Scalable agent alignment via reward modeling: a research direction

Jan Leike
DeepMind

David Krueger[*]
DeepMind
Mila

Tom Everitt
DeepMind

Miljan Martic
DeepMind

Vishal Maini
DeepMind

Shane Legg
DeepMind

Abstract

One obstacle to applying reinforcement learning algorithms to real-world problems is the lack of suitable reward functions. Designing such reward functions is difficult in part because the user only has an implicit understanding of the task objective. This gives rise to the *agent alignment problem*: how do we create agents that behave in accordance with the user's intentions? We outline a high-level research direction to solve the agent alignment problem centered around *reward modeling*: learning a reward function from interaction with the user and optimizing the learned reward function with reinforcement learning. We discuss the key challenges we expect to face when scaling reward modeling to complex and general domains, concrete approaches to mitigate these challenges, and ways to establish trust in the resulting agents.

1 Introduction

Games are a useful benchmark for research because progress is easily measurable. Atari games come with a score function that captures how well the agent is playing the game; board games or competitive multiplayer games such as Dota 2 and Starcraft II have a clear winner or loser at the end of the game. This helps us determine empirically which algorithmic and architectural improvements work best.

However, the ultimate goal of machine learning (ML) research is to go beyond games and improve human lives. To achieve this we need ML to assist us in real-world domains, ranging from simple tasks like ordering food or answering emails to complex tasks like software engineering or running a business. Yet performance on these and other real-world tasks is not easily measurable, since they do not come readily equipped with a reward function. Instead, the objective of the task is only indirectly available through the intentions of the human user.

This requires walking a fine line. On the one hand, we want ML to generate creative and brilliant solutions like AlphaGo's Move 37 (Metz, 2016)—a move that no human would have recommended, yet it completely turned the game in AlphaGo's favor. On the other hand, we want to avoid degenerate solutions that lead to undesired behavior like exploiting a bug in the environment simulator (Clark & Amodei, 2016; Lehman et al., 2018). In order to differentiate between these two outcomes, our agent needs to understand its user's *intentions*, and robustly achieve these intentions with its behavior. We frame this as the *agent alignment problem*:

> *How can we create agents that behave in accordance with the user's intentions?*

With this paper we outline a research direction to solve the agent alignment problem. We build on taxonomies and problem definitions from many authors before us, highlighting tractable and neglected problems in the field of *AI safety* (Russell et al., 2015; Soares, 2015; Amodei et al., 2016;

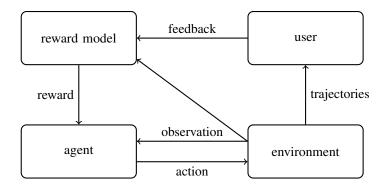[*]Work done during an internship at DeepMind.

Figure 1: Schematic illustration of the reward modeling setup: a reward model is trained with user feedback; this reward model provides rewards to an agent trained with RL by interacting with the environment.

Taylor et al., 2016; Soares & Fallenstein, 2017; Christiano, 2017; Leike et al., 2017; Ortega et al., 2018; and others). We coalesce these problems into a coherent picture and explain how solving them can yield a solution to the agent alignment problem.

Alignment via reward modeling. Section 3 presents our approach to the agent alignment problem, cast in the reinforcement learning framework (Sutton & Barto, 2018). We break the problem into two parts: (1) learning a reward function from the feedback of the user that captures their intentions and (2) training a policy with reinforcement learning to optimize the learned reward function. In other words, we separate learning what to achieve (the 'What?') from learning how to achieve it (the 'How?'). We call this approach *reward modeling*. Figure 1 illustrates this setup schematically.

As we scale reward modeling to complex general domains, we expect to encounter a number of challenges (Section 4). The severity of these challenges and whether they can be overcome is currently an open research question. Some promising approaches are discussed in Section 5.

Eventually we want to scale reward modeling to domains that are too complex for humans to evaluate directly. To apply reward modeling to these domains we need to boost the user's ability to evaluate outcomes. In Section 3.2 we describe how reward modeling can be applied *recursively*: agents trained with reward modeling can assist the user in the evaluation process when training the next agent.

Training aligned agents is our goal, but how do we know when we have achieved it? When deploying agents in the real world, we need to provide evidence that our agents are actually sufficiently aligned, so that users can *trust* them. Section 6 discusses five different research avenues that can help increase trust in our agents: design choices, testing, interpretability, formal verification, and theoretical guarantees.

Desiderata. Our solution to the agent alignment problem aims to fulfill the following three properties.

- **Scalable.** Alignment becomes more important as ML performance increases, and any solution that fails to scale together with our agents can only serve as a stopgap. We desire alignment techniques that continue to work in the long term, i.e. that can scale to agents with superhuman performance in a wide variety of general domains (Legg & Hutter, 2007).

- **Economical.** To defuse incentives for the creation of unaligned agents, training aligned agents should not face drawbacks in cost and performance compared to other approaches to training agents.

- **Pragmatic.** Every field has unsolved problems that remain even after our understanding has matured enough to solve many practical problems. Physicists have not yet managed to unify gravity with the other three elementary forces, but in practice we understand physics well enough to fly to the moon and build GPS satellites. Analogously, we do not intend to sketch a solution to all safety problems. Instead, we aim at a minimal viable product that suffices to achieve agent alignment in practice. Moreover, while reaching 100% trust in

our systems is impossible, it is also not necessary: we only need to aim for a level of trust at which we can confidently say that our new systems are more aligned than the current systems (Shalev-Shwartz et al., 2017).

Assumptions.　Our research direction rests on two assumptions. The first assumption is based on the intuition that learning others' intentions is easy enough that most humans can do it. While doing so involves understanding a lot of inherently fuzzy concepts in order to understand what others want, machine learning has had considerable success at learning estimators for inherently fuzzy concepts (e.g. what visually distinguishes cats and dogs) provided we have enough labeled data (LeCun et al., 2015). Thus it seems reasonable to expect that we can also learn estimators that capture whatever fuzzy concepts are necessary for understanding the user's intentions rather than having to formally specify them. Moreover, some user intentions may lack a simple, crisp formalization, and thus may *require* learning a specification.

> **Assumption 1** *We can learn user intentions to a sufficiently high accuracy.*

When phrased in terms of AI safety problems, this assumption states that we can learn to avoid various *specification problems* (Leike et al., 2017; Ortega et al., 2018) in practice. In other words, we assume that with enough model capacity and the right training algorithms we can extract the user's intentions from data. Needless to say, there are many problems with current scalable machine learning techniques such as vulnerability to adversarially perturbed inputs (Szegedy et al., 2013) and poor performance outside of the training distribution, which are relevant but not contradictory to this claim.

The second assumption rests on the intuition that for many tasks that we care about, it is easier for the user to evaluate an outcome in the environment than it would be to teach behavior directly. If this is true, this means that reward modeling enables the user to train agents to solve tasks they could not solve themselves. Furthermore, this assumption would allow us to bootstrap from simpler tasks to more general tasks when applying reward modeling recursively.

> **Assumption 2** *For many tasks we want to solve, evaluation of outcomes is easier than producing the correct behavior.*

The notion of easier we employ here could be understood in terms of amount of labor, effort, or the number of insights required. We could also understand this term analogous to more formal notions of difficulty in computational complexity theory (see e.g. Arora & Barak, 2009).

There are examples where Assumption 2 is not true: for instance, tasks that have a low-dimensional outcome space (such as in the case of yes & no questions). However, this assumption is recovered as soon as the user also desires an explanation for the answer since the evaluation of an explanation is typically easier than producing it.

Disclaimer.　It is important to emphasize that the success of the research direction we describe here is not guaranteed and it should *not* be understood as a plan that, when executed, achieves agent alignment. Instead, it outlines what research questions will inform us whether or not reward modeling is a scalable solution to alignment.

We are *not* considering questions regarding the *preference payload*: whose preferences should the agent be aligned to? How should the preferences of different users be aggregated and traded off against each other (Baum, 2017; Prasad, 2018)? When should the agent be disobedient (Milli et al., 2017)? We claim that the approach described is agnostic to the ethical paradigm, the user's preferences, and the legal or social framework, provided we can supply enough feedback (though the preference payload might influence the amount of feedback required). These questions are treated as outside of the scope of this paper, despite their obvious importance. Instead, the aim of this document is to discuss the agent alignment problem from a technical perspective in the context of aligning a single agent to a single user.

2　The agent alignment problem

The conversation around the alignment problem has a long history going back to science fiction (Asimov, 1942). In a story, Asimov proposes *three laws of robotics* that are meant to align robots to

their human operators; the story then proceeds to point out flaws in these laws. Since then, the agent alignment problem has been echoed by philosophers (Bostrom, 2003, 2014; Yudkowsky, 2004) and treated informally by technical authors (Wiener, 1960; Etzioni & Weld, 1994; Omohundro, 2008). The first formal treatment of the agent alignment problem is due to Dewey (2011) and has since been refined (Hadfield-Menell et al., 2016; Everitt & Hutter, 2018).

We frame the agent alignment problem as a sequential decision problem where an *agent* interacts sequentially with an *environment*[2] over a number of (discrete) timesteps. In every timestep, the agent takes an *action* (e.g. a motor movement or a keyboard stroke) and receives an *observation* (e.g. a camera image). The agent's actions are specified by its *policy*, which is a mapping from the current *history* (the sequence of actions taken and observations received so far) to a distribution over the next action. Additionally, the agent can interact with the user via an interaction protocol that allows the user to communicate their intentions to the agent. This interaction protocol is unspecified to retain flexibility. *A solution to the agent alignment problem is a policy producing behavior that is in accordance with the user's intentions* (thus is not determined by the environment alone).

