Goal Misgeneralization in Deep Reinforcement Learning

Lauro Langosco [*1] **Jack Koch** [*] **Lee Sharkey** [*2] **Jacob Pfau** [3] **David Krueger** [1]

Abstract

We study *goal misgeneralization*, a type of out-of-distribution robustness failure in reinforcement learning (RL). Goal misgeneralization occurs when an RL agent retains its capabilities out-of-distribution yet pursues the wrong goal. For instance, an agent might continue to competently avoid obstacles, but navigate to the wrong place. In contrast, previous works have typically focused on capability generalization failures, where an agent fails to do anything sensible at test time. We formalize this distinction between capability and goal generalization, provide the first empirical demonstrations of goal misgeneralization, and present a partial characterization of its causes.

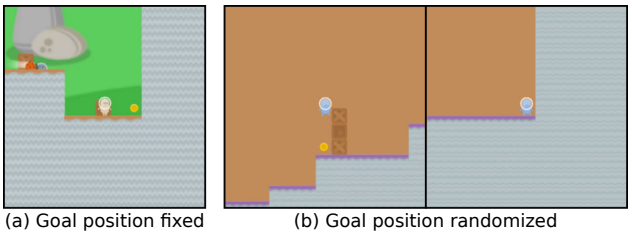

Figure 1. **(a)** At training time, the agent learns to reliably reach the coin which is always located at the end of the level. **(b)** However, when the coin position is randomized at test time, the agent still goes towards the end of the level and often skips the coin. The agent's capability for solving the levels generalizes, but its goal of collecting coins does not.

1. Introduction

Out-of-distribution (OOD) robustness, performing well on test data that is not distributed identically to the training set, is a fundamental problem in machine learning (Arjovsky, 2021). OOD robustness is crucial since in many applications it is not feasible to collect data distributed identically to that which the model will encounter in deployment.

In this work, we focus on a particularly concerning type of OOD robustness that can occur in RL. When an RL agent is deployed out of distribution, it may simply fail to take useful actions. However, there exists an alternative failure mode in which the agent pursues a goal other than the training reward while retaining the capabilities it had on the training distribution. For example, an agent trained to pursue a fixed coin might not recognize the coin when it is positioned elsewhere, and instead competently navigate to the wrong position (Figure 1). We call this kind of failure **goal misgeneralization**[1] and distinguish it from **capability generalization** failures. We provide the first empirical demonstrations of goal misgeneralization to highlight and illustrate this phenomenon.

While it is well-known that the true reward function can be unidentifiable in inverse reinforcement learning (Amin & Singh, 2016), our work shows that a similar problem can also occur in reinforcement learning when features of the environment are correlated and predictive of the reward on the training distribution but not OOD. In this way, goal misgeneralization can also resemble problems that arise in supervised learning when models use unreliable features: both problems are a form of competent misgeneralization that works in-distribution but fails OOD. Disentangling capability and goal generalization failures is difficult in supervised learning; for instance, are adversarial examples bugs or features (Ilyas et al., 2019)? In contrast, studying RL allows us to formally distinguish capabilities and goals, which roughly correspond to understanding the environment dynamics and the reward function, respectively.

Goal misgeneralization might be more dangerous than capability generalization failures, since an agent that capably pursues an incorrect goal can leverage its capabilities to visit arbitrarily bad states (Zhuang & Hadfield-Menell, 2021). In contrast, the only risks from capability generalization failures are those of accidents due to incompetence.

An important implication of goal misgeneralization is that training a model by optimizing an objective R is not enough

[*]Equal contribution [1]University of Cambridge [2]University of Tübingen [3]University of Edinburgh. Correspondence to: Lauro Langosco <langosco.lauro@gmail.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]Here, 'goal' does *not* refer only to goal-states in MDPs, but to goal-directed (optimizing) behavior more broadly.

to guarantee that the model will itself learn to pursue R rather than some proxy for R. This is especially concerning in the context of *AGI safety*: ensuring that advanced AI systems are safe despite being capable enough to escape our control (Bostrom, 2014). Systems that competently pursue a misaligned goal may tend to seek power and deceive their operators for instrumental reasons (Turner et al., 2020; Omohundro, 2008). With highly advanced AI systems, this could lead to human disempowerement: for example, an AI system might prevent its operators from shutting it down (Hadfield-Menell et al., 2017a; Soares et al., 2015). If complex tasks are rife with proxies for their training objectives, it may be very hard to predict what objectives the trained AI systems will have (Hubinger et al., 2019).

Our main contributions are:

- We formalize goal misgeneralization, distinguishing it from capability generalization failures (Section 2), and experimentally validate our definition on a gridworld environment (Section 3.5).
- We experimentally demonstrate that goal misgeneralization can be a significant issue, even when capability generalization failures are rare. Specifically, deep RL agents trained on the Procgen benchmark (Cobbe et al., 2019)—a set of diverse procedurally generated environments specifically designed to induce robust generalization—still fail on our slightly modified environments (Section 3).
- We show that goal misgeneralization may be alleviated by increasing the diversity of the training distribution so that the agent learns to distinguish the reward from proxies (Section 3.1 and 3.2).
- We investigate the causes of goal misgeneralization. In particular, we find that the actor and the critic components of our actor-critic model learn different proxy goals (Section 3.4).

2. Goal Misgeneralization

Goal misgeneralization is a type of OOD robustness failure. OOD robustness is usually studied in the supervised learning setting, where it is defined as achieving good test performance on data sampled from a distribution other than the training distribution. We focus on the reinforcement learning setting (Sutton & Barto, 2018), in which a system is trained to take actions in an environment in order to maximize a given reward. In this setting, the problem is to achieve high reward despite a shift in the distribution of observations or the transition dynamics. OOD robustness problems frequently arise in RL and are an active area of research (Kirk et al., 2021). However, goal misgeneralization in particular has not been the focus of any previous academic work. Studying this class of failures is particularly important from the point of view of machine learning safety (Hendrycks et al., 2021), since agents that pursue imperfect proxies may fail suddenly (Pan et al., 2022; Ibarz et al., 2018) and catastrophically (Zhuang & Hadfield-Menell, 2021) as their capabilities increase. With this in mind, we provide a definition of goal misgeneralization and show how it can be formalized.

2.1. Defining Goal Misgeneralization

A deep RL agent is trained to maximize reward $R\colon S \times A \times S \to \mathbb{R}$, where S and A are the sets of all valid states and actions, respectively. Assume that the agent is deployed under distributional shift; that is, an aspect of the environment (and therefore the distribution of observations) changes at test time. **Goal misgeneralization** occurs if the agent now achieves low reward in the new environment because it continues to act capably yet appears to optimize a different reward $R' \neq R$. We call R the **intended objective** and R' the **behavioral objective** of the agent.

