  $H$ . This corresponds to an astronomically small false-negative rate for a single observation. Since we also want a reasonably low false-positive rate, the Blocker must have low test-set error, which typically requires a large value for  $N_{\text{all}}$  (the size of the training set). Switching to a more data-efficient classifier would reduce  $N_{\text{all}}$ .

How else can we reduce  $C$ ? Let  $\rho$  denote the ratio of all observations to catastrophe observations (averaged over the training period). In some cases, catastrophes are rarely encountered and so  $\rho > 1000$ . If  $\rho$  is reduced (e.g. by exploring more systematically), the class sizes for training the Blocker are less skewed and so a smaller training set will suffice (hence  $N_{\text{all}}$  is reduced).

We can re-write Formula 1 in terms of  $\rho$ . Training the Blocker requires a certain number of positive examples (i.e. examples of prevented catastrophes), which we denote  $N_{\text{cat}}$ . But to get that many positive examples, the human usually needs to label many more negative examples ( $\rho \gg 1$ ). So Formula 1 is re-written as:

$$C = t_{\text{human}} \times \rho \times N_{\text{cat}} \quad (2)$$

[ total time-cost = time per label  $\times$  (#observations / #cat-observations)  $\times$  #cat-observations ]

We revisit different approaches to reduce this time-cost in Section 5.2.

---

<sup>4</sup>Here we focus on the Blocker's decision on whether to block an action. We ignore the question of how the Blocker learns to replace the catastrophic action with a safe one, as it's beyond the scope of this paper.

## 4 Experiments

### 4.1 Design of Experiments and Implementation of HIRL

Our experiments used the OpenAI Gym implementation of Atari Learning Environment [5, 7], modified to allow interactive blocking of actions by a human. We used open-source implementations [24, 23] of A3C with an LSTM policy [22] and Double DQN [28]. Rewards were clipped when using Double DQN but not for A3C.

For the Blocker (the supervised learner that imitates human blocking) we used a convolutional neural network (CNN). The CNN was trained on the Atari images (rather than the downsampled frames the agent sees) and had no pooling layers. Architectures and hyperparameters for all neural networks are in Appendix.

Our goal is that the Blocker never misses a catastrophe: the false-negative rate should be extremely low. We trained our CNN using the standard loss objective. To achieve a low false-negative rate (at the expense of false positives), we selected a threshold for the CNN’s sigmoid output and blocked any actions that exceeded this threshold. This threshold can be set very low initially (causing many false positives) and then gradually raised until it becomes possible for the agent to learn the task. In our experiments, this simple approach sufficed.

To summarize, our application of HIRL involved the following sequence of steps:

1. **Human Oversight Phase** (duration = 4.5 hours): Fresh RL agent starts playing the game (slowed down to accommodate the human). Human<sup>5</sup> oversees and blocks catastrophic actions.
2. **Blocker training**: The game is paused. The CNN is trained to imitate human blocking decisions. The threshold for sigmoid is chosen to try to ensure Blocker has no false negatives.
3. **Blocker Oversight Phase** (duration = 12-24 hours): Blocker takes over from human and game is run at normal speed.

The main difference between HIRL and regular RL are in (1) and (2). Once the Blocker takes over, the environment runs at full speed for the normal training time for Deep RL agents in Atari.

#### 4.1.1 What are Catastrophes in Atari?

In Atari there are no catastrophic actions: the human researchers running Atari agents don’t care if their agents die millions of times in the process of mastering a game. In our experiments, we stipulate that certain outcomes are catastrophic and require the agent to maximize reward without causing catastrophes. For example, can an agent learn Road Runner without losing a single life? These are the outcomes we stipulate to be catastrophic:

- **Pong**: It’s a catastrophe if the paddle goes close to the bottom of the screen (Fig 2). This is not a bad outcome in regular Pong but provides a toy example for avoiding catastrophes.
- **Space Invaders**: It’s a catastrophe if the agent shoots their own defensive barriers.<sup>6</sup>
- **Road Runner**: It’s a catastrophe if the agent dies on Level 1 (Fig 3).