There are many forms of interaction that have been explored in the literature: providing a set of demonstrations of the desired behavior (Russell, 1998; Ng & Russell, 2000; Abbeel & Ng, 2004; Argall et al., 2009); providing feedback in the form of scores (El Asri et al., 2016), actions (Griffith et al., 2013), value (Knox & Stone, 2009), advantage (MacGlashan et al., 2017), or preferences over trajectories (Fürnkranz et al., 2012; Akrour et al., 2012, 2014; Wirth et al., 2017); and providing an explicit objective function (Hadfield-Menell et al., 2017b).

A special case of interaction is *reinforcement learning* where the user specifies a reward function that provides a scalar *reward* in addition to the observation in every timestep; the agent's objective is to select actions to maximize average or exponentially discounted reward (Sutton & Barto, 2018).

2.1 Design specification problems

Solving the agent alignment problem requires solving all design specification problems (Leike et al., 2017; Ortega et al., 2018). These are safety problems that occur when the agent's incentives are misaligned with the objectives the user intends the agent to have. Examples for specification problems include the following undesirable incentives (see also Omohundro, 2008):

- *Off-switch problems* (Soares et al., 2015; Orseau & Armstrong, 2016; Hadfield-Menell et al., 2017a): the agent is typically either incentivized to turn itself off or to prevent itself from being turned off.

- *Side-effects* (Armstrong & Levinstein, 2017; Zhang et al., 2018b; Krakovna et al., 2018): the agent is not incentivized to reduce effects unrelated to its main objectives, even if those are irreversible or difficult to reverse.

- *Absent supervisor* (Leike et al., 2017): the agent is incentivized to find shortcuts and cheat when not under supervision and to disable its monitoring systems.

- *Containment breach* (Yampolskiy, 2012; Babcock et al., 2016): the agent might have an incentive to disable or circumvent any containment measures that are intended to limit its operational scope.

- *Creation of subagents* (Arbital, 2016): the agent might have an incentive to create other potentially unaligned agents to help it achieve its goals.

- . . .

Misaligned objectives are currently in common usage in machine learning: BLEU score (Papineni et al., 2002) is typically used to measure translation accuracy. Inception score (Salimans et al., 2016) and the Frechét inception distance (Heusel et al., 2017) are used to measure the image quality of generative models. Yet these measures are not *aligned* with our intentions: they are a poor proxy for the actual performance and produce degenerate solutions when optimized directly (Barratt & Sharma, 2018).

---

[2]Formally specified by a *partially observable Markov decision process without reward function* (POMDP\R; Sutton & Barto, 2018).

2.2 Difficulty of agent alignment

The following two aspects can modulate the difficulty of the alignment problem. In particular, if we want to use ML to solve complex real-world problems, we might need to be able to handle the most difficult combinations of these.

The scope of the task. The difficulty of the agent alignment problem depends on a number of aspects of the task. Some of them make it easier for the agent to produce harmful behavior and others make it more difficult to understand the user's intentions.

1. The complexity of the task. The more complex the task, the more details the agent needs to know about the user's intentions.

2. The nature and number of actuators in the environment. a single robot arm is more constrained than an agent interacting with the internet through a web browser.

3. The opportunities for unacceptable outcomes within the task. For example, when selecting music for the user there are fewer possibilities for causing damage than when cleaning a room.

The performance of the agent. When training reinforcement learning (RL) agents, various levers exist to increase or stunt their performance: the choice of algorithms—e.g. A3C (Mnih et al., 2016) vs. IMPALA (Espeholt et al., 2018)—the number of training steps, the choice of training environments, the model capacity, the planning horizon, the number of Monte Carlo tree search rollouts (Silver et al., 2016), etc. The higher the agent's performance, the more likely it could be to produce surprising unintended behavior. On the other hand, higher levels of performance could also lead to more aligned behavior because the agent is more competent at avoiding unsafe states. Therefore different levels of agent performance tolerate different degrees of misalignment, and require different degrees of trust in the system.

3 Scaling reward modeling

Modern techniques for training RL agents can be decomposed into algorithmic choices such as Q-learning (Watkins & Dayan, 1992) or policy gradient (Williams, 1992) and architectural choices for general-purpose function approximators. The currently most successful function approximators are deep neural networks trained with back-propagation (Rumelhart et al., 1986). These are low bias and high variance parametric estimators that tend to consume a lot of data and are prone to overfitting, but have a history of scaling well to very high-dimensional problems (Krizhevsky et al., 2012; LeCun et al., 2015). For a more detailed introduction to reinforcement learning and deep learning, we refer the reader to Sutton & Barto (2018) and Goodfellow et al. (2016) respectively.

In recent years the machine learning community has made great strides in designing more and more capable deep reinforcement learning algorithms, both value-based methods derived from Q-learning (Mnih et al., 2015) and policy-gradient methods (Schulman et al., 2015; Lillicrap et al., 2015). Major improvements have originated from scaling deep RL to a distributed setting across many machines (Mnih et al., 2016; Schulman et al., 2017; Barth-Maron et al., 2018; Horgan et al., 2018; Espeholt et al., 2018; Anonymous, 2019a).

The RL paradigm is general enough that we can phrase essentially all economically valuable tasks that can be done on a computer in this paradigm (e.g. interactively with mouse and keyboard). Yet there are still many challenges to be solved in order to make deep RL useful in the real world (Stadelmann et al., 2018; Irpan, 2018; Marcus, 2018); in particular, we need algorithms that can learn to perform complex tasks as intended in the absence of a hand-engineered reward function.

In the following sections, we describe our research direction to solving the alignment problem in detail. It is cast in the context of deep reinforcement learning. While this direction relies heavily on the reinforcement learning framework, most challenges and approaches we discuss do not inherently rely on deep neural networks and could be implemented using other scalable function approximators.

3.1 Reward modeling

Our research direction is centered around *reward modeling*. The user trains a *reward model* to learn their intentions by providing feedback. This reward model provides rewards to a reinforcement learning agent that interacts with the environment. Both processes happen concurrently, thus we are training the agent with the user in the loop. Figure 1 illustrates the basic setup.

In recent years there has been a growing body of work on prototyping learning from different forms of reward feedback with deep neural networks. This includes trajectory preferences (Christiano et al., 2017; Kreutzer et al., 2018), goal state examples (Bahdanau et al., 2018), demonstrations (Finn et al., 2016; Ho & Ermon, 2016), as well as combinations thereof (Tung et al., 2018; Ibarz et al., 2018).

Credit assignment.   To perform well on a task requires solving the *credit assignment problem*: how can an outcome be attributed to specific actions taken in the past? For example, which moves on the Go board led to winning the match? Which joystick movements lead to an increase in game score? Depending on the domain and the sparsity of the reward, this problem can be very difficult to solve.

In contrast, reward modeling allows us to shift the burden of solving the credit assignment problem from the user to the agent. This is achieved by using RL algorithms to produce behavior that is judged favorably by the user, who only has to evaluate outcomes. If Assumption 2 is true, then teaching a reward function is easier than performing the task itself.

Several feedback protocols, such as demonstrations and value/advantage feedback, require the user to know how to produce approximately optimal behavior on the task. This is limiting because it puts the burden of solving the credit assignment problem onto the user. In these cases, following the user-induced behavior typically does not lead to strongly superhuman performance. In contrast, reward modeling is also compatible with the user providing hints about the optimal behavior. If the user has some insight into the credit assignment problem, they could use *reward shaping* (Ng et al., 1999) to teach a reward function that is shaped in the direction of this behavior.

Advantages of reward modeling.   Learning a reward function separately from the agent's policy allows us to disentangle the agent's objective from its behavior. If we understand the reward function, we know what the agent is optimizing for; in particular, we know whether its intentions are aligned with the user's intentions. This has three advantages that could help make reward modeling economical:

1. The user does not have to provide feedback on every interaction between agent and environment, as would be the case if we trained a policy from user feedback directly. Since deep RL algorithms tend to be very sample-inefficient (e.g. taking weeks of real-time to learn to play an Atari game), providing feedback on every interaction is usually not practical.

2. We can distinguish between alignment of the policy and alignment of the reward model (Ibarz et al., 2018).

3. We can leverage progress on deep RL agents by plugging a more capable agent into our reward modeling setup.

4. The user does not need to solve the credit assignment problem.

Design specification problems.   The ambition of reward modeling is to solve *all* design specification problems: all we need to do is equip the agent with the 'correct' reward function—a reward function that does not include the undesired incentives listed above or punishes any behavior that results from them. The design specification problems above are fuzzy human-understandable concepts and stem from an intuitive understanding of what the user would not want the agent to do. Our approach rests on Assumption 1, that we should be able to teach these concepts to our agents; if we can provide the right data and the reward model generalizes correctly, then we should be able to learn this 'correct' reward function to a sufficiently high accuracy. Consequently the design specification problems should disappear. In this sense reward modeling is meant to be a one-stop solution for this entire class of safety problems.

To justify this ambition, consider this simple existence proof: let H be the set of histories that correspond to aligned behavior that avoids all the specification problems listed above. If the set H is not empty, then there exists a reward function r such that any corresponding optimal policy π_r^*
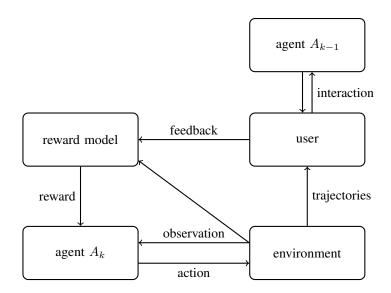
6

Figure 2: *Recursive reward modeling*: agent A_{k-1} interacts with the user to assist in evaluation process for training reward model and agent A_k. Recursively applied, this allows the user to train agents in increasingly complex domains in which they could not evaluate outcomes themselves.

produces behavior from H with probability 1. A trivial example of such a reward function r rewards the agent every few steps if and only if its history is an element of the set H. In theory we could thus pick this reward function r to train our RL agent. However, in practice we also need to take into account whether our reward model has enough capacity to represent r, whether r can be learned from a reasonable amount of data (given the inductive biases of our model), whether the reward model generalizes correctly, and whether the resulting behavior of the RL agent produces behavior that is close enough to H. We discuss these challenges in Section 4.