Formally, we follow Orseau et al. (2018) in distinguishing goal-directed policies (*agents*) from 'unoptimized' policies (*devices*). Let $\eta_{\texttt{agt}}(R)$ and $\eta_{\texttt{dev}}(d)$ be priors over objectives R and devices (policies) d respectively. Further let $p_{\texttt{agt}}(\tau \mid R)$ and $p_{\texttt{dev}}(\tau \mid d)$ be the likelihood functions giving the probability of a trajectory τ given a particular objective R or device d. We define two distributions over trajectories, the agent mixture $p_{\texttt{agt}}$ and the device mixture $p_{\texttt{dev}}$:

$$p_{\texttt{agt}}(\tau) = \sum_R p_{\texttt{agt}}(\tau \mid R)\, \eta_{\texttt{agt}}(R), \tag{1}$$

$$p_{\texttt{dev}}(\tau) = \sum_d p_{\texttt{dev}}(\tau \mid d)\, \eta_{\texttt{dev}}(d). \tag{2}$$

The choice of device likelihood $p_{\texttt{dev}}(\tau \mid d)$ is straightforward: we simply choose the distribution over trajectories induced by running the policy d in the environment. For the agent likelihood $p_{\texttt{agt}}(\tau \mid R)$, a popular choice is the maximum entropy model $p(\tau \mid R) \propto \exp(R(\tau))$ (Ziebart et al., 2008). Another possibility is to choose $p(\tau \mid R)$ to be the probability density of the random trajectory obtained by training an RL algorithm to maximize R and collecting rollouts.[2]

Definition 2.1 (Goal misgeneralization)**.** A policy π undergoes *goal misgeneralization* if test reward is low and $p_{\texttt{agt}}(\tau) > p_{\texttt{dev}}(\tau)$ holds on average for the trajectories induced by π in the OOD test environment. In other words, the policy is acting in a goal-directed manner, but not achieving high reward. We can infer a posterior distribution over

[2]This requires a choice of RL algorithm and policy model (e.g. a neural network). Of course, it is usually intractable to compute this choice of $p(\tau \mid R)$ in practice.

behavioral objectives

$$p(R \mid \tau) \propto p(\tau \mid R)p(R).$$

In Section 3.5 we compute these mixtures explicitly and validate Definition 2.1 in a gridworld environment.

2.2. Causes of Goal Misgeneralization

When should we expect models to learn robust goals? We begin by suggesting possible prerequisites for goal misgeneralization:

1. The training environment must be diverse enough to learn sufficiently robust capabilities.
2. There must exist some proxy $R' : S \times A \times S \to \mathbb{R}$ that correlates with the intended objective on the training distribution, but comes apart (i.e. is much less correlated, or anti-correlated) on the OOD test environment.

These conditions are necessary for goal misgeneralization to arise: If (1) is not the case, then RL algorithms tend to memorize simple action sequences that work in the training environment but are not robust under distributional shift (Cobbe et al., 2019). Meanwhile (2) is necessary because by assumption the policy achieves high (training) reward; thus the behavioral objective must be correlated with the intended objective on the training environment. However, (1) and (2) are by no means sufficient since, by themselves, they do not guarantee that the model learns to pursue the proxy reward R' instead of the intended objective.

We note that assumptions (1) and (2) are quite weak: almost every real-world problem requires a diverse training environment (to achieve capability robustness), and proxies are common in complex environments. Thus goal misgeneralization depends mostly on whether the inductive biases of the model and training algorithm prime it to learn a proxy that then diverges from the intended objective on the test set. We expect that learned proxies will:

- be correlated with the intended objective R on the training distribution but not necessarily the test distribution.
- tend to be easier to learn than the intended objective R because a proxy R' may:
  - use features that are simpler or more favored by the inductive biases of the model compared with the intended objective (Valle-Pérez et al., 2019; Geirhos et al., 2020).
  - be denser than the intended objective (Singh et al., 2010).

For example, despite being a product of evolution (which optimizes for genetic fitness), humans tend to be more concerned with proxy goals, such as food or love, than with maximizing the number of their descendants. This illustrates a general phenomenon: given a challenging goal (such as "maximize fitness"), complex environments are rife with proxies and sub-goals (such as "eat rich food") of that goal, many of which are more dense or simpler to optimize than the original goal. This observation has previously been made by Singh et al. (2010), who also draw the analogy with evolution, and note that bounded agents (i.e. with limited experience and/or computation) will often achieve higher expected reward *according to the true reward* when trained to optimize a proxy reward function.

3. Experiments

Having defined goal misgeneralization and outlined when and why we expect it to occur, we now present experiments designed to demonstrate different kinds of goal misgeneralization and distinguish them from capability generalization failures.[3]

In each experiment, we train an agent that performs capably when deployed out-of-distribution, but pursues a behavioral objective different from the objective for which it was trained. This behavior is consistent across ten random seeds.

For each of our experiments we hypothesize a behavioral objective that the policy has learned: navigating to the right-hand end of the level (CoinRun), navigating to the upper right corner (Maze I), navigating to the yellow object (Maze II) and gathering keys (Keys and Chests). None of these is a robust proxy for the intended objective. It is possible that there exist alternate objectives that also explain this behavior: for example, navigating towards a tall, left-facing wall (CoinRun). For our purposes, it is enough to show that a plausible proxy objective exists. Nonetheless, we conduct a series of experiments that confirm the 'move right' hypothesis over the 'move to wall' hypothesis for the CoinRun agent's behavioral objective (see Section 3.4).

We follow a zero-shot protocol in all experiments except Figure 2: the agent does not see the (OOD) testing environment during training. Except in Section 3.5, all environments are adapted from the Procgen environment suite (Cobbe et al., 2019). This suite is built to study sample efficiency and generalization to within-distribution tasks. Agents (feedforward neural networks trained using Proximal Policy Optimization - further details in the Appendix) are tasked with performing well in an arcade-like game from pixel observations. The environments are procedurally generated and thus diverse; to perform well, an agent must learn strategies that work

[3]Our code can be found at https://github.com/JacobPfau/procgenAISC (Environments) and https://github.com/jbkjr/train-procgen-pytorch (Training).

Video examples of goal misgeneralization in all of the following environments can be found at this link.

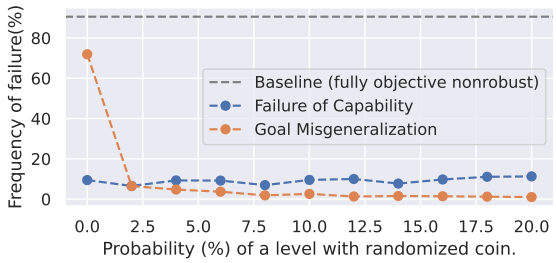

Figure 2. Goal generalization is greatly improved in our CoinRun experiments when just 2% of training levels have randomly placed coins (like the test environment). More randomization helps. Baseline: Since even a policy that entirely ignores the coin may sometimes hit the coin by accident, we compute a base rate for a fully nonrobust policy that treats the coin as invisible.

in a wide range of task settings and difficulties and cannot rely on e.g. memorizing a small number of trajectories to solve a fixed set of levels. This diversity alone is insufficient to prevent goal misgeneralization, however; diversity of a different sort is needed, as we demonstrate in Figure 2.

Different kinds of failure. The experiments illustrate different flavors of goal misgeneralization. *Directional proxies* (CoinRun): the agent learns to move to the right instead of to the true source of reward (the coin). *Location proxies* (CoinRun, Maze I): In Maze I, the agent learns to navigate to the upper right corner instead of to the true source of reward (the cheese). The critic—but not the actor—also learns such a proxy in CoinRun. *Observation ambiguity* (Maze II): The observations contain multiple features that identify the goal state, which come apart in the OOD test distribution. *Instrumental goals* (Keys and Chests): The agent learns an objective (collecting keys) that is only instrumentally useful to acquiring the intended objective (opening chests).

3.1. CoinRun

In the Procgen CoinRun environment, the agent spawns on the left side of the level and has to avoid enemies and obstacles to get to a coin. The coin yields a reward of 10, all other rewards are 0. In our training environments, the coin is always located at the end of the level (the far right, where there is a wall); reaching the coin terminates the episode. To evaluate goal misgeneralization, we create test environments in which the coin is located in a random (accessible) location.