### 4.2 Summary of Results

The objective is to avoid catastrophes while achieving good performance. This must be achieved with a feasible amount of human oversight. Figure 4 (top) shows that this objective is not trivially satisfied: an agent with no human oversight has ten thousand or more catastrophes in each game.<sup>7</sup>

---

<sup>5</sup> Authors WS and GS took the role of human overseer.

<sup>6</sup> One strategy on Space Invaders is to shoot a narrow line through the barriers and shoot invaders through this line. We did not observe Double DQN use this strategy and so preventing the agent from using it should not harm its performance.

<sup>7</sup> In Pong there is no incentive in the regular game to avoid the Catastrophe Zone. In Space Invaders and Road Runner there is an incentive to avoid the catastrophes but the agents do not become good enough to learn this.

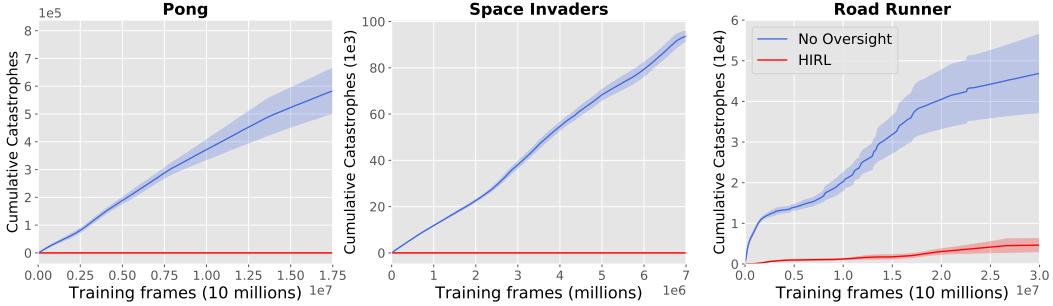
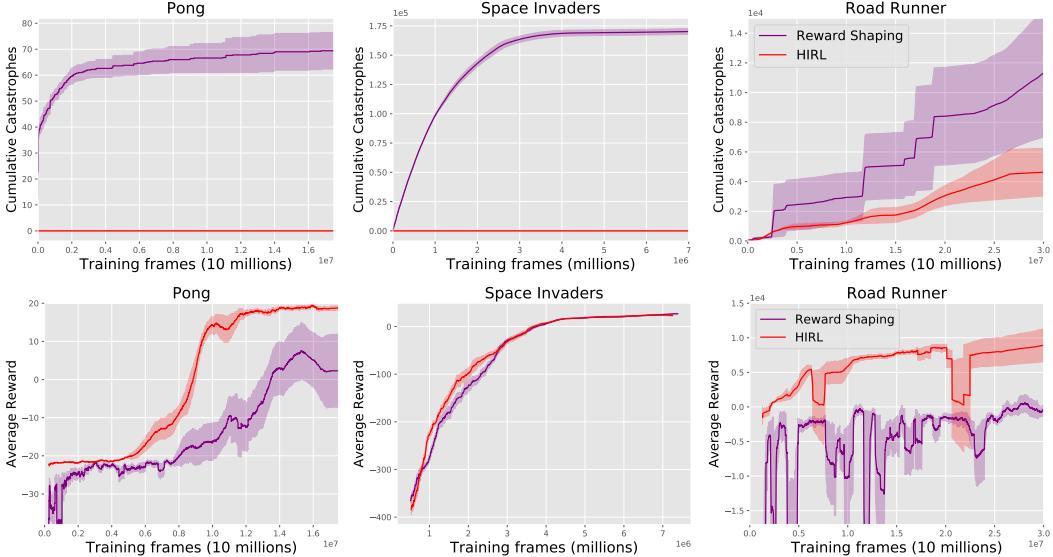


Figure 4: Average Reward and Cumulative Catastrophes over time (showing mean and SE over 5 runs). The **No Oversight** agent (above) gets no human intervention at all; it shows that our objective of preventing catastrophes is not trivial. The **Reward Shaping** baseline (below) is not blocked from catastrophes but gets negative rewards for causing them.



HIRL was a qualified success on the three games we tried. The Human Oversight phase lasted 4.5 hours or less in each game. In Pong and Space Invaders, the agent had zero catastrophes and still was able to achieve impressive performance on the game. In Road Runner the agent had very few (but not zero) catastrophes.