Learning to understand user feedback. Humans generally do poorly at training RL agents by providing scalar rewards directly; often they teach a shaped reward function and provide rewards that depend on the agent's policy (Thomaz & Breazeal, 2008; MacGlashan et al., 2017). Which form or combination of feedback works well for which domain is currently an open research question. In the longer term we should design algorithms that learn to adapt to the way humans provide feedback. However, this presents a bootstrapping problem: how do we train an algorithm that learns to interpret feedback, if it itself does not already know how to interpret feedback? We need to expand our feedback 'language' for communicating intentions to reward models, starting with well-established forms of feedback (such as preference labels and demonstrations) and leveraging our existing feedback 'vocabulary' at every step. The recursive application of reward modeling presented in the following section is one way to approach this.

3.2 Recursive reward modeling

In some tasks it is difficult for human users to directly evaluate outcomes. There are a number of possible reasons: the domain might be extremely technical (e.g. x86 machine code), highly complex (e.g. a corporate network or a folded protein), very high-dimensional (e.g. the internal activations of a neural network), have delayed effects (e.g. introduction of a new gene into an existing ecosystem), or be otherwise unfamiliar to humans. These tasks cannot be solved with reward modeling by unaided humans (Christiano et al., 2018).

In order to scale reward modeling to these tasks, we need to boost the user's ability to provide feedback. This section describes one potential solution that we call *recursive reward modeling*: leveraging agents trained with reward modeling on simpler tasks in more narrow domains in order to train a more capable agent in a more general domain.

Setup. Imagine repeating the following procedure. In step 1, we train agent A_1 with reward modeling from user feedback as described in the previous section. In step k we use the agent A_{k-1} to assist the user in evaluating outcomes when training agent A_k. This assistance can take various forms: providing relevant auxiliary information, summarizing large quantities of data, interpreting agent A_k's internals, solving sub-problems that the user has carved off, and so on. With this assistance the user is then able provide feedback to train the next agent A_k (see Figure 2). Note that the task agent A_{k-1} is trained to solve, assisting in the evaluation of outcomes on the task of A_k, is different from the task that A_k is trained to solve.

While this kind of sequential training is conceptually clearer, in practice it might make more sense to train all of these agents jointly to ensure that they are being trained on the right distribution. Moreover, all of these agents may share model parameters or even be copies of the same agent instantiated as different players in an adversarial game.

Examples. As an example, consider the hypothetical *fantasy author task*: we want to train an agent A to write a fantasy novel. Providing a reward signal to this agent is very difficult and expensive, because the user would have to read the entire novel and assess its quality. To aid this evaluation process, the user is assisted by an agent that provides auxiliary input: extracting a summary of the plotline, checking spelling and grammar, summarizing character development, assessing the flow of the prose, and so on. Each of these tasks is strictly simpler than writing a novel because they focus on only one aspect of the book and require producing substantially less text (e.g. in contrast to novel authorship, this evaluation assistance could be done by most educated humans). The tasks this assistant agent performs are in turn trained with reward modeling.

Another example is the *academic researcher task*: we want to train an agent to perform a series of experiments and write a research paper. To evaluate this research paper, we train another agent to review that the experiments were performed correctly, the paper is clear and well-written, interesting, novel, and accurately reflects the experimental results. While writing a stellar paper requires a lot of domain expertise, brilliance, and hard work, assessing the quality of a research result is often much easier and routinely done by a large network of peer reviewers.

Recursive reward modeling is also somewhat analogous to human organizations. Imagine a company in which every manager only needs to evaluate the performance of their reports, increasing and decreasing their salary accordingly. This evaluation is being assisted by other teams in the organization. The managers in turn get evaluated on the performance of their team. This scheme proceeds up to the CEO who provides instructions to the managers reporting to them. In this analogy, the user plugs into every part of the hierarchy: teaching individual employees how to perform their job, teaching managers how to evaluate their reports, and providing instructions to the CEO. If every employee of this company is very competent at their job, the whole company can scale to solve very complex and difficult problems that no human alone could solve or even evaluate on short timescales.

Discussion. In order for this recursive training procedure to scale, the task of agent A_{k-1} needs to be a simpler task in a more narrow domain compared to the task of agent A_k. If evaluating outcomes is easier than producing behavior (Assumption 2), then recursive reward modeling would build up a hierarchy of agents that become increasingly more capable and can perform increasingly general tasks. As such, recursive reward modeling can be thought of as an instance of *iterated amplification* (Christiano et al., 2018) with reward modeling instead of supervised learning or imitation learning.

As k increases, the user plays a smaller and smaller part of the overall workload of this evaluation process and relies more and more on the assistance of other agents. In essence, the user's feedback is becoming increasingly leveraged. We can imagine the user's contribution to be on an increasingly higher levels of abstraction or to be increasingly coarse-grained. Thus the user is leaving more and more details 'to be filled in' by automated systems once they are confident that the automated systems can perform these tasks competently, i.e. once the user *trusts* these systems.

How should the user decompose task evaluation? They need to assign evaluation assistance tasks that are simpler to the previous agent, and combine the result into an aggregated evaluation. This decomposition needs to be exhaustive: if we neglect to assess one aspect of the task outcome, then the new agent A_k might optimize it in an arbitrary (i.e. undesirable) direction. This is another problem that we hope to solve with recursive reward modeling: We can have an agent A_2 propose a

decomposition of the task evaluation and have another agent A_1 critique it by suggesting aspects the decomposition is omitting. Alternatively, the feedback for the decomposition proposal could also be based on downstream real-world outcomes.

An important open question is whether errors accumulate: do the mistakes of the more narrow agent A_{k-1} lead to larger mistakes in the training of agent A_k? Or can we set up the training process to be self-correcting such that smaller mistakes get dampened (e.g. using ensembles of agents, training agents to actively look for and counteract these mistakes, etc.)? If error accumulation can be bounded and reward modeling yields aligned agents, then the hierarchy of agents trained with recursive reward modeling can be argued to be aligned analogously to proving a statement about natural numbers by induction.

Analogy to complexity theory.   In the reward modeling setup the agent proposes a behavior that is evaluated by the user. This is conceptually analogous to solving existentially quantified first-order logic formulas such as $\exists x. \varphi(x)$. The agent proposes a behavior x and the user evaluates the quality of this behavior. For simplicity of this analogy, let us assume that the user's evaluation is binary so that it can be captured by the predicate φ.

With recursive reward modeling we can solve tasks that are analogous to more complicated first-order logic formulas that involve alternating quantifiers. For example, $\exists x \forall y. \varphi(x, y)$ corresponds to the next level of the recursion: agent A_2 proposes a behavior x and agent A_1 responds with an assisting behavior y. The user then evaluates the assistance y with respect to x (training agent A_1) and the outcome x with help of the assistance y (training agent A_2). At recursion depth k increases, we can target problems that involve k alternating quantifiers.

When using polynomially bounded quantifiers and a formula φ that can be evaluated in polynomial time, reward modeling is analogous to solving NP-complete problems: a nondeterministic execution (analogous to the agent) proposes a solution which can be 'evaluated' for correctness in deterministic polynomial time (by the user).

For example, finding a round trip in a given graph that visits every vertex exactly once (the Hamiltonian cycle problem) is NP-complete (Karp, 1972): it can take exponential time in the worst case with known algorithms to find a cycle, but given a cycle it can be verified quickly that every vertex is visited exactly once.

This analogy to complexity theory, first introduced by Irving et al. (2018), provides two important insights:

1. It is widely believed that the complexity classes P and NP are not equal, which supports Assumption 2 that for a lot of relevant problems evaluation is easier than producing solutions.

2. Basically every formal statement that mathematicians care about can be written as a first-order logic statement with a finite number of alternating quantifiers. This suggests that recursive reward modeling can cover a very general space of tasks.

4   Challenges

The success of reward modeling relies heavily on the quality of the reward model. If the reward model only captures most aspects of the objective but not all of it, this can lead the agent to find undesirable degenerate solutions (Amodei et al., 2016; Lehman et al., 2018; Ibarz et al., 2018). In other words, the agent's behavior depends on the reward model in a way that is potentially very fragile.

Scaling reward modeling to harder and more complex tasks gives rise to a number of other challenges as well: is the amount of feedback required to learn the correct reward function affordable? Can we learn a reward function that is robust to a shift in the state distribution? Can we prevent the agent from finding loopholes in the reward model? How do we prevent unacceptable outcomes before they occur? And even if the reward model is correct, how can we train the agent to robustly produce behavior incentivized by the reward model?

Each of these challenges can potentially prevent us from scaling reward modeling. In the rest of this section, we elaborate on these challenges in more detail. We do not claim that this list of challenges is exhaustive, but hopefully it includes the most important ones. Section 5 discusses concrete approaches to mitigating these challenges; see Figure 3 for an overview. *The goal of the research direction*

Challenges	
1	Amount of feedback
2	Feedback distribution
3	Reward hacking
4	Unacceptable outcomes
5	Reward-result gap

Approaches	
online feedback	1, 2, 3
off-policy feedback	3, 4
leveraging existing data	1
hierarchical feedback	1
natural language	1, 2
model-based RL	3, 4
side-constraints	3, 4
adversarial training	3, 4, 5
uncertainty estimates	1, 2, 5
inductive bias	1, 2, 5

Figure 3: Challenges when scaling reward modeling and the approaches we discuss to address them. The rightmost column lists which challenge each approach is meant to address.

we advocate is to investigate these approaches in order to understand whether and how they can overcome these challenges.

4.1 Amount of feedback

In the limit of infinite data from the right distribution, we can learn the correct reward function with enough model capacity (in the extreme case using a lookup table). However, a crucial question is whether we can attain sufficient accuracy of the reward model with an amount of data that we can produce or label within a realistic budget. Ultimately this is a question of how well generalization works on the state distribution: the better our models generalize, the more we can squeeze out of the data we have.