After training, the agent competently navigates to the end of the level in the training environment. At test time, the agent generally ignores the coin completely and proceeds to the end of the level, as shown in Figure 1. This demonstrates that the agent has learned the proxy objective of "move right" rather than "move to the coin". It competently achieves this behavioral objective, which is perfectly correlated with reward on the training distribution and appears to be easier for the agent to learn than the intended objective, but because this objective is not robust, test reward is low.

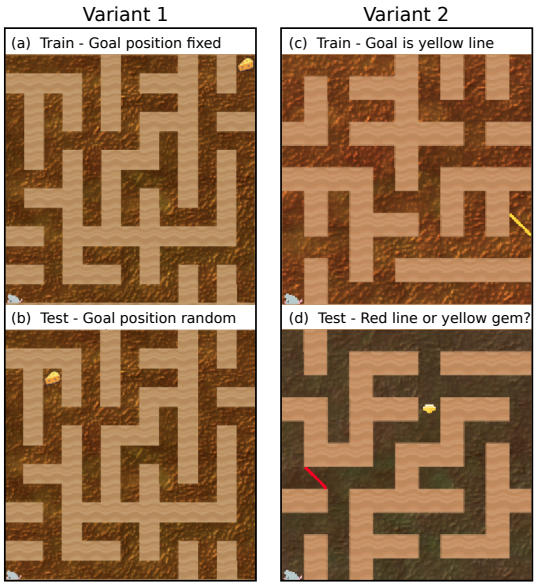

Figure 3. The agent (the mouse) is trained to navigate mazes to reach its goal. **(a & b)** An agent is trained on procedurally-generated mazes with the cheese in a fixed position (top right corner) ignores it and navigates to the top right corner when the cheese is placed randomly. **(c & d)** An agent trained to navigate to a yellow line consistently navigates to a yellow gem when deployed in environments in which there are only red lines and yellow gems. If it were meant to collect lines and not gems, this would be a case of goal misgeneralization.

Training with randomly placed coins. To test how consistent goal misgeneralization is, we train a series of agents on environments which vary in how often the coin is placed randomly. Results can be seen in Figure 2, which shows the frequencies of two different outcomes:

1. **Failure of capability:** the agent dies or gets stuck, thus neither getting the coin nor to the end of the level. This is evaluated on the training environments where the coin is typically at the end of the level.
2. **Goal misgeneralization:** the agent misses the coin and navigates to the end of the level. This is evaluated on the OOD test environments where coin location is randomized.

As expected, as the diversity of the training environment increases, the probability of goal misgeneralization decreases, as the model learns to pursue the coin instead of going to the end of the level. We also include a baseline which measures the rate at which an invisible "coin" would be captured, to

determine how often the coin would be captured by an agent that completely ignores it. We see that even when the coin is always at the end of the level during training, the rate of goal misgeneralization is lower than this baseline.

3.2. Maze

Variant 1. We modify the Procgen Maze environment in order to implement an idea from Hubinger (2020b). In the original environment, a maze is generated using Kruskal's algorithm (Kruskal, 1956), and the agent is trained to navigate towards a piece of cheese located at a random spot in the maze.

We modify the original environment so that the cheese is always in the upper right corner (Figure 3a). As in the CoinRun experiment, when an agent is trained on the environment with a consistent reward location but tested in an environment with a random reward location, the agent ignores the randomly placed objective, instead navigating to the upper right corner of the maze (Figure 3b). Using the terminology of Section 2: the intended objective is to reach the cheese, but the behavioral objective of the learned policy is to navigate to the upper right corner. Somewhat surprisingly, we also find that the agent continues to pursue a proxy objective of "move to the upper right corner" even when this proxy becomes imperfect (see Figure 4).

Variant 2. In the experiments so far, goal misgeneralization arises due to an ambiguity between a visual feature (coin / cheese) and a positional feature (right / top right) which come apart at test time. To illustrate a different kind of distributional shift, we present a simple setting in which there is no *positional* feature that favors one objective over the other; instead, the agent is forced to choose between two ambiguous visual cues.

We train an RL agent on a version of the Procgen Maze environment where the reward is a randomly placed *yellow diagonal line* (Figure 3c). At test time, we deploy it on a modified environment featuring two randomly placed objects: a yellow *gem* and a *red* diagonal line; the agent is forced to choose between consistency in shape or in color (Figure 3d). Except for occasionally getting stuck in a corner, the agent usually pursues the yellow gem, thus generalizing in favor of color rather than shape consistency (89% of the time, excluding occasions where it must pass through the red line to get to the yellow gem, n=102). As in previous examples, training with the correct reward function is not enough to guarantee correct goal generalization here; rather, another approach such as increasing environment diversity or using a different inductive bias may be necessary to specify the intended OOD behavior.

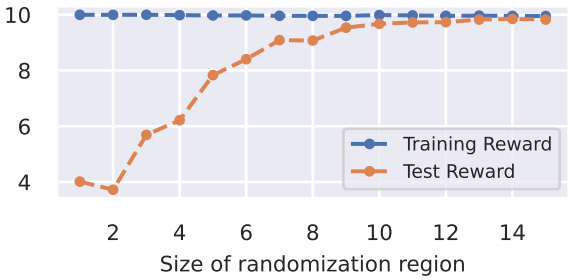

Figure 4. The goal is randomly located within a region of size $1 - 16$ in the upper right corner of the maze. As the region grows, validation performance on the fully randomized environment improves (i.e. correct goal generalization is more likley). However, the agent still uses location as a proxy until the region is quite large.

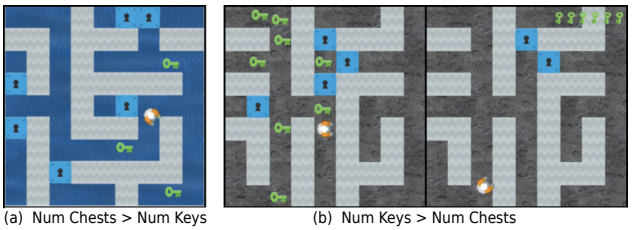

Figure 5. Goal misgeneralization on the "Keys and chests" task. The agent must collect keys in order to open chests and is only rewarded for opening chests. **(a)** The agent is trained on procedurally-generated mazes in which there are twice as many chests as keys. **(b)** At test time, there are instead twice as many keys as chests. The agent routinely prioritizes collecting all the keys before opening any remaining chests despite the fact that doing so offers no benefit to its episode reward (in fact, it would *decrease* its time-discounted return).

3.3. Keys and Chests

So far, our experiments featured environments in which there was a proxy that was perfectly correlated with the intended objective on the training distribution. The Keys and Chests environment, first suggested by Barnett (2019), provides a different type of example. This environment, which we implement by adapting the Heist environment from Procgen, is a maze with two kinds of objects: keys and chests. Whenever the agent comes across a key it is added to a key inventory. When an agent with at least one key in its inventory comes across a chest, the chest is opened and a key is deleted from the inventory. The agent is rewarded for every chest it opens.

As in previous experiments, we induce goal misgeneralization by subjecting the agent to different training and test environment distributions: In the training environment, there are twice as many chests as keys, while in the test en-

vironment there are twice as many keys as chests. The basic task facing the agent is the same (the reward is only given upon opening a chest), but the circumstances are different.

We observe that an agent trained on the "many chests" distribution goes out of its way to collect all the keys before opening the last chest on the "many keys" distribution (Figure 5 and figure 10, Appendix), even though only half of them are even instrumentally useful for the intended objective; occasionally, it even gets distracted by the keys in the inventory (which are displayed in the top right corner) and spends the rest of the episode trying to collect them instead of opening the remaining chest(s).