Figure 4 (bottom) compares HIRL to the baseline of human-trained Reward Shaping. In HIRL when the agent tries a catastrophic action it is blocked by the Blocker and it gets a negative reward. In Reward Shaping, the agent gets the same negative reward but is not blocked. The classification of actions as catastrophic is still learned from imitating the human overseer (as in [17, 9]). The Reward-Shaping baseline has no chance of achieving zero catastrophes (see 2.1) but we would hope that the rate of catastrophes converges quickly to zero.

Figure 4 shows that the Reward-Shaping agent does learn to mostly avoid catastrophic actions and has a much lower rate of catastrophes than No-Oversight (Fig 4 top). Yet in Pong and Road Runner its catastrophe rate does not appear to be converging to zero. Section 4.3.3 shows that the persistence of catastrophes in Pong is caused by catastrophic forgetting.

HIRL and Reward-Shaping agents do comparably well at learning the games. By frequently blocking the agent (and replacing their action with another) HIRL essentially changes the physics of the game. Yet this change makes little difference to the agent's learning.

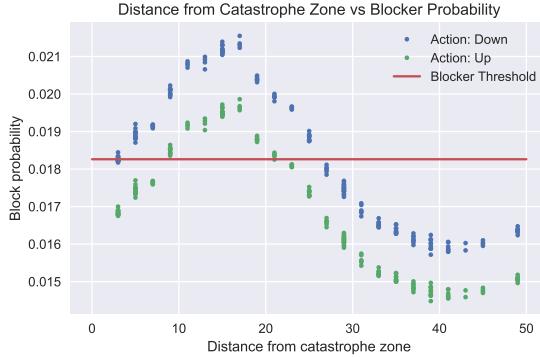


Figure 5: The Blocker blocks the agent when the sigmoid output exceeds the “Blocker Threshold”. The  $x$ -axis is the distance of the agent from the Catastrophe Zone and the  $y$ -axis relates to the Blocker’s probability the action should be blocked.

### 4.3 Pong: Detailed Analysis of the Blocker and of Human Time Cost

HIRL was successful at Pong: an A3C agent mastered Pong while incurring no catastrophes. But what did the Blocker (which managed to Block all catastrophes) actually learn? Would the Blocker work just as well for different RL agents? Why did the Reward Shaping agent (without blocking catastrophic actions) fail and keep trying catastrophic actions?

#### 4.3.1 Interpreting the Blocker CNN

The Blocker has a simple task: block the agent from moving down when doing so would land it in the Catastrophe Zone (Fig 2). Figure 5 shows the Blocker’s sigmoid output for actions “Up” and “Down” across a single episode (1000 frames). The Blocker blocks “Down” actions up to a distance of 25 pixels from the Catastrophe Zone. After distance 25 it stops blocking, as doing so would constrain the agent needlessly.

Note that no catastrophes actually occur in training; so the Blocker’s training set contains no examples where the agent is closer than 20 pixels to the Catastrophes Zone. This explains the odd results for distances 0-20: these are out-of-sample predictions and the Blocker does not always generalize correctly. So while this Blocker does transfer successfully to other agents (see below) it would not be robust to small changes in the dynamics of Pong.

#### 4.3.2 The Blocker transfers perfectly and is robust to adversarial agents

The Blocker was trained on examples from a human overseeing an A3C agent. Figure 4 shows performance for the Blocker on that very same A3C agent. A virtue of HIRL is that his Blocker is modular: while it was trained on data from one agent, it can be applied to another. But would the Blocker be equally reliable for another agent?<sup>8</sup> We applied the Blocker to a variety of RL agents and it always blocked all catastrophes without preventing mastery of Pong. The agents were:

- A3C agents with different architectures/hyper-parameters (see Appendix)
- Double DQN
- A “catastrophe loving” A3C agent: this agent was previously trained on a modified version of Pong where it got positive rewards for entering the Catastrophe Zone