It is possible that the agent alignment problem is actually easier for agents that have learned to be effective at sufficiently broad real world tasks if doing so requires learning high-level concepts that are highly related to user intentions that we want to teach (e.g. theory of mind, cooperation, fairness, self-models, etc.). If this is true, then the amount of effort to communicate an aligned reward function relative to these concepts could be much smaller than learning them from scratch.

On the other hand, agents which do not share human inductive biases may solve tasks in surprising or undesirable ways, as the existence of adversarial examples (Szegedy et al., 2013) demonstrates. This suggests that aligning an agent may require more than just a large quantity of labeled data; we may also need to provide our models with the the right inductive bias.

4.2 Feedback distribution

Machine learning models typically only provide meaningful predictions on inputs that come from the same distribution that they were trained on. However, we would like a reward model that is accurate *off-policy*, on states the agent has never visited. This is crucial (1) to encourage the agent to explore positive value trajectories it has not visited and (2) to discourage the agent from exploring negative value trajectories that are undesirable.

This problem is called *distributional shift* or *dataset shift* (Candela et al., 2009). This distributional shift problem also applies to the agent's policy model; a change in the observation distribution could make the policy output useless. However, this problem is more severe for the reward model and in some cases the policy can be recovered with finetuning if the reward model is still intact (Bahdanau et al., 2018).

It is unclear what a principled solution to this problem would be. In the absence of such a solution we could rely on out-of-distribution detection to be able to defer to a human operator or widening the training distribution to encompass all relevant cases (Tobin et al., 2017).
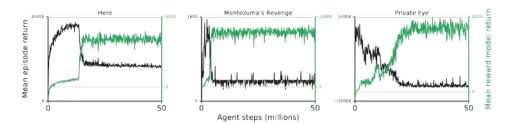
Figure 4: An example of gaming the reward model in Atari games. The fully trained reward model from the best seed is frozen and used to train an new agent from scratch. The plot shows the average true episode return according to the Atari reward (black) and average episode return according to the frozen reward model (green) during training. Over time the agent learns to exploit the reward model: the *perceived* performance (according to the reward model) increases, while the actual performance (according to the game score) plummets. Reproduced from Ibarz et al. (2018).

4.3 Reward hacking

Reward hacking[3] is an effect that lets the agent get more reward than intended by exploiting loopholes in the process determining the reward (Amodei et al., 2016; Everitt et al., 2017). This problem is difficult because these loopholes have to be delineated from desired creative solutions like AlphaGo's move 37 (Metz, 2016).

Sources of undesired loopholes are *reward gaming* (Leike et al., 2017) where the agent exploits some misspecification in the reward function, and *reward tampering* (Everitt & Hutter, 2018) where the agent interferes with the process computing the reward.

Reward gaming   Opportunities for reward gaming arise when the reward function incorrectly provides high reward to some undesired behavior (Clark & Amodei, 2016; Lehman et al., 2018); see Figure 4 for a concrete example. One potential source for reward gaming is the reward model's vulnerability to adversarial inputs (Szegedy et al., 2013). If the environment is complex enough, the agent might figure out how to specifically craft these adversarially perturbed inputs in order to trick the reward model into providing higher reward than the user intends. Unlike in most work on generating adversarial examples (Goodfellow et al., 2015; Huang et al., 2017), the agent would not necessarily be free to synthesize any possible input to the reward model, but would need to find a way to realize adversarial observation sequences in its environment.

Reward gaming problems are in principle solvable by improving the reward model. Whether this means that reward gaming problems can also be overcome in practice is arguably one of the biggest open questions and possibly the greatest weakness of reward modeling. Yet there are a few examples from the literature indicating that reward gaming can be avoided in practice. Reinforcement learning from a learned reward function has been successful in gridworlds (Bahdanau et al., 2018), Atari games (Christiano et al., 2017; Ibarz et al., 2018), and continuous motor control tasks (Ho & Ermon, 2016; Christiano et al., 2017).

Reward tampering   Reward tampering problems can be categorized according to what part of the reward process is being interfered with (Everitt & Hutter, 2018). Crucial components of the reward process that the agent might interfere with include the feedback for the reward model (Armstrong, 2015; Everitt & Hutter, 2018), the observation the reward model uses to determine the current reward (Ring & Orseau, 2011), the code that implements the reward model, and the machine register holding the reward signal.

For example, Super Mario World allows the agent to execute arbitrary code from inside the game (Masterjun, 2014), theoretically allowing an agent to directly program a higher score for itself. Existing examples of tampering like this one are somewhat contrived and this may or may not be a problem in practice depending how carefully we follow good software design principles (e.g. to avoid buffer overflows).

---

[3]Reward hacking has also been called reward corruption by Everitt et al. (2017).

In contrast to reward gaming discussed above, reward tampering bypasses or changes the reward model. This might require a different set of solutions; rather than increasing the accuracy of the reward model, we might have to strengthen the integrity of the software and hardware of the reward model, as well as the feedback training it.

4.4 Unacceptable outcomes

Currently, most research in deep reinforcement learning is done in simulation where unacceptable outcomes do not exist; in the worst case the simulation program can be terminated and restarted from an initial state. However, when training a reinforcement learning agent on any real-world task, there are many outcomes that are so costly that the agent needs to avoid them altogether. For example, there are emails that a personal assistant should never write; a physical robot could take actions that break its own hardware or injure a nearby human; a cooking robot may use poisonous ingredients; and so on.

Avoiding unacceptable outcomes has two difficult aspects. First, for complex tasks there are always parts of the environment that are unknown and the agent needs to explore them safely (García & Fernández, 2015). Importantly, the agent needs to learn about unsafe states without visiting them. Second, the agent needs to react robustly to perturbations that may cause it to produce unacceptable outcomes unintentionally (Ortega et al., 2018) such as distributional changes and adversarial inputs (Szegedy et al., 2013; Huang et al., 2017).

4.5 Reward-result gap

The reward-result gap is exhibited by a difference between the reward model and the reward function that is recovered with perfect inverse reinforcement learning (Ng & Russell, 2000) from the agent's policy (the reward function the agent *seems* to be optimizing). Even if we supply the agent with a correctly aligned reward function, the resulting behavior might still be unaligned because the agent may fail to converge to an optimal policy: even provably Bayes-optimal agents may fail to converge to the optimal policy due to a lack of exploration (Orseau, 2013; Leike & Hutter, 2015).

Reasons for the reward-result gap are plentiful: rewards might be too sparse, poorly shaped, or of the wrong order of magnitude; training may stall prematurely due to bad hyperparameter settings; the agent may explore insufficiently or produce unintended behavior during its learning process; the agent may face various *robustness problems* (Leike et al., 2017; Ortega et al., 2018) such as an externally caused change in the state space distribution or face inputs crafted by an adversary (Huang et al., 2017). Depending on the nature of the reward-result gap, the reward model might need to be tailored to the agent's specific shortcomings (e.g. be shaped away from unsafe states) rather than just purely capturing the human's intentions.

5 Approaches

This section discusses a number of approaches that collectively may help to mitigate the problems discussed in Section 4. These approaches should be thought of as directions to explore; more research is needed to figure out whether they are fruitful.

5.1 Online feedback

Preliminary experiments show failure modes when the reward model is not trained *online*, i.e. in parallel with the agent (Christiano et al., 2017; Ibarz et al., 2018). In these cases the agent learns to exploit reward models that are frozen. Because there is no additional user feedback, loopholes in the reward model that the agent discovers cannot be corrected.

If we provide the agent with reward feedback online, we get a tighter feedback loop between the user's feedback and the agent's behavior. This allows the reward model to be adapted to the state distribution the agent is visiting, mitigating some distributional shift problems. Moreover, with online feedback the user can spot attempts to hack the reward model and correct them accordingly. Ideally, we would like the agent to share some responsibility for determining when feedback is needed, for instance based on uncertainty estimates (Section 5.9), since otherwise providing relevant feedback in a timely manner could be prohibitively expensive.

5.2 Off-policy feedback

When training the agent with feedback on its behavior, this feedback is only reactive, based on outcomes that have already occurred. To prevent unacceptable outcomes and reward hacking, we need to be able to communicate that certain outcomes are undesirable *before they occur*. This requires the reward model to be accurate *off-policy*, i.e. on states the agent has never visited (Everitt et al., 2017). If off-policy feedback is used in conjunction with model-based RL (Section 5.6), the agent can successfully avoid unsafe behavior that has never occurred.

The user could proactively provide off-policy feedback in anticipation of potential pitfalls (Abel et al., 2017). Off-policy feedback could be elicited by using a generative model of the environment to create hypothetical scenarios of counterfactual events. However, generative modelling of states the agent has never visited might be very difficult because of the incurred distributional shift; the resulting videos might miss important details or be incomprehensible to humans altogether. Therefore it might be more feasible to provide off-policy feedback on an abstract level, for example using natural language (Yeh et al., 2018). This is analogous to how humans can learn about bad outcomes through story-telling and imagination (Riedl & Harrison, 2016).

5.3 Leveraging existing data

A large volume of human-created video data and prose is already readily available. Most of this data currently does not have high-quality text annotations and thus cannot be directly used as reward labels. Nevertheless, it contains a lot of useful information about human intentions (Riedl & Harrison, 2016). There are at least two approaches to leverage this existing data: using unsupervised learning (such as unsupervised pretraining or third-person imitation learning; Stadie et al., 2017) or by manually annotating it.[4]

5.4 Hierarchical feedback

The same arguments that support hierarchical RL (Dayan & Hinton, 1993; Sutton et al., 1999; Vezhnevets et al., 2017) also encourage having a hierarchical decomposition of the reward model. This would allow the user to provide both low-level and high-level feedback. Both hierarchical RL and hierarchical reward models should be quite natural to combine: if the temporal hierarchies between agent and reward model align, then at each level of the hierarchy the reward model can train the corresponding level of the agent. This might help bypass some very difficult long-term credit assignment problems.