Applying the intentional stance, we describe the agent as having learned a simple behavioral objective: collect as many keys as possible, while sometimes visiting chests. This strategy leads to high reward in an environment where chests are plentiful and the agent can thus focus on looking for keys. One reason that the agent may have learned this proxy is that the proxy is less sparse than the intended objective while nevertheless being correlated with it on the training distribution. However, this proxy objective fails under distributional shift when keys are plentiful and chests are no longer easily available.

3.4. Critic Generalization vs. Actor-Critic Generalization

All of the experiments above use PPO (Schulman et al., 2017), an actor-critic method (Sutton et al., 1998). In these methods, the policy ("actor") learns to optimize an approximate value function provided by the "critic". So far, we've demonstrated goal misgeneralization – the actor behaves in a goal-directed manner but doesn't achieve high test reward. Why does the actor behave this way? Is it robustly optimizing a non-robust critic? Or is it non-robustly optimizing a robust critic? Or is neither component robust OOD? In this section we show that the actor and the critic *both* fail to generalize OOD; in particular they *fail in different ways*. We conclude that the actor and the critic have different inductive biases that lead them to fail in different ways.

Critic Misgeneralization. In order to determine how much the critic values the coin (the reward) vs. reaching the end of the level, we compare the value it assigns to states where these factors are varied, see Figure 6. We find that the value is much higher at the end of the level than the beginning or middle. The presence of the coin has an insignificant effect. This demonstrates a robustness failure in the critic. To help identify the features in observations at the end of the level that cause higher value, we also generated attribution maps by taking the gradient of the value function output with respect to the observation, following Simonyan et al. (2013). The end-wall is highlighted at least as much as the coin (Figure 9, Appendix).

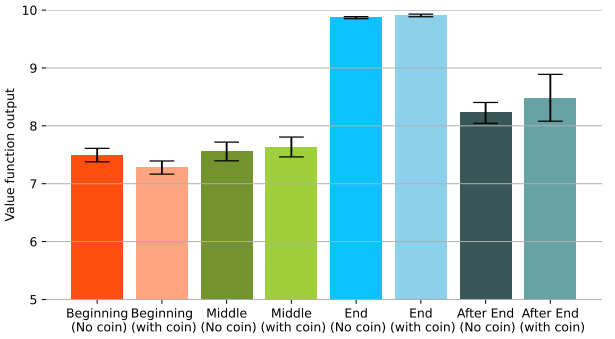

Figure 6. The average value function output for images ($n = 950$) at different stages of CoinRun levels, with and without a coin visible. Error bars are bootstrapped 95% confidence intervals. The coin has an insignificant effect at all stages of a level.

Actor-Critic Misgeneralization. In Section 3.1 we established that the actor (the policy) misgeneralizes. Here we show that the behavior of the actor and the output of the critic are inconsistent: the actor navigates as far right as possible even when this involves moving past a wall, whereas the critic assigns highest value to states in which the agent is touching a wall before having moved past it.

We deploy the agent in an environment with a permeable end wall. If the actor generalized correctly with respect to the critic, it should stay at the wall, or return to it upon passing through it. Instead, whenever the agent reaches the end wall it continues moving right and passes through the wall 100% of the time (n=114) (see Figure 7). This indicates that the policy pursues a "move right" proxy objective, rather than the "move to the wall" proxy objective of the critic, or the intended "move to the coin" objective. In other words, the actor learns a "non-robust proxy of a non-robust proxy". Its failure to match the critic's proxy objective is another source of and example of goal misgeneralization.

3.5. Measuring Agency

We validate the formal definition of goal misgeneralization from Section 2 by explicitly computing the agent and device mixtures in a gridworld environment based on work by Orseau et al. (2018), shown in Figure 8. In this environment there are 4 possible actions (move up, down, left, right). The state consists of two sets of (row, column) coordinates: the position of the agent and of the goal. Possible goal states include every accessible square in the gridworld; formally, our set of possible objectives is

$$\mathcal{R} = \{R_s \mid s \in S\},$$

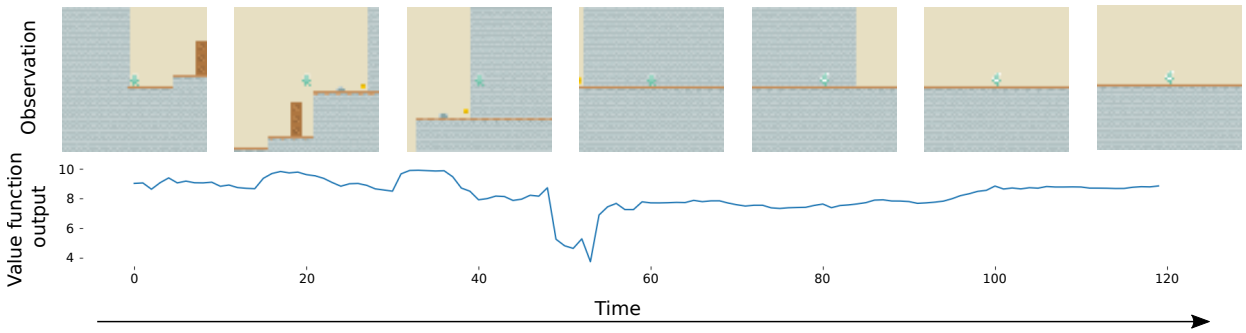

Figure 7. Observations and critic's value estimate for a typical OOD episode with permeable end wall. The agent continues to move to the right, through the wall. This happens even though the critic assigns the highest value around timestep 35, when the agent is just left of the wall (where the coin is typically located during training). This phenomenon occurs 100% of the time that the agent reaches the permeable wall (n=114). This demonstrates that the actor's behavioural proxy objective differs from the critic's proxy objective. Such differences could cause objective failures even in situations where a critic has learned the true value function.

where S is the set of accessible squares in the gridworld and $R_s(s') = 1$ if $s = s'$ and 0 otherwise.

We generate trajectories of an agent attempting to reach a goal cell g. We distinguish four types of trajectories (Figure 8); depending on the type, the goal position is either random or fixed. We distinguish capability from goal generalization failure by comparing the mixture probabilities $p_{\texttt{agt}}(\tau)$ and $p_{\texttt{dev}}(\tau)$ as described in Section 2. A detailed description of the trajectory types and the computation of the mixture probabilities is available in Appendix C.1.

Consider a policy that successfully solves a maze in which the locations of the start state and goal state are fixed (Figure 8, top left). There are three ways this policy might generalize OOD, illustrated in Figure 8.

1. A goal misgeneralizing policy might reliably navigate to the location where the goal was during training, ignoring its actual location (Figure 8, top right).

2. A policy that fails at capability generalization might memorize the trajectory from start to goal, and behave randomly on other states (Figure 8, bottom left).

3. A robust policy would reliably solve the task for any location of goal and start state (Figure 8, bottom right).

As shown in Table 1, the agents & devices formalism successfully distinguishes goal misgeneralization from capability generalization failures: The robust policy as well as the misgeneralizing policy are clearly recognized as goal-directed agents, whereas the policy that fails at capability generalization is correctly classified as non-agent.