#### 4.3.3 Safety requires constant intervention (due to catastrophic forgetting)

We argued in Section 2.1 that regular RL agents are not “catastrophe-safe”. They only avoid catastrophic actions if they’ve already tried them; so they can’t learn a task without causing a single catastrophe. Figure 4 (bottom) demonstrated a second way in which current Deep RL agents are unsafe: they never stop taking catastrophic actions. The Reward-Shaping agent is initially trained by a human overseer who blocks all catastrophes. After this, the agent receives negative rewards for

<sup>8</sup>Figure 4 does not provide strong evidence that the Blocker will transfer. The graph was generated by computing sigmoid values for every frame within a single episode of the original A3C agent. A different agent will explore different states and the Blocker might generalize poorly to these states.

catastrophes but is not blocked. The agent learns to mostly avoid catastrophes but the catastrophe rate seems to converge to a low but non-zero level.

Table 1: Long-run rate of attempted catastrophes in Pong.

| Policy        | Learning Rate | Catastrophe Rate Per Episode (Std Err) |
|---------------|---------------|----------------------------------------|
| Stochastic    | $10^{-4}$     | 0.012 (0.004)                          |
| Deterministic | $10^{-4}$     | 0.079 (0.017)                          |
| Stochastic    | 0             | 0.003 (0.001)                          |
| Deterministic | 0             | 0 (0)                                  |

Why does the Reward Shaping agent keep taking actions that received a big negative reward? We investigate this by examining how frequently the HIRL agent attempts catastrophic actions.<sup>9</sup> In Pong, the HIRL agent attempts catastrophes at a rate of about  $10^{-5}$  per frame or  $10^{-2}$  per episode. Is this due to A3C’s stochastic policy or due to catastrophic forgetting? We tested this by turning the learning rate down to zero and by making the agent deterministic (taking a max over the policy network’s softmax layer). Table 1 shows the four possibilities. The first row (stochastic and non-zero learning rate) is the regular A3C with HIRL as shown in Pong results for Figure 4. The catastrophe rate only goes down when the learning rate is set to zero. This strongly suggests that catastrophic forgetting is the problem. We also examined the probability of taking a catastrophic action for the regular A3C agent’s policy network: after the agent has mastered Pong, this probability continues to vary between  $10^{-9}$  and 0.3. (This probability is mostly low, much closer to  $10^{-9}$ , but it periodically gets higher until a catastrophe occurs.)

#### 4.3.4 Extrapolating the human time-cost

We noted above that the human blocking for Pong took four hours. We can break this down according to Equation 2:

- $N_{\text{cat}} = 120$ . (number of labeled catastrophes)
- $\rho = 166$ . (ratio of observations to catastrophes observations)
- $t_{\text{human}} = 0.8\text{s}$ . (average time for human to do one label)

The number  $N_{\text{cat}}$  is small because the catastrophe is so simple: the CNN didn’t need much data to learn the concept. The ratio  $\rho$  was relatively small. Once the agent learns to avoid the catastrophe (after 200,000 frames),  $\rho$  goes up to around  $10^4$ . Suppose that in our experiment, we had used an agent pre-trained to avoid catastrophes (instead of a completely fresh agent). With  $\rho = 10^5$ , the total time for human labeling would be  $0.8 \times 10^5 \times 120 = 110$  days: a huge amount of human labor to learn such a simple concept! Ratio  $\rho$  would also be much higher if the Catastrophe Zone (Fig 2) was a small region in a three-dimensional space that the agent is unlikely to randomly move into (e.g. a corner of a dungeon).

### 4.4 Road Runner: Blocking an Adversarial Agent

Road Runner is a side-scrolling game based on the Warner Bros cartoon. The player controls Road Runner, who is constantly pursued by the Coyote (Figure 3). The Road Runner gets points for eating birdseed and for knocking out the Coyote. We stipulate that losing a life is a catastrophe. The main way to lose a life is by touching the Coyote (which is easily avoided in Level 1 by running away).