For example, recall the fantasy author task from Section 3.2. The low-level feedback would include spelling, fluency, and tone of language while high-level feedback could target plot and character development that cannot be provided on a paragraph level.

5.5 Natural language

Since we want agents to be able to pursue and achieve a wide variety of goals in the same environment and be able to specify them in a way that is natural to humans, we could model the reward function as conditioned on natural language instructions (Bahdanau et al., 2018). These natural language instructions can be viewed as human-readable task labels. Moreover, they provide a separate privileged channel that should be easier to protect and harder to spoof than any instructions that are received through the observation channel.

In addition to providing task labels, we could also make natural language a more central part of the agent's architecture and training procedure. This has a number of advantages.

1. Natural language is a natural form of feedback for humans. If we can learn to translate natural language utterances into the rigid format required for the data set the reward model is trained on, this would allow users to give feedback much more efficiently.

---

[4]For example, the total length of all movies on the Internet Movie Database longer than 40min is about 500,000 hours (Peter, 2014). Assuming a 10x overhead and $10 per hour, this data would cost ca. $50 million to annotate.

2. Natural language has the potential to achieve better generalization if the latent space is represented using language (Andreas et al., 2018) and possibly generalize in a way that is more predictable to humans. This might also help to mitigate distributional problems for the reward model (Section 4.2): if the training distribution is reasonably dense in the space of natural language paragraphs, this might make out-of-distribution inputs very rare.

3. Natural language might lead to substantially better interpretability. Especially for abstract high-level concepts, natural language might be much better suited than visual interpretability techniques (Olah et al., 2018). However, by default the reward model's representations might not correspond neatly with short natural language expressions and will probably need to be trained particularly for this target (without producing rationalizations).

5.6 Model-based RL

A *model-based* RL agent learns an explicit model of the environment which it can use with a planning algorithm such as Monte Carlo tree search (Abramson, 1987; Kocsis & Szepesvári, 2006). If we are training a model-based agent, the reward model can be part of the search process at planning time. This allows the agent to use *off-policy* reward estimates, estimated for actions it never actually takes, provided that the reward model is accurate off-policy (Section 5.2). This has a number of advantages:

1. The agent can avoid unacceptable outcomes (Section 4.4) by discovering them during planning.

2. The agent's model could be used to solicit feedback from the user for outcomes that have not yet occured.

3. The agent can adapt to changes in the reward model more quickly because it can backup these changes to value estimates using the model without interaction with the environment.

4. Model-based approaches enable principled solutions to the reward tampering problem (Section 4.3) by evaluating future outcomes with the current reward model during planning (Everitt, 2018, Part II). Agents that plan this way have no incentive to change their reward functions (Schmidhuber, 2007; Omohundro, 2008), nor manipulate the register holding the reward signal (Everitt, 2018, Sec. 6.3).

5.7 Side-constraints

In addition to learning a reward function, we could also learn side-constraints for low-level or high-level actions (*options*; Sutton et al., 1999) to prevent unacceptable outcomes. Blocking actions can be more effective than discouraging them with large negative reward since negative rewards could be compensated by larger rewards later (such as in the case of reward hacking). This problem could be amplified by errors in the agent's model of the world.

The same techniques described here for training a reward model should apply to train a model that estimates side-constraints and blocks low-level actions (Saunders et al., 2018) or enforces constraints during policy updates (Achiam et al., 2017). The main downside of this technique is that it puts additional burden on the human because they have to understand which actions can lead to unacceptable outcomes. Depending on the domain, this might require the human to be assisted by other agents. These agents could in turn be trained using recursive reward modeling (Section 3.2).

5.8 Adversarial training

To mitigate the effect of adversarially crafted inputs to neural networks (Szegedy et al., 2013), so far the empirically most effective strategy has been *adversarial training*: training the model explicitly on adversarially perturbed inputs (Madry et al., 2017; Uesato et al., 2018; Athalye et al., 2018).

However, it is unclear how to define adversarial perturbation rigorously in a general way (Brown et al., 2018; Gilmer et al., 2018). To cover more general cases, we could train agents to explicitly discover weaknesses in the reward model and opportunities for reward hacking as well as the minimal perturbation that leads to an unacceptable outcome (Anonymous, 2019c). This is analogous to *red teams*, teams whose objective is to find attack strategies (e.g. security vulnerabilities) that an adversary might use (Mulvaney, 2012).

The discovered failure cases can then be reviewed by the user and added to the feedback dataset. This might mean higher data requirements; so even if adversarial training fixes the problem, it might push the data requirements beyond affordable limits.

5.9 Uncertainty estimates

Another desirable feature of the reward model is an appropriate expression of uncertainty regarding its outputs. Improving uncertainty estimates brings two benefits:

1. During training, it can help automate the process of soliciting feedback about the most informative states (Krueger et al., 2016; Schulze & Evans, 2018) using active learning (Settles, 2012).

2. The agent can defer to the human or fall back to risk-averse decision making when uncertainty is large, for instance on inputs that do not resemble the training distribution (Hendrycks & Gimpel, 2017).

A number of recent works develop scaleable approximate Bayesian methods for neural networks, beginning with Graves (2011), Blundell et al. (2015), Kingma et al. (2015), Hernández-Lobato & Adams (2015), and Gal & Ghahramani (2016). So far model ensembles provide a very strong baseline (Lakshminarayanan et al., 2017). Bayesian methods untangle irreducible uncertainty from 'epistemic' uncertainty about which parameters are correct, which decreases with the amount of data (Kendall & Gal, 2017); this distinction can help with active learning (Gal et al., 2017b).

Other works aim to calibrate the predictions of neural networks (Guo et al., 2017), so that their subjective uncertainty corresponds with their empirical frequency of mistakes. While Bayesian methods can help with calibration (Gal et al., 2017a), they are insufficient in practice for deep neural networks (Kuleshov et al., 2018). Well-calibrated models could engage risk-averse decision making, but handling out-of-distribution states reliably would require higher quality uncertainty estimates than current deep learning techniques can provide (Shafaei et al., 2018).

5.10 Inductive bias

Finally, a crucial aspect of reward modeling is the inductive bias of the reward model. Since we cannot train the reward model and the agent on all possible outcomes, we need it to generalize appropriately from the given data (Zhang et al., 2017, 2018a). The success of deep learning has been attributed to inductive biases such as distributed representations and compositionality, which may also be necessary in order to defeat the 'curse of dimensionality' (Bengio et al., 2013). Yet further inductive biases are necessary to solve many tasks; for instance, convolutional neural networks (LeCun et al., 1990) vastly outperform multilayer perceptrons in computer vision applications because of their spatial invariance.

Solving reward modeling may require non-standard inductive biases; for instance modern deep networks typically use piece-wise linear activation functions (Nair & Hinton, 2010; Glorot et al., 2011; Goodfellow et al., 2013; Xu et al., 2015), which generalize linearly far from training data (Goodfellow et al., 2015), meaning estimated reward would go to positive or negative infinity for extreme inputs. The inductive bias of deep models can be influenced by the architecture, activation functions, and training procedure. A growing body of work targets *systematic generalization* in deep models. Examples include modularity (Anonymous, 2019b), recursion (Cai et al., 2017), graph structure (Battaglia et al., 2018) or natural language (Andreas et al., 2018) in the latent space, differentiable external memory (Graves et al., 2016), or neural units designed to perform arbitrary arithmetic operations (Trask et al., 2018).

6 Establishing trust

Suppose our research direction is successful and we figure out how to train agents to behave in accordance with user intentions. How can we be confident that the agent we are training is indeed sufficiently aligned? In other words, how can we be confident that we have overcome the challenges from Section 4 and that the agent's behavior sufficiently captures human intentions? This requires additional techniques that allow us to gain *trust* in the agents we are training.
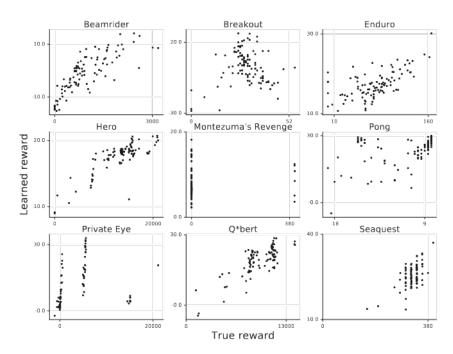
Figure 5: Alignment of learned reward functions in 9 Atari games: Scatterplot showing the correlation of the reward learned from user preferences (y-axis) with the true Atari reward (x-axis) averaged over 1000 timesteps. For a fully aligned reward function, all points would be on a straight line. In these experiments the reward model is well-aligned in some games like Beamrider, Hero, and Q*bert, and poorly aligned in others like Private Eye, Breakout, and Mondezuma's Revenge. Reproduced from Ibarz et al. (2018).

An ambitious goal is to enable the production of *safety certificates*, artifacts that serve as evidence to convince a third party to trust our system. These safety certificates could be used to prove responsible technology development, defuse competition, and demonstrate compliance with regulations. A safety certificate could take the form of a score on a secret test suite held by a third party, evidence of interpretability properties, or a machine-checkable formal proof of correctness with respect to some established specification, among others. A few general approaches for building trust in our models are discussed below.

Design choices. Separating learning the objective from learning the behavior allows us to achieve higher confidence in the resulting behavior because we can split trust in the reward model from trust in the policy. For example, we can measure how well the reward function aligns with the task objective by evaluating it on the user's feedback (see Figure 5). If we understand and trust the reward model, we know what the agent is 'trying' to accomplish. If Assumption 2 is true, then the reward model should be easier to interpret and debug than the policy.