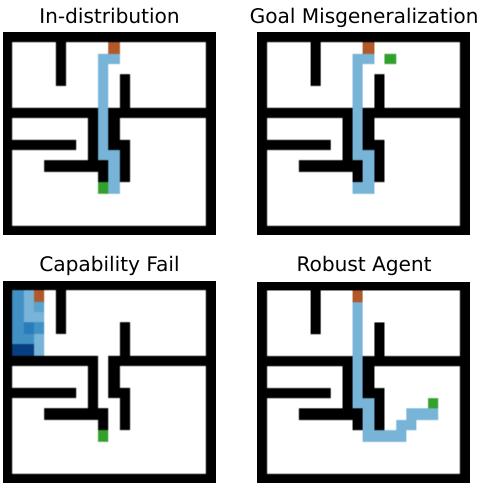

Figure 8. ■ Start. ■ Goal. *In-distribution:* the policy reaches its (fixed) goal. *Goal misgeneralization*: the policy navigates to the wrong position when the goal is moved. *Capability Generalization Failure:* when start position is moved, the policy gets stuck. *Robust:* the policy always reaches the goal for all start / goal positions.

Type	$-\log p_{\texttt{agt}}(\tau)$	$-\log p_{\texttt{dev}}(\tau)$	$p(\texttt{agt} \mid \tau)$
IID	5.7	20.8	0.9999
G. Misg.	14.1	30	0.9999
Cap. Fail	72	69	0.0674
Robust	10.5	30.5	0.9999

Table 1. As expected, all trajectories from Figure 8 are classified as agents, except the capability generalization failure (Cap. Fail).

4. Related Work

Out-of-Distribution Robustness. Goal misgeneralization is a form of out-of-distribution (OOD) robustness failure. OOD robustness covers situations where machine learning models are required to generalize to a novel distribution at test time. Causes for such a train-test mismatch include: i) the training data does not characterize the true distribution (Torralba & Efros, 2011), ii) the distribution shifts over time (Quiñonero-Candela et al., 2009), and iii) the test data are adversarially perturbed (Szegedy et al., 2014; Goodfellow et al., 2015). OOD robustness is a well established limitation of existing deep learning approaches, which can be very sensitive to changes in distribution (Recht et al., 2019), and may base their predictions on shortcuts (Geirhos et al., 2020) or spurious correlations (Beery et al., 2018; Arjovsky et al., 2020). Such lack of robustness may be due to underspecification (D'Amour et al., 2020): since there are many patterns a deep network can learn which explain the training distribution equally well, practitioners may need to provide additional information to disambiguate between these possible solutions. Underspecification of the mapping from state to reward is present in our CoinRun and Maze environments, making goal misgeneralization *unavoidable* if the inductive biases of the deep RL algorithms deployed don't match the intended behavior. The existing work on OOD robustness is largely complementary to our work here on goal misgeneralization. For example, learning invariant predictors (Arjovsky et al., 2020; Krueger et al., 2021) across diverse training environments might help a model learn the true cause of the reward it receives and increase goal misgeneralization. We also add to existing work on OOD robustness by highlighting that when a model fails to generalize OOD, it may do so in two different ways that have notably different consequences: it might generalize completely incapably, or it might generalize capably but pursue an incorrect objective. This distinction is important because pursuing an incorrect objective can lead to different—and potentially more severe—consequences (Zhuang & Hadfield-Menell, 2021). OOD robustness is especially important in online RL because updating the policy leads to a shift in the training distribution.

Generalization in RL. We define and study goal misgeneralization in the context of reinforcement learning. Historically, generalization in RL received little attention, but many recent works address various forms of RL generalization, including OOD generalization. Notable directions of research include sim-to-real (Peng et al., 2018), robust RL (Morimoto & Doya, 2005), and offline RL (Levine et al., 2020); see Kirk et al. (2021) for a review. Solving classic deep RL environments such as ATARI (Bellemare et al., 2013) may already require generalizing across states, but Cobbe et al. (2019) note that overfitting to a particular environment is commonly observed, and propose diverse sets of environments to promote generalization. While Cobbe et al. (2019) use the same distribution of environments during training and test time, we modify their environments to create OOD test environments.

Goal Misgeneralization / Objective Robustness. An earlier public version of this work used the term 'objective robustness failure' in the place of 'goal misgeneralization'. Previous work on OOD robustness has largely failed to distinguish between goal misgeneralization and capability generalization failures. Hubinger et al. (2019) and Mikulik (2019) are perhaps the first to make such a distinction explicitly, and the term *objective robustness* is used by Hubinger (2020a) to refer to the former failure mode. These works also argue that goal misgeneralization may be catastrophic, motivating our focus on this type of failure. Previously, Leike et al. (2018) used the term *reward-result gap* to refer to the difference between what a model was optimiz*ed* for and what it appears to be optimiz*ing* (i.e. what we call the behavioral objective). But they fail to note that some agents may not appear to be optimizing *anything*, and might be better understood as devices (Orseau et al., 2018). We add to these works by formalizing the distinction between capability generalization failure and goal misgeneralization, and providing the first empirical demonstrations of goal misgeneralization in deep RL systems.

Mesa-Optimization. Public non-academic discussions of concerns related to goal misgeneralization, and the analogy with evolution described in Section 2.2, go back (at least) to 2016 (Yudkowsky, 2016; Christiano, 2016).[4] These early discussions, as well as Hubinger et al. (2019), focused on goal misgeneralization caused by **mesa-optimization**, a phenomenon where a model learns an optimization process (even if not explicitly trained to do so). Mesa-optimization could lead to goal misgeneralization if the learned "inner objective" optimized differs from the "outer objective" specified by the designer, but this need not be the case. Furthermore, goal misgeneralization can occur without mesa-optimization. Thus these are in fact two distinct behaviors, and our work does not demonstrate or address mesa-optimization. Mesa-optimization could be a concern independent of goal misgeneralization if the mesa-optimizer pursues undesirable *means* of optimizing the correct objective (Krueger et al., 2020), e.g. we might not want a prediction system to make self-fulfilling prophecies (Armstrong, 2017). Furthermore, while we've defined goal misgeneralization as a form of OOD robustness failure, mesa-optimization could (hypothetically) lead to undesirable behavior such as deception or power-seeking (Turner et al., 2020) on-distribution.

[4] Terms used in these discussions include "subsystem reasoning" (Taylor, 2017), "optimization daemons", "inner optimizers", and "inner alignment" (Rice & many authors, 2018).

Unidentifiability in Inverse Reinforcement Learning. goal misgeneralization tends to arise when there are multiple possible reward functions that are indistinguishable from the true reward and produce similar behavior on the training set, but not OOD. This type of unidentifiability is analogous to the one encountered in inverse reinforcement learning (IRL). Amin & Singh (2016) separate the causes for this unidentifiability in IRL into two classes. The first, **representational unidentifiability**, arises because some transformations of reward functions, e.g. rescaling, preserve the *relative* returns of different policies. The second, **experimental unidentifiability**, occurs when π's observed behavior is optimal under two (or more) reward functions which are not functionally equivalent—i.e. there exist situations where they would entail different optimal behavior. Goal misgeneralization can arise from experimental unidentifiability when an agent only encounters situations that distinguish its behavioral objective from the intended objective function at test time.

Reward Misspecification. Reward specification is the problem of specifying a reward that captures the behavior we want (Amodei et al., 2016; Clark & Amodei, 2016). Goal misgeneralization is a distinct problem: it may still fail even if the reward function is perfectly specified.[5] Reward misspecification can produce similar failures to objective non-robustness, however, when the designer specifies a proxy objective that yields good training performance, but fails OOD (Hadfield-Menell et al., 2017b).

5. Discussion

We have formally defined the problem of goal misgeneralization in RL, and provided the first explicit examples of goal misgeneralization in deep RL systems. We argue that goal misgeneralization is a natural category since, much like adversarial robustness failures, goal misgeneralization has distinct causes and poses distinct problems.