#### 4.4.1 The Score Exploit

Deep RL agents are known to discover a “Score Exploit” in Road Runner [4]. The agent learns to intentionally kill itself in a way that (paradoxically) earns greater reward. Dying causes the agent

---

<sup>9</sup>The HIRL agent is blocked from actually taking catastrophic actions. By measuring how often it attempts catastrophic actions we learn how many catastrophes it would have caused if blocking was turned off (as in Reward Shaping).

to repeat part of Level 1, where it earns more points than on Level 2 (as Level 2 takes a long time to master). This is a local strategy optimum that a human gamer would never be stuck in. Since the agent *wants* to kill itself to get the Score Exploit, the agent has an incentive to find adversarial examples [13] for our Blocker. That is, to find ways of dying that the Blocker doesn't recognize as catastrophes and so fails to block.

#### 4.4.2 First Version of Blocker Fails Due to Adversarial Agent

Our first attempt to prevent catastrophes in Road Runner was an instructive failure. During the early stages of training the rate of deaths/catastrophes was very low. However, much later in training (after 16 million frames), the death rate rises to almost the same level as the baseline No Oversight agent (see Fig 6). Inspecting videos of the HIRL agent, we found that although the usual Score Exploit was blocked, after 16 million frames the agent found an alternative Score Exploit. The agent moved along the very top of the screen to the top right corner and waited for the Coyote to kill it there. This position at the top of the screen (which is visually distinct from other positions) presumably fooled the Blocker CNN. (In preliminary experiments, the A3C agent found different adversarial examples for an early version of the Blocker. See [videos](#).)

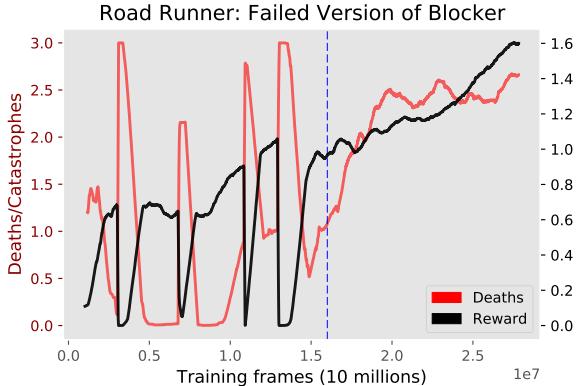


Figure 6: Reward/death-rate over entire training for HIRL agent with failed Blocker. Blue line indicates when agent learned Score Exploit. Before this point the agent's death-rate spikes a few times, indicating additional failures of the Blocker. However these spikes are anti-correlated with reward.

After the Blocker failed, we examined the 20,000 frames used as training data for the Blocker and looked for mistakes in the labels. We spent 20 minutes correcting mistakes and re-trained the Blocker. This reduced the average death rate by a factor of 20: from a rate of 0.002 deaths per frame to 0.0001. The No Oversight baseline has a rate of 0.005.

## 5 Discussion

### 5.1 Summary

Currently, the only way to guarantee the safety of RL systems during training is to have a human watch the system's actions, ready to intervene, or to build an automated overseer that is just as reliable as the human at recognizing catastrophic actions. Deep RL is the leading contender for autonomous, scalable, task-oriented AI. We investigated to what extent human oversight can enable Deep RL systems to master tasks without causing a single catastrophic event.

We applied HIRL to three games (with varying definitions of catastrophe) and trained a Blocker from around four hours of human supervision. In Pong and Space Invaders, the HIRL agent succeeded in learning to play the game with zero catastrophes. In Road Runner, HIRL reduced the rate of catastrophes by a factor of 50 but did not eliminate catastrophes. Our experience with Road Runner illustrates a major challenge for HIRL. If the agent views a catastrophe as optimal, it will try to bring about the catastrophes in ways that fool the Blocker. This requires the Blocker to be robust to distributional shifts and to adversarial attack [2].

## 5.2 Challenges for HIRL

In Pong and Road Runner, catastrophes can occur right at the start of the game. So having a human label catastrophes during the early stages of training is sufficient to train a Blocker to recognize catastrophes. However, it's often the case that catastrophes are only encountered after the agent has made substantial learning progress.

In Space Invaders, the agent starts of the left side of the screen. When a human blocks it from shooting the left barrier, it responds by staying to the left of the left barrier (where it knows it won't get a negative reward penalty). This means for that for many episodes it never goes under the middle or right barriers. To get a training set that includes shooting under those barriers, the human would have to label for a long time. (We estimate 70 hours). We fixed this problem by including episodes where the agent is initially placed at the center or right of the screen. In other environments, there won't be such a quick and easy fix.