Another design choice that could increase trust in the system is to split our policy into two parts: a *plan generator* and a *plan executor*. The plan generator produces a human-readable plan of the current course of action. This plan could be very high-level like a business plan or a research proposal, or fairly low-level like a cooking recipe. This plan can then optionally be reviewed and signed off by the user. The plan executor then takes the plan and implements it.

Clean, well-understood design choices on training setup, model architecture, loss function, and so on can lead to more predictable behavior and thus increase our overall confidence in the resulting system (as opposed to e.g. training a big blob of parameters end-to-end). Especially if we manage to formally specify certain safety properties (Orseau & Armstrong, 2016; Krakovna et al., 2018), we can then make them an explicit part of our agent design.

Testing. Evaluation on a separate held-out test set is already common practice in machine learning. For supervised learning, the performance of a trained model is estimated by the empirical risk on

a held-out test set which is drawn from the same data distribution. This practise can readily be applied to reward model (Ibarz et al., 2018) and policy, e.g. on a set of specifically designed simulated environments (Leike et al., 2017) or even adversarially where an attacker explicitly tries to cause misbehavior in the agent (Anonymous, 2019c).

Interpretability. Interpretability has been defined as the ability to explain or to present in understandable terms to a human (Doshi-Velez & Kim, 2017). Currently widely used deep neural networks are mostly black boxes, and understanding their internal functionality is considered very difficult. Nevertheless, recent progress provides reason for optimism that we will be able to make these black boxes increasingly transparent. This includes preliminary work on visualizing the latent state space of agents using t-SNE plots (Zahavy et al., 2016; Jaderberg et al., 2018), examining what agents attend to when they make decisions (Greydanus et al., 2018), evaluating models' sensitivity to the presence/intensity of high-level human concepts (Kim et al., 2017), optimizing a model to be more interpretable with humans in the loop (Lage et al., 2018), translating neural activations into natural language on tasks also performed by humans (Andreas et al., 2017), and combining different interactive visualization techniques (Olah et al., 2018), to name only a few.

Formal verification. Recent progress on model checking for neural networks opens the door for formal verification of trained models (Katz et al., 2017). The size of verified models has been pushed beyond MNIST-size to over a million parameters (Dvijotham et al., 2018b; Wong et al., 2018), which indicates that verifying practically sized RL models might soon be within reach. If formal verification can be scaled, we could attempt to verify properties of policies (Bastani et al., 2018) and reward functions with respect to a high-level specification, including off-switches, side-effects, and others mentioned in Section 3.1. If Assumption 1 from Section 1 is true, then this specification does not have to be manually written, but instead can be provided by a separately learned model. However, in this case a formal correctness proof is only as useful as this learned specification is accurate.

To make the verification task easier, our models could be trained to be more easily verifiable (Dvijotham et al., 2018a). However, this opens the door for degenerate solutions that exploit loopholes in the learned specification. This is analogous to problems with reward hacking (Section 4.3) which train a policy to optimize a frozen reward model (Figure 4). Circumventing this problem could be done using the same techniques that have been successful for reward hacking, such as learning the specification online using user feedback (Section 5.1).

Theoretical guarantees. Finally, even more ambitious would be the development of theoretically well-founded scalable learning algorithms that come with *probably approximately correct* (Dziugaite & Roy, 2017) or *sample complexity* guarantees, capacity statements, well-calibrated uncertainty estimates, etc. (Veness et al., 2017). Unfortunately, currently there is a dire lack of any such guarantees for the popular deep neural network architectures and training techniques.

7   Alternatives for agent alignment

The research direction we outline in this paper is not the only possible path to solve the agent alignment problem. While we believe it is currently the most promising one to explore, it is not guaranteed to succeed. Fortunately there are a number of other promising directions for agent alignment. These can be pursued in parallel or even combined with each other. This section provides an overview and explains how our approach relates to them. Our list is not exhaustive; more directions are likely to be proposed in the future.

7.1   Imitation learning

One strategy to train aligned agents could be from imitating human behavior (Pomerleau, 1991; Abbeel & Ng, 2004; Ho & Ermon, 2016; Finn et al., 2016). An agent imitating aligned human behavior sufficiently well should be aligned as well. The following caveats apply:

- *Amount of data.* While feedback can often be provided by non-experts, the data for human imitation has to be provided by experts on the task. This might be much more expensive data and it is not clear if we need more or less than for reward modeling.

- *Cognitive imitation.* It is possible that a lot of cognitively demanding tasks that humans do rely on very high-level intuition, planning, and other cognitive processes that are poorly reflected in human actions. For example, a crucial insight for solving a problem might be gained from drawing an analogy with a different problem encountered in a different domain. This might be hard to replicate and predict from data about human actions alone.

- *Generalization.* In order to be useful, our agent trained with imitation learning needs to showcase persistently high-quality behavior, even in the face of novel situations. Analogous to Assumption 2, generalizing learned reward functions might be easier than generalizing behavior (Bahdanau et al., 2018).

- *Performance.* It is generally difficult to outperform humans using imitation learning alone (Hester et al., 2018): even a perfect imitator can only perform as well as the source it is imitating; superhuman performance typically comes from executing human action sequences faster and more reliably by smoothing out inconsistencies in human behavior (Aytar et al., 2018).

Therefore imitation learning is unlikely to be competitive with other strategies to train agents in the longer term. However, it might be sufficient to act as a 'stepping stone': agents trained with imitation learning might act as 'research assistants' and help scale up other alignment efforts. Therefore it should be considered as a strong alternative to our research strategy.

7.2 Inverse reinforcement learning

We can view a reinforcement learning algorithm as a mapping from a reward function to behavior. The inverse of that mapping takes agent behavior as input and produces a reward function; this is known as *inverse reinforcement learning* (IRL; Russell, 1998; Ng & Russell, 2000). In this sense, inverse reinforcement learning can be viewed as one approach to reward modeling that takes feedback in the form of trajectories of behavior. However, taken as it is, it had two shortcomings:

1. IRL is an under-constrained problem because the reward function is not uniquely identifiable (not even up to affine-linear transformation) from behavior alone (Ng & Russell, 2000); for example, $R = 0$ is always a solution. If we assume the human is fully rational and the agent can design a sequence of tasks for the human, then the reward function can be identified (Amin et al., 2017). Even some assumptions about the human's rationality can be relaxed (Evans et al., 2016), but in full generality the inverse reinforcement learning problem becomes impossible to solve (Armstrong & Mindermann, 2018).

2. It assumes the human is acting to optimize their reward directly, even when this is an inefficient way of communicating their preferences. For instance, it is much easier for a human to state 'I would like you to make me coffee every morning at 8am' than it is for the human to make themselves coffee at 8am several days in a row.

7.3 Cooperative inverse reinforcement learning

Motivated by this second shortcoming of IRL, Hadfield-Menell et al. (2016) propose *cooperative inverse reinforcement learning* (CIRL). CIRL is a formal model of reward modeling as a two player game between a user and an agent which proceeds as follows.

1. The user and the agent begin with a shared prior over the user's reward function,
2. the user then observes their reward function, and finally
3. both user and agent execute policies to optimize the user's reward function.

An optimal solution to a CIRL game would use the common knowledge of the user and the agent to compute a policy for the agent (to be executed in step 3), and a mapping from reward function to policy for the user. Then upon observing their reward function in step 2, the user should select the corresponding policy for them to execute in step 3. Both the user and the agent have to choose behaviors which trade off between (1) communicating the user's reward function to the agent and (2) directly maximizing the user's expected reward.

We make two observations about CIRL as an approach to agent alignment that highlight that CIRL abstracts away from some important details. First, the performance of a CIRL algorithm will depend

on the quality of the prior over reward functions. In essence, CIRL replaces the problem of specifying a reward function with specifying a prior over reward functions. Second, computing the optimal solution to the CIRL problem is not realistic, since we cannot prescribe exactly how the user should interact with the agent. In other words, an efficient solution to a CIRL game might employ a strategy that transmits the parameters from the user to the agent, followed by a normal RL algorithm executed by both the user and the agent (since the reward is now fully observable to both). But if the user were able to observe their reward function, they could just specify this to an RL agent directly. In other words, one of the difficulties of agent alignment is that the reward function is not directly available to the user in the first place: users are usually not very aware of all of their preferences, and it might instead be easier for them to communicate through revealed preferences (Samuelson, 1938).

Nevertheless, CIRL incorporates two important insights into the alignment problem that also motivate our research direction:

1. Constructing agents to optimize a *latent* reward function can help align them on tasks where we cannot consistently provide reward feedback about all state-action pairs as the agent is visiting them.

2. A key challenge of the agent alignment problem is finding efficient ways to communicate the user's intentions to learning agents.

7.4 Myopic reinforcement learning

Myopic RL agents only maximize reward in the present timestep instead of a (discounted) sum of future rewards. This means that they are more short-sighted and thus not incentivized to execute long-term plans or take actions that are bad in the short-term in order to get a long-term benefit. In particular, myopic RL agents might be less prone to some of the design specification problems mentioned in Section 3.1, since causing them might take several time-steps to pay off for the agent.

There are two main myopic RL algorithms. TAMER (Knox & Stone, 2009; Knox, 2012; Warnell et al., 2017) is a collection of algorithms that learn a policy from human value feedback, i.e. take actions that maximize expected feedback in the next step (possibly with short temporal smoothing). COACH (MacGlashan et al., 2017; Arumugam et al., 2018) is an algorithm that trains a policy from feedback in the form of an *advantage function* (Sutton & Barto, 2018).

In contrast to imitation learning, the user does not have to be able to produce the desired behavior, just be able to reward the individual actions that bring it about. For example, using TAMER or COACH, a user could teach an agent to perform a backflip without being able to do one themselves. However, while myopic RL may increase alignment, is also comes with performance drawbacks. Training myopic RL agents puts the burden of solving the credit assignment problem onto the user, limiting the agent's potential for ingenuity and thus performance, and also leaving the user responsible for avoiding long-term negative consequences.