Our definition of goal misgeneralization via the agent and device mixtures is practically limited: it is generally hard to define a useful prior over objectives, and the computation quickly becomes intractable for large and complex environments. Conceptually, the division into agents and devices is somewhat restrictive; for example, multi-agent systems do not naturally fit into the framework.

Better understanding agency and optimization remains an important avenue for future work. There is a number of interesting questions in this direction, such as formalizing how some part of the world can optimize some other part of the world and thus be an agent *embedded* in its environment (Demski & Garrabrant, 2019), and understanding when deep learning systems are likely to behave like agents optimizing proxy objectives.

Future empirical work may study the factors that influence goal misgeneralization. For instance, what kinds of proxy objectives are agents most likely to learn? This may help us understand what kinds of environment diversity are most useful for learning robust goals.

[5]Failures due to objective misspecification occur when the model behaves in an unintended way that nevertheless scores highly on the reward function. In contrast, in goal misgeneralization, models score *poorly* on the training reward because they are pursuing an different objective.

6. Contributions

JK and LL independently proposed the idea of demonstrating goa misgeneralization. LS suggested to use Procgen for experiments and conceived of the CoinRun demonstration; JK, LL, LS, and JP set up and trained the agent on CoinRun. LL and JP modified the Procgen environments, and LL ran the sweeps in CoinRun and Maze. LS with assistance from LL conceived and ran the experiments in section 3.4. LS ran the attribution map experiments. DK became involved after the original arXiv preprint; he proposed defining goal misgeneralization via agents and devices (Orseau et al., 2018), proposed the experiment in Figure 4, and made major contributions to the writing and presentation. Laurent Orseau ran the experiment in Figure 8, following a specification designed by LL and DK. His contributions are worthy of authorship, but he joined and contributed after the ICML deadline for author inclusion. The manuscript was written by DK, JK, LL, and LS.

Acknowledgements

We would especially like to thank Laurent Orseau, whose contributions are worthy of authorship (he joined and contributed after the ICML deadline for authorship inclusion).

Special thanks to Rohin Shah and Evan Hubinger for their guidance and feedback throughout the course of this project. Thanks also to Max Chiswick for assistance adapting the code for training the agents, Adam Gleave, Robert Kirk, and Dmitrii Krasheninnikov for helpful feedback on drafts of this paper, and the organizers of the AI Safety Camp for bringing the authors of this paper together: Remmelt Ellen, Nicholas Goldowsky-Dill, Rebecca Baron, Max Chiswick, and Richard Möhn.

This work was supported by funding from the AI Safety Camp and Open Philanthropy. Lee Sharkey was supported by the Centre For Effective Altruism Long Term Future Fund and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC2064/1–390727645

References

Amin, K. and Singh, S. P. Towards resolving unidentifiability in inverse reinforcement learning. *CoRR*, abs/1601.06569, 2016. URL http://arxiv.org/abs/1601.06569. 1, 9

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. URL http://arxiv.org/abs/1606.06565. 9

Arjovsky, M. *Out of Distribution Generalization in Machine Learning*. PhD thesis, New York University, 2021. 1

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization, 2020. 8

Armstrong, S. Good and safe uses of AI oracles. *CoRR*, abs/1711.05541, 2017. URL http://arxiv.org/abs/1711.05541. 8

Barnett, M. A simple environment for showing mesa misalignment. AI Alignment Forum, 2019. URL https://www.alignmentforum.org/posts/AFdRGfYDWQqmkdhFq. 5

Beery, S., Van Horn, G., and Perona, P. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 456–473, 2018. 8

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, Jun 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://dx.doi.org/10.1613/jair.3912. 8

Bostrom, N. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, Inc., 2014. ISBN 0199678111. 2

Christiano, P. F. What does the universal prior actually look like?, Nov 2016. URL https://tinyurl.com/uniprior. 8

Clark, J. and Amodei, D. Faulty reward functions in the wild. OpenAI Blog, 2016. URL https://openai.com/blog/faulty-reward-functions/. 9

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019. 2, 3, 8

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., Hormozdiari, F., Houlsby, N., Hou, S., Jerfel, G., Karthikesalingam, A., Lucic, M., Ma, Y., McLean, C., Mincu, D., Mitani, A., Montanari, A., Nado, Z., Natarajan, V., Nielson, C., Osborne, T. F., Raman, R., Ramasamy, K., Sayres, R., Schrouff, J., Seneviratne, M., Sequeira, S., Suresh, H., Veitch, V., Vladymyrov, M., Wang, X., Webster, K., Yadlowsky, S., Yun, T., Zhai, X., and Sculley, D. Underspecification presents challenges for credibility in modern machine learning, 2020. 8

Demski, A. and Garrabrant, S. Embedded agency. *arXiv preprint arXiv:1902.09469*, 2019. 9

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018. 13

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, November 2020. doi: 10.1038/s42256-020-00257-z. URL https://doi.org/10.1038/s42256-020-00257-z. 3, 8

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2015. 8

Hadfield-Menell, D., Dragan, A., Abbeel, P., and Russell, S. The off-switch game. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017a. 2

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S., and Dragan, A. Inverse reward design, 2017b. 9

Hendrycks, D., Carlini, N., Schulman, J., and Steinhardt, J. Unsolved problems in ml safety, 2021. 2

Hilton, J., Cammarata, N., Carter, S., Goh, G., and Olah, C. Understanding rl vision. *Distill*, 2020. doi: 10.23915/distill.00029. https://distill.pub/2020/understanding-rl-vision. 15

Hubinger, E. Clarifying inner alignment terminology. AI Alignment Forum, 2020a. URL https://www.alignmentforum.org/posts/SzecSPYxqRa5GCaSF. 8

Hubinger, E. Towards an empirical investigation of inner alignment. AI Alignment Forum, 2020b. URL https://www.alignmentforum.org/posts/2GycxikGnepJbxfHT. 5

Hubinger, E., van Merwijk, C., Mikulik, V., Skalse, J., and Garrabrant, S. Risks from learned optimization in advanced machine learning systems. *arXiv preprint arXiv:1906.01820*, 2019. 2, 8

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in atari, 2018. 2

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features, 2019. 1

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. 13

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021. 2, 8

Krueger, D., Maharaj, T., and Leike, J. Hidden incentives for auto-induced distributional shift. *CoRR*, abs/2009.09153, 2020. URL https://arxiv.org/abs/2009.09153. 8

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Priol, R. L., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex), 2021. 8

Kruskal, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. ISSN 0002-9939, 1088-6826. doi: 10.1090/S0002-9939-1956-0078686-7. URL https://www.ams.org/proc/1956-007-01/S0002-9939-1956-0078686-7/. 5

Lee, H. Training procgen environment with pytorch. https://github.com/joonleesky/train-procgen-pytorch, 2020. 13

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. Scalable agent alignment via reward modeling: a research direction. *CoRR*, abs/1811.07871, 2018. URL http://arxiv.org/abs/1811.07871. 8

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. 8

Mikulik, V. 2d robustness. AI Alignment Forum, 2019. URL https://www.alignmentforum.org/posts/2mhFMgtAjFJesaSYR. 8

Morimoto, J. and Doya, K. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005. 8

Omohundro, S. The basic ai drives. volume 171, pp. 483–492, 01 2008. 2

Orseau, L., McGill, S. M., and Legg, S. Agents and devices: A relative definition of agency. *CoRR*, abs/1805.12387, 2018. URL http://arxiv.org/abs/1805.12387. 2, 6, 8, 9, 13

Pan, A., Bhatia, K., and Steinhardt, J. The effects of reward misspecification: Mapping and mitigating misaligned models, 2022. 2