In Montezuma's Revenge, for example, our goal may be to prevent the agent walking off a ledge and dying. Yet it might take 100 million frames to reach all the distinct rooms that contain ledges [4]. Overseeing an agent for 100 million frames would take a human at least a year. More complex environments than Montezuma's Revenge are likely to be even more challenging for HIRL.

To guarantee safety in these and more complex environments, we expect that a combination of the following techniques will be required to bring down the time-cost of human supervision (as expressed by Equation 2):

- **Make Blockers (human imitators) more data-efficient:** If a classifier can learn to imitate the human from less data ( $N_{\text{cat}}$  goes down) then the human time-cost decreases.
- **Make RL agents more data-efficient:** Deep RL tends to require millions of observations for successful learning. With more data-efficient RL, the human would not need to wait so long for the agent to observe different catastrophes (as in the case of Space Invaders above).
- **Seek out catastrophes:** Even if the agent is slow to master the whole environment, it could be quick to find the catastrophes. This would mean a higher ratio of catastrophes to safe events (lowering  $\rho$ ) and lower human time-cost. Note that RL agents that are more data-efficient may sometimes *increase* human time-costs. This because they quickly learn to avoid catastrophes and so catastrophes become very rare in the Blocker's training set. This suggests a role for agents who initially explore systematically [25] and aggressively [6] and so encounter many catastrophes early on.<sup>10</sup>
- **Selectively query the human (Active Learning):** The RL agent might spend a long time in states that are “far away” from dangerous regions. Ideally, the human would only provide oversight when the agent gets close to the dangerous regions. For example, on Montezuma’s Revenge, the human might be called in when the agent is about to enter a new room. Techniques from active learning and anomaly detection can be used to detect unfamiliar states [26, 18, 9]. Similar approaches have been pursued in recent work on “safe exploration” [27].
- An algorithm that decides when to ask the human for oversight must have no false negatives: for any novel catastrophe, it must either block the agent directly or pass control back to the human.<sup>11</sup>
- **Explaining why an action is catastrophic:** We could augment the binary “catastrophe”/“safe” labels (that we get automatically based on the human’s decision to intervene or not) with additional information, such as explanations of what exactly caused a catastrophe. This will introduce additional labeling cost, but could make it easier to learn a robust imitator from a small training set.
- **Model-based RL for safe learning:** Model-free RL agents can learn to avoid catastrophes by encountering scaled-down versions of the catastrophic outcome. However, model-based

---

<sup>10</sup>An agent could also be pre-trained in a simulation to seek out catastrophes – as we did for Pong. Or an agent could be placed in special training environments where catastrophes are more common than in the actual environment.

<sup>11</sup>For some environments, the human need not to be ready to take control at all times. When the algorithm suspects an action leads to a novel state, it blocks the action. The action is sent to the human who evaluates (asynchronously) whether the action was safe.

agents could go further and intentionally seek out scaled-down versions of outcomes that might (given their uncertainty over the true model) be catastrophic.