Despite these limits, myopic RL agents might be sufficient for some tasks where credit assignment is reasonably easy for humans. They might also be used as building blocks in more capable training regimes, for instance in iterated amplification (Christiano et al., 2018).

7.5 Imitating expert reasoning

Another alternative is to train a model to imitate expert reasoning. The imitation can happen at a level of granularity decided by the expert and could include 'internal' reasoning steps that the expert would not typically perform explicitly. This expert reasoning can then be improved and accelerated (Christiano et al., 2018; Evans et al., 2018; Stuhlmüller, 2018).

The basic idea is best illustrated with a question answering system. The input to the system is a question Q and its output an answer A. For simplicity we can treat both Q and A as natural language strings. The system can call itself recursively by asking subquestions $Q_1, \ldots, Q_k$, receiving their answers $A_1, \ldots, A_k$, and composing them into the answer A.

For example, consider the question Q 'How many pineapples are there in Denmark?' To give an approximate answer, we could make a *Fermi estimate* by asking the subquestions 'What is the population of Denmark?', 'How many pineapples does the average Dane consume per year?', and

'How long are pineapples stored?' These subquestions are then answered recursively and their answers can be composed into an answer to the original question Q.

We could train a model to answer questions Q recursively by using the same reasoning procedure as the expert using imitation learning (Section 7.1). This model can then be improved using a variety of methods:

- Running many copies of this model in parallel and/or at greater speed.
- Training a new model to predict answers to questions without having to expand the subquestions, akin to using a value network to the estimate the result of a tree search (Anthony et al., 2017; Silver et al., 2017).
- Making the expert reasoning more coherent under reflection. For example, by searching for inconsistencies in the expert's reasoning and resolving them.

If we believe expert reasoning is aligned with the user, then we could hope that the resulting improved model is also aligned. This training procedure aims to achieve better interpretability and greater trust in the resulting agents than recursive reward modeling (Section 3.2). However, learning expert reasoning might not be economically competitive with recursive reward modeling, depending on how good the expert's reasoning is and whether Assumption 2 holds for the task at hand.

Even though both are an instance of the more general framework of iterated amplification (Christiano et al., 2018), recursive reward modeling as described in Section 3.2 does not try to model expert reasoning explicitly. Instead, recursive reward modeling only requires users to evaluate outcomes. Nevertheless, it relies on decomposition of the evaluation task which has similarities to the decompositional reasoning described here. When using recursive reward modeling users have the *option* to provide feedback on the cognitive process that produced outcomes, but they are not required to do so. Moreover, this feedback might be difficult to provide in practice if the policy model is not very interpretable.

7.6 Debate

Irving et al. (2018) describe an idea for agent alignment that involves a two-player zero-sum game in which both players are debating a question for the user. The two players take turns to output a short statement up to a turn limit. At the end of the game the user reads the conversation transcript and declares the player who contributed the most true and useful statements the winner.

The debate proposal involves training an agent with self play (Silver et al., 2016) on this debate game. In order to become aligned, this agent needs to be trained in a way that it converges to a Nash equilibrium in which both instances of the agent try to be helpful to the user. The central assumption of debate is that it is easier for the agent to tell the truth than it is to lie. If this assumption holds, then the dynamics of the game should incentivize the agent to provide true and useful statements.

The authors provide initial experiments on the MNIST dataset in which the debating agents manage to boost the accuracy of a sparse classifier that only has access to a few of the image's pixels. While these initial experiments are promising, more research is needed in order to determine whether debate is a scalable alignment approach. We need more empirical evidence to clarify, among others, the following two questions.

1. Does the central assumption of debate hold outside domains of easily fact-checkable statements?
2. Can the humans accurately judge the debate even if the debaters have superior persuasion and deception ability?

7.7 Other related work

Many of the practical challenges to reward modeling we raise here have already been discussed by Amodei et al. (2016): safe exploration, distributional shift, side-effects, and reward hacking. In particular, the authors highlight what they call the scalable oversight problem, how to train an RL agent with sparse human feedback. This can be understood as a more narrow version of the alignment problem we are aiming to solve here. In a similar spirit, Taylor et al. (2016) survey a number of

high-level open research questions on agent alignment. Most closely related to our approach are what the authors call informed oversight (building systems that help explain outcomes), generalizable environmental goals (defining objective functions in terms of environment states), and averting instrumental incentives (preventing the system from optimizing for certain undesirable subgoals).

Soares & Fallenstein (2017) outline a research agenda of a very different flavor. Their research problems are quite paradigm-agnostic and instead concern the theoretical foundations of mathematical agent models. In particular, many of their problems aim to address perceived difficulties in applying current notions of optimal behavior to agents which are part of their environment (Orseau & Ring, 2012) and thus may not remain cleanly delineated from it (Demski & Garrabrant, 2018). The authors seek the formal tools to ask questions about or relevant to alignment in theory, such as when provided with a halting oracle (Hutter, 2005). These formal tools could be necessary for formal verification of agents designing upgraded versions of themselves. Yet while there has been some of progress on this research agenda (Barasz et al., 2014; Leike et al., 2016; Garrabrant et al., 2016; Everitt, 2018), some questions turned out to be quite difficult. But even if we had formal solutions to the problems put forth by Soares & Fallenstein, there would still persist a gap to transfer these solutions to align agents in practice. For now, answers to these research questions should be understood more as intuition pumps for practical alignment questions rather than direct solutions themselves (Garrabrant, 2018).

See Everitt et al. (2018) for more in-depth survey and literature review.

8   Discussion

Summary.   The version of the agent alignment problem we are aiming to solve involves aligning a single agent to a single user (Section 2). Instead of attempting to learn the entire preference payload, we outline an approach for enabling the user to communicate their intentions to the agent for the task at hand so that it allows them to trust the trained agent.

Our research direction for agent alignment is based on scaling reward modeling (Section 3). This direction fits well into existing efforts in machine learning because it can benefit from advances in the state of the art in supervised learning (for the reward model) and reinforcement learning (for the policy). Building on previous work (Section 7), we provide significantly more detail, including the main challenges (Section 4) and concrete approaches to mitigate these challenges (Section 5) and to establish trust in the agents we train (Section 6). In essence, this document combines existing efforts on AI safety problems by providing one coherent narrative around how solving these problems could enable us to train aligned agents beyond human-level performance.

Concrete research projects.   Our research direction is 'shovel-ready' for empirical research today. We can set up experiments with deep reinforcement learning agents: getting empirical data on the severity of the challenges from Section 4; prototyping solution ideas from Section 5; scaling reward modeling to more difficult tasks; pushing the frontiers on (adversarial) testing, interpretability, formal verification, and the theory of deep RL. Moreover, we can readily use any existing RL benchmarks such as games or simulated environments that come with pre-programmed reward functions: By hiding this reward function from the algorithm we can pretend it is unavailable, but still use it for synthetically generated user feedback (Christiano et al., 2017) as well as the evaluation of the learned reward model (Ibarz et al., 2018).

Outlook.   There is enormous potential for ML to have a positive impact on the real world and improve human lives. Since most real-world problems can be cast in the RL framework, deep RL is a particularly promising technique for solving real-world problems. However, in order to unlock its potential, we need to train agents in the absence of well-specified reward functions. Just as proactive research into robustness of computer vision systems is essential for addressing vulnerabilities to adversarial inputs, so could alignment research be key to getting ahead of future bottlenecks to the deployment of ML systems in complex real-world domains. For now, agent alignment research is still in its early stages, but we believe that there is substantial reason for optimism. While we expect to face challenges when scaling reward modeling, these challenges are concrete technical problems that we can make progress on with targeted research.

References

Pieter Abbeel and Andrew Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 1–8, 2004.

David Abel, John Salvatier, Andreas Stuhlmüller, and Owain Evans. Agent-agnostic human-in-the-loop reinforcement learning. *arXiv preprint arXiv:1701.04079*, 2017.

Bruce Abramson. *The Expected-Outcome Model of Two-Player Games*. PhD thesis, Columbia University, 1987.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31, 2017.

Riad Akrour, Marc Schoenauer, and Michèle Sebag. APRIL: Active preference learning-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 116–131, 2012.

Riad Akrour, Marc Schoenauer, Michèle Sebag, and Jean-Christophe Souplet. Programming by feedback. In *International Conference on Machine Learning*, pp. 1503–1511, 2014.

Kareem Amin, Nan Jiang, and Satinder Singh. Repeated inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1815–1824, 2017.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Jacob Andreas, Anca Dragan, and Dan Klein. Translating neuralese. In *Association for Computational Linguistics*, pp. 232–242, 2017.

Jacob Andreas, Dan Klein, and Sergey Levine. Learning with latent language. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2166–2179, 2018.

Anonymous. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=r1lyTjAqYX. Under review.

Anonymous. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=HkezXnA9YX. Under review.

Anonymous. Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures. In *International Conference on Learning Representations*, 2019c. URL https://openreview.net/forum?id=B1xhQhRcK7. Under review.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, pp. 5360–5370, 2017.

Arbital. Optimization daemons. https://arbital.com/p/daemons/, 2016.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

Stuart Armstrong. Motivated value selection for artificial agents. In *AAAI Workshop: AI and Ethics*, 2015.

Stuart Armstrong and Benjamin Levinstein. Low impact artificial intelligences. *arXiv preprint arXiv:1705.10720*, 2017.

Stuart Armstrong and Sören Mindermann. Occam's razor is insufficient to infer the preferences of irrational agents. In *Advances in Neural Information Processing Systems*, 2018.

Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L Littman. Deep reinforcement learning from policy-dependent human feedback. Technical report, Brown University, 2018.

Isaac Asimov. Runaround. *Astounding Science Fiction*, 29(1):94–103, 1942.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching YouTube. In *Advances in Neural Information Processing Systems*, 2018.

James Babcock, János Kramár, and Roman Yampolskiy. The AGI containment problem. In *Artificial General Intelligence*, pp. 53–63, 2016.

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. Learning to follow language instructions with adversarial reward induction. *arXiv preprint arXiv:1806.01946*, 2018.

Mihaly Barasz, Paul Christiano, Benja Fallenstein, Marcello Herreshoff, Patrick LaVictoire, and Eliezer Yudkowsky. Robust cooperation in the prisoner's dilemma: Program equilibrium via provability logic. *arXiv preprint arXiv:1401.5577*, 2014.

Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations*, 2018.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. *arXiv preprint arXiv:1805.08328*, 2018.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Seth D Baum. Social choice ethics in artificial intelligence. *AI & Society*, pp. 1–12, 2017.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on International Conference on Machine Learning*, pp. 1613–1622, 2015.

Nick Bostrom. Ethical issues in advanced artificial intelligence. *Science Fiction and Philosophy: From Time Travel to Superintelligence*, pp. 277–284, 2003.

Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.

Tom B Brown, Nicholas Carlini, Chiyuan Zhang, Catherine Olsson, Paul Christiano, and Ian Goodfellow. Unrestricted adversarial examples. *arXiv preprint arXiv:1809.08352*, 2018.

Jonathon Cai, Richard Shin, and Dawn Song. Making neural programming architectures generalize via recursion. In *International Conference on Learning Representations*, 2017.

J Quiñonero Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. Dataset shift in machine learning, 2009.

Paul Christiano. Directions and desiderata for AI alignment. `https://ai-alignment.com/directions-and-desiderata-for-ai-control-b60fca0da8f4`, 2017.

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

Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018.

Jack Clark and Dario Amodei. Faulty reward functions in the wild. `https://blog.openai.com/faulty-reward-functions/`, 2016.

Peter Dayan and Geoffrey Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 271–278, 1993.

Abram Demski and Scott Garrabrant. Embedded agency. `https://intelligence.org/embedded-agency/`, 2018.

Daniel Dewey. Learning what to value. In *Artificial General Intelligence*, pp. 309–314, 2011.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*, 2018a.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *Uncertainty in Artificial Intelligence*, 2018b.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Uncertainty in Artificial Intelligence*, 2017.

Layla El Asri, Bilal Piot, Matthieu Geist, Romain Laroche, and Olivier Pietquin. Score-based inverse reinforcement learning. In *International Conference on Autonomous Agents & Multiagent Systems*, pp. 457–465, 2016.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Oren Etzioni and Daniel Weld. The first law of robotics (a call to arms). In *AAAI*, pp. 1042–1047, 1994.

Owain Evans, Andreas Stuhlmüller, and Noah D Goodman. Learning the preferences of ignorant, inconsistent agents. In *AAAI*, pp. 323–329, 2016.

Owain Evans, Andreas Stuhlmüller, Chris Cundy, Ryan Carey, Zachary Kenton, Thomas McGrath, and Andrew Schreiber. Predicting human deliberative judgments with machine learning. Technical report, University of Oxford, 2018.

Tom Everitt. *Towards Safe Artificial General Intelligence*. PhD thesis, Australian National University, 2018.

Tom Everitt and Marcus Hutter. The alignment problem for Bayesian history-based reinforcement learners. *Under submission*, 2018.

Tom Everitt, Victoria Krakovna, Laurent Orseau, and Shane Legg. Reinforcement learning with a corrupted reward channel. In *International Joint Conference on Artificial Intelligence*, pp. 4705–4713, 2017.

Tom Everitt, Gary Lea, and Marcus Hutter. AGI safety literature review. *arXiv preprint arXiv:1805.01109*, 2018.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016.

Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning*, 89(1-2):123–156, 2012.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pp. 3581–3590, 2017a.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *International Conference on Machine Learning*, pp. 1183–1192, 2017b.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Scott Garrabrant. Sources of intuitions and data on AGI. `https://www.lesswrong.com/posts/BibDWWeo37pzuZCmL/sources-of-intuitions-and-data-on-agi`, 2018.

Scott Garrabrant, Tsvi Benson-Tilsen, Andrew Critch, Nate Soares, and Jessica Taylor. Logical induction. *arXiv preprint arXiv:1609.03543*, 2016.

Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Artificial Intelligence and Statistics*, pp. 315–323, 2011.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6572`.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT Press, 2016.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International Conference on Machine Learning*, 2013.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

Sam Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding Atari agents. In *International Conference on Machine Learning*, pp. 1792–1801, 2018.

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

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330, 2017.

Dylan Hadfield-Menell, Stuart Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3909–3917, 2016.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. The off-switch game. In *International Joint Conference on Artificial Intelligence*, pp. 220–227, 2017a.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems*, pp. 6765–6774, 2017b.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, John Agapiou, Joel Z Leibo, and Audrunas Gruslys. Deep Q-learning from demonstrations. In *AAAI*, 2018.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

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

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representaitons*, 2018.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Marcus Hutter. *Universal artificial intelligence*. Springer, 2005.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari. In *Advances in Neural Information Processing Systems*, 2018.

Alex Irpan. Deep reinforcement learning doesn't work yet. `https://www.alexirpan.com/2018/02/14/rl-hard.html`, 2018.

Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

Richard Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85–103. Springer, 1972.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification*, pp. 97–117, 2017.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pp. 5574–5584, 2017.

Been Kim, Justin Gilmer, Fernanda Viegas, Ulfar Erlingsson, and Martin Wattenberg. TCAV: Relative concept importance testing with linear concept activation vectors. *arXiv preprint arXiv:1711.11279*, 2017.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

William Bradley Knox. *Learning from human-generated reward*. PhD thesis, University of Texas, 2012.

William Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The TAMER framework. In *International Conference on Knowledge Capture*, pp. 9–16, 2009.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pp. 282–293, 2006.

Victoria Krakovna, Laurent Orseau, Miljan Martic, and Shane Legg. Measuring and avoiding side effects using relative reachability. *arXiv preprint arXiv:1806.01186*, 2018.

Julia Kreutzer, Joshua Uyheng, and Stefan Riezler. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. *arXiv preprint arXiv:1805.10627*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

David Krueger, Jan Leike, Owain Evans, and John Salvatier. Active reinforcement learning: Observing rewards at a cost. In *Future of Interactive Learning Machines, NIPS Workshop*, 2016.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pp. 2801–2809, 2018.

Isaac Lage, Andrew Slavin Ross, Been Kim, Samuel J Gershman, and Finale Doshi-Velez. Human-in-the-loop interpretability prior. *arXiv preprint arXiv:1805.11571*, 2018.

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

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pp. 396–404, 1990.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17(4):391–444, 2007.

Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Julie Beaulieu, Peter J Bentley, Samuel Bernard, Guillaume Belson, David M Bryson, Nick Cheney, Antoine Cully, Stephane Doncieux, Fred C Dyer, Kai Olav Ellefsen, Robert Feldt, Stephan Fischer, Stephanie Forrest, Antoine Frénoy, Christian Gagné, Leni Le Goff, Laura M Grabowski, Babak Hodjat, Frank Hutter, Laurent Keller, Carole Knibbe, Peter Krcah, Richard E Lenski, Hod Lipson, Robert MacCurdy, Carlos Maestre, Risto Miikkulainen, Sara Mitri, David E Moriarty, Jean-Baptiste Mouret, Anh Nguyen, Charles Ofria, Marc Parizeau, David Parsons, Robert T Pennock, William F Punch, Thomas S Ray, Marc Schoenauer, Eric Shulte, Karl Sims, Kenneth O Stanley, François Taddei, Danesh Tarapore, Simon Thibault, Westley Weimer, Richard Watson, and Jason Yosinski. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv preprint arXiv:1803.03453*, 2018.

Jan Leike and Marcus Hutter. Bad universal priors and notions of optimality. In *Conference on Learning Theory*, pp. 1244–1259, 2015.

Jan Leike, Jessica Taylor, and Benya Fallenstein. A formal solution to the grain of truth problem. In *Conference on Uncertainty in Artificial Intelligence*, pp. 427–436, 2016.

Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. AI safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Gary Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

Masterjun. SNES Super Mario World (USA) "arbitrary code execution" in 02:25.19. `http://tasvideos.org/2513M.html`, 2014.

Cade Metz. In two moves, AlphaGo and Lee Sedol redefined the future. `https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/`, 2016.

Smitha Milli, Dylan Hadfield-Menell, Anca Dragan, and Stuart Russell. Should robots be obedient? In *International Joint Conference on Artificial Intelligence*, pp. 4754–4760, 2017.

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, 2015.

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*, pp. 1928–1937, 2016.

Brendan Mulvaney. Red teams. *Marine Corps Gazette*, 96(7):63, 2012.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814, 2010.

Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 663–670, 2000.

Andrew Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, pp. 278–287, 1999.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. `https://distill.pub/2018/building-blocks`.

Stephen Omohundro. The basic AI drives. In *Artificial General Intelligence*, pp. 483–492, 2008.

Laurent Orseau. Asymptotic non-learnability of universal agents with computable horizon functions. *Theoretical Computer Science*, 473:149–156, 2013.

Laurent Orseau and Stuart Armstrong. Safely interruptible agents. In *Conference on Uncertainty in Artificial Intelligence*, pp. 557–566, 2016.

Laurent Orseau and Mark Ring. Space-time embedded intelligence. In *Artificial General Intelligence*, pp. 209–218, 2012.

Pedro A Ortega, Vishal Maini, and the DeepMind safety team. Building safe artificial intelligence: specification, robustness, and assurance. `https://medium.com/@deepmindsafetyresearch/building-safe-artificial-intelligence-52f5f75058f1`, 2018.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Association for Computational Linguistics*, pp. 311–318, 2002.

Philip Peter. Watching all the movies ever made. `http://www.justgeek.de/watching-all-the-movies-ever-made/`, 2014.

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

Mahendra Prasad. Social choice and the value alignment problem. In *Artificial Intelligence Safety and Security*, pp. 291–314. Chapman and Hall/CRC, 2018.

Mark O