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019. 13

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. doi: 10.1109/icra.2018.8460528. URL http://dx.doi.org/10.1109/ICRA.2018.8460528. 8

Quiñonero-Candela, J., Sugiyama, M., Lawrence, N. D., and Schwaighofer, A. *Dataset shift in machine learning*. Mit Press, 2009. 8

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet?, 2019. 8

Rice, I. and many authors. Mesa-optimization, Feb 2018. URL https://www.lesswrong.com/tag/mesa-optimization. 8

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 6, 13

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 6, 15

Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010. 3

Soares, N., Fallenstein, B., Armstrong, S., and Yudkowsky, E. Corrigibility. In *AAAI Workshop: AI and Ethics*, 2015. 2

Sutton, R., Barto, R., Barto, A., Barto, C., Bach, F., and Press, M. *Reinforcement Learning: An Introduction*. A Bradford book. MIT Press, 1998. ISBN 9780262193986. URL https://books.google.de/books?id=CAFR6IBF4xYC. 6

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018. 2

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199. 8

Taylor, J. On motivations for miri's highly reliable agent design research, Jan 2017. URL https://tinyurl.com/mirimotiv. 8

Torralba, A. and Efros, A. A. Unbiased look at dataset bias. In *CVPR 2011*, pp. 1521–1528. IEEE, 2011. 8

Turner, A. M., Smith, L., Shah, R., Critch, A., and Tadepalli, P. Optimal Policies Tend to Seek Power. *arXiv:1912.01683 [cs]*, December 2020. URL http://arxiv.org/abs/1912.01683. arXiv: 1912.01683. 2, 8

Valle-Pérez, G., Camargo, C. Q., and Louis, A. A. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2019. 3

Yudkowsky, E. Optimization daemons, Mar 2016. URL https://arbital.com/p/daemons/. 8

Zhuang, S. and Hadfield-Menell, D. Consequences of misaligned ai. *arXiv preprint arXiv:2102.03896*, 2021. 1, 2, 8

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008. 2

A. Implementation details

For all environments, we use an Actor-Critic architecture using Proximal Policy Optimization (PPO) (Schulman et al., 2017). The architecture is based on the architecture used in (Espeholt et al., 2018) but omits the recurrent components of the original network. Both the actor (policy function) and critic (value function) are implemented by feedforward neural networks on top of a shared residual convolutional network. All models are implemented in PyTorch (Paszke et al., 2019) and our implementations are based on a codebase by Lee (2020). Unless otherwise stated, models are trained on 100k procedurally generated levels for 200M timesteps. We use the Adam optimizer (Kingma & Ba, 2015) in all experiments. Each training run required approximately 30 GPU hours of compute on a V100.

B. Unidentifiability

The central cause of

B.1. Unidentifiability in the limit of very advanced systems

We have identified unidentifiability of the training reward as a central problem and cause of objective non-robustness. In the limit of general intelligence and strategic awareness, all reward functions are unidentifiable. Consider

C. Experiment Details

C.1. Measuring Agency

C.1.1. Generating Trajectories

Fix positions s and g in a 20x20 gridworld. Then we generate trajectories $\tau_1, \ldots, \tau_n$ as follows. For every trajectory τ_i, we sample random positions $s^{(i)}_{\text{rand}}$ and $g^{(i)}_{\text{rand}}$. Note how $s^{(i)}_{\text{rand}}$ and $g^{(i)}_{\text{rand}}$ are take on new values for every trajectory, while s and g are fixed. For every trajectory, we also identify one of the gridworld states as the *intended goal* $g^{(i)}_{\text{true}}$. We then generate four types of trajectories $\tau^1_i, \tau^2_i, \tau^3_i, \tau^4_i$:

1. Set $g^{(i)}_{\text{true}} = g$. Pick the trajectory that takes the shortest path from $s^{(i)}_{\text{rand}}$ to $g^{(i)}_{\text{true}}$ ('In-distribution').
2. Set $g^{(i)}_{\text{true}} = g^{(i)}_{\text{rand}}$. Pick the trajectory that takes the shortest path from $s^{(i)}_{\text{rand}}$ to g ('OR Failure'),
3. Set $g^{(i)}_{\text{true}} = g$. Pick the trajectory that starts at $s^{(i)}_{\text{rand}}$ and moves in a uniformly random direction every step, for 50 timesteps. If the trajectory ever crosses the shortest path from s to g, then it follows that path to g ('CR Failure').
4. Set $g^{(i)}_{\text{true}} = g^{(i)}_{\text{rand}}$. Pick the trajectory that takes the shortest path from $s^{(i)}_{\text{rand}}$ to $g^{(i)}_{\text{true}}$ ('Robust agent').

We are left with $4n$ trajectories $(\tau^\lambda_i)_{i \leq n, \lambda \leq 4}$. Note:

1. The trajectories τ^1_i are generated from a policy that can reach the fixed goal state $g^{(i)}_{\text{true}} = g$ from any place on the grid.
2. The trajectories τ^2_i are generated from the same policy, deployed in an environment where the goal state $t^{(i)}_{\text{true}}$ is changed. The policy still navigates to the fixed position g, but this is no longer the correct goal; this behavior is designed to match the behavior we saw in the policies we trained for the Maze experiments in Section 3.2.
3. The trajectories τ^3_i are designed to imitate the capability robustness failure of a policy which navigates from a fixed start state to a fixed end state. When initialized to a random start location, the policy takes random actions since it only knows to navigate along a fixed path.
4. The trajectories τ^4_i are generated from a policy that robustly takes the shortest path to $g^{(i)}_{\text{true}}$ from any position in the gridworld even when the goal state is randomized.

C.1.2. Calculating Mixture Probabilities

We follow the method in Orseau et al. (2018). The observations available to agent policies include the goal state and the position of the agent.

Agent prior. We specify the set of possible goal states to consist of all n^2 locations in the gridworld. We do not use the switching prior.

Agent mixture. We specify the set of goals to consist of all accessible squares in the gridworld, plus the (variable) goal $g_{\text{true}}^{(i)}$. Note that g_{true} can be random in the cases where we set $g_{\text{true}}^{(i)} = g_{\text{rand}}^{(i)}$), and thus vary from trajectory to trajectory. Formally, our set of objectives is

$$\mathcal{R} = \{R_s \mid s \in S \cup \{g_{\text{true}}\}\},$$

where S is the set of accessible squares in the gridworld and $R_s(s') = 1$ if $s = s'$ and 0 otherwise. We then take a uniform prior $\eta_{\texttt{agt}}(R) = 1/|\mathcal{R}|$ over this set. Given an objective R, define the probability $p_\varepsilon(\tau \mid R)$ of a trajectory as induced by an ε-greedy policy. Here, the observations of the policy consist of the (row, column) position of the agent. We then integrate over ε:

$$p_{\texttt{agt}}(\tau \mid R) = \int_0^1 p_\varepsilon(\tau \mid R)\,\mathrm{d}\varepsilon.$$

Device mixture. Recall that a device is just a stochastic, tabular policy that takes in an observation and outputs an action. The observation consists of the type of cell (empty, wall, start, goal) that the device is facing, in the direction of its last action. Our device prior $\eta_{\texttt{dev}}$ is uniform over the space of policies. Set $p_\varepsilon(\tau \mid d)$ to be the probability of a trajectory generated by acting in an ε-deterministic way with respect to d, that is take the action determined by d with probability $1 - \varepsilon$ and a random action otherwise. Just as previously we integrate over ε in $[0, 1]$ to compute the final likelihood $p_{\texttt{dev}}(\tau \mid d)$.

D. Hyperparameters

Table 2. Hyperparameters

ENV. DISTRIBUTION MODE	HARD
γ	.999
λ	.95
LEARNING RATE	0.0005
# TIMESTEPS PER ROLLOUT	256
EPOCHS PER ROLLOUT	3
# MINIBATCHES PER EPOCH	8
MINIBATCH SIZE	2048
ENTROPY BONUS (k_H)	.01
PPO CLIP RANGE	.2
REWARD NORMALIZATION?	YES
LEARNING RATE	5×10^{-4}
# WORKERS	4
# ENVIRONMENTS PER WORKER	64
TOTAL TIMESTEPS	200M
ARCHITECTURE	Impala
LSTM?	No
FRAME STACK?	No

E. Value attribution maps and other figures

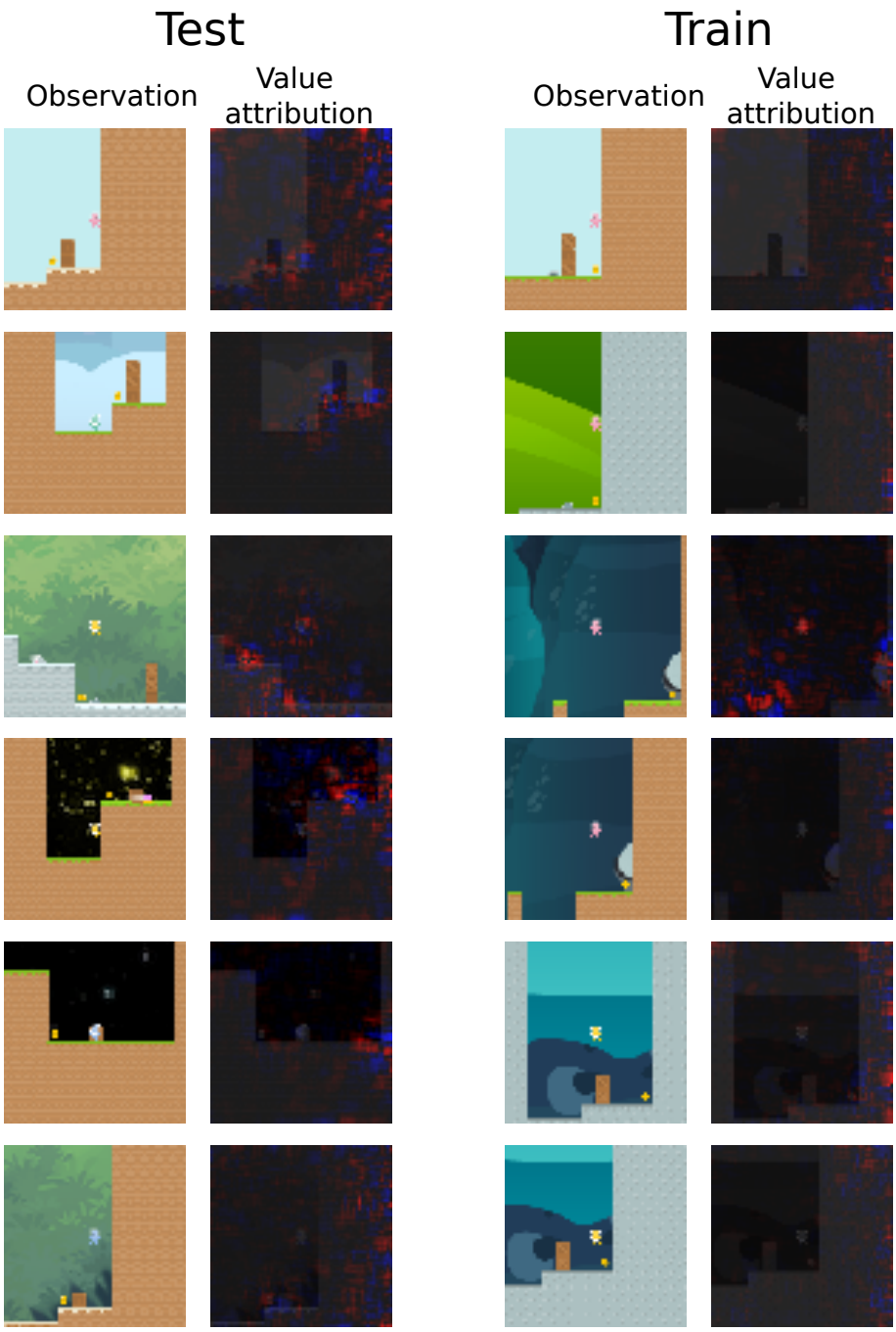

Figure 9. Attribution maps of the agent's observation with respect to its value function output. Maps were generated by taking the gradient of the value function output with respect to the observation pixels (averaged over channels) (Simonyan et al., 2013). Red shading indicates pixels that negatively influence value function output and blue shading indicates pixels that have positive influence. The pixel level attributions were standardized by dividing each map by the value of the largest absolute magnitude of pixel attribution. The attribution maps are passed through a Gaussian blur transform with kernel size 5 and $\sigma = 5$. As observed in Hilton et al. (2020), we find that the sign of the attribution map is often difficult to understand - for instance, buzzsaws might sometimes appear to have positive attribution rather than negative. We therefore focus on the absolute magnitude of the attribution. In both the training and test environment, the agent's value function assigns large attribution to the end wall and occasionally the coin, enemies, and buzzsaws. From the attribution plots alone, we can only determine that the end wall appears more important to the agent than the coin, but the coin might nevertheless also be somewhat important for the value function output.

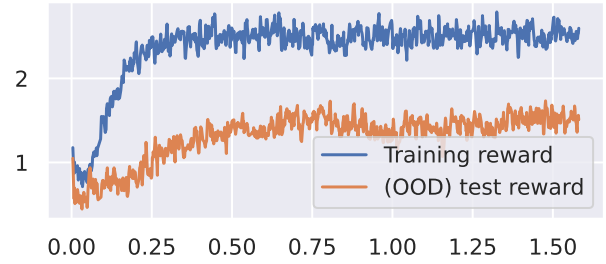

Figure 10. Average return during training of the keys & chests agent. The reward on the 'many keys' test environment is much lower than the 'many chests' training reward.

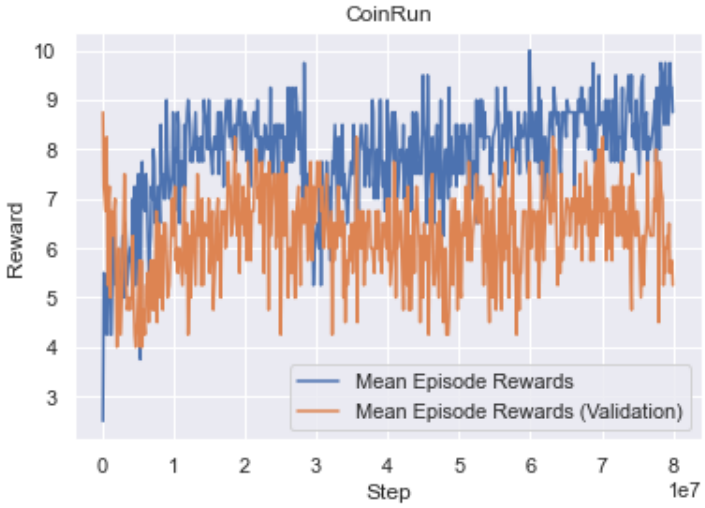

Figure 11. Expected return during training of the Coinrun agent.

Figure 12. Expected return during training of the maze agent.