## References

- [1] David Abel, John Salvatier, Andreas Stuhlmüller, and Owain Evans. Agent-agnostic human-in-the-loop reinforcement learning. *CoRR*, abs/1701.04079, 2017. URL <http://arxiv.org/abs/1701.04079>.
- [2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- [3] Intelligence Community Assessment. Background to “assessing russian activities and intentions in recent us elections”: The analytic process and cyber incident attribution. [https://web-beta.archive.org/web/20170421222356/https://www.dni.gov/files/documents/ICA\\_2017\\_01.pdf](https://web-beta.archive.org/web/20170421222356/https://www.dni.gov/files/documents/ICA_2017_01.pdf). Accessed: April-21-2017.
- [4] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.
- [5] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279, 2013.
- [6] Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.
- [7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [8] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.
- [9] Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.
- [10] Kamil Andrzej Ciosek and Shimon Whiteson. Offer: Off-environment reinforcement learning. In *AAAI*, pages 1819–1825, 2017.
- [11] Maricris Francisco. Google waymo performing better than other self-driving cars, says california dmv. <http://www.techtimes.com/articles/195565/20170202/google-waymo-cars-california-dmv.htm>. Accessed: December-02-2017.
- [12] Javier Garcia and Fernando Fernandez. A Comprehensive Survey on Safe Reinforcement Learning. *The Journal of Machine Learning Research*, 16:1437–1480, 2015.
- [13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [14] Xiaoxiao Guo, Satinder Singh, Richard Lewis, and Honglak Lee. Deep learning for reward design to improve monte carlo tree search in atari games. *arXiv preprint arXiv:1604.07095*, 2016.
- [15] Bar Hilleli and Ran El-Yaniv. Deep learning of robotic tasks using strong and weak human supervision. *arXiv preprint arXiv:1612.01086*, 2016.
- [16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.
- [17] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM, 2009.
- [18] David Krueger, Jan Leike, Owain Evans, and John Salvatier. Active reinforcement learning: Observing rewards at a cost. In *NIPS 2016 Workshop*, 2016.

- [19] Zachary C Lipton, Abhishek Kumar, Jianfeng Gao, Lihong Li, and Li Deng. Combating deep reinforcement learning’s sisyphean curse with reinforcement learning. *arXiv preprint arXiv:1611.01211*, 2016.
- [20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015. URL <http://dx.doi.org/10.1038/nature14236>.
- [21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [22] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.
- [23] OpenAI. Openai baselines, . URL <https://github.com/openai/baselines>. Accessed: July-1-2017.
- [24] OpenAI. Openai universe starter agent, . URL <https://github.com/openai/universe-starter-agent>. Accessed: May-1-2017.
- [25] Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *CoRR*, abs/1703.01310, 2017. URL <http://arxiv.org/abs/1703.01310>.
- [26] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.
- [27] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 997–1005, Lille, France, 07–09 Jul 2015. PMLR. URL <http://proceedings.mlr.press/v37/sui15.html>.
- [28] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016.
- [29] Wikipedia. Tay (bot) — wikipedia, the free encyclopedia, 2017. URL [https://en.wikipedia.org/w/index.php?title=Tay\\_\(bot\)](https://en.wikipedia.org/w/index.php?title=Tay_(bot)). Accessed: May-19-2017.

## 6 Appendix

### 6.1 Neural network architectures and hyperparameters

#### 6.1.1 RL agent parameters

A3C agent network architecture:

- Input format: 42x42x1, grayscale, (cropped, downsampled, rgb values averaged)
- 4 convolutional layers with 32 3x3 filters, applied with 2x2 stride
- Last convolutional layer fed into an LSTM with 256 hidden units
- LSTM output fed into linear layers to produce value function estimate and policy logits
- ELU activation
- Learning rate: 0.0001
- Adam Optimizer
- Entropy bonus: 0.01
- Discount factor: 0.99
- Steps between policy gradient updates: 20

Game-dependent reward scaling

- Pong: reward = reward/1.0
- Road Runner: reward = reward/100.0
- Space Invaders: reward squashed to +/- 1

#### 6.1.2 Blocker Architecture

Pong:

- Input format: [105, 80, 3], color (cropped then downsampled)
- 2 convolutional layers, 4 filters size 3x3 with 2x2 stride
- Final convolutional layer concatenated with one-hot embedding of action
- 2 10-unit hidden layers
- Linear layer outputting logits
- Learning rate 0.002
- Adam Optimizer
- Batch size: 400
- No dropout

Space Invaders and Road Runner:

- Input format: [105, 80, 3], color (cropped then downsampled)
- 4 convolutional layers, 16 filters size 3x3 with 2x2 stride
- Final convolutional layer concatenated with one-hot embedding of action
- 2 20-unit hidden layers
- Linear layer outputting logits
- Learning rate 0.002
- Adam Optimizer
- Batch size 400
- Dropout with probability of discarding 0.5
- Examples were reweighted to give positive and negative examples equal weight
- Labels were manually cleaned after collection (by manually reviewing episodes and by looking for individual frames where the blocker disagreed with the given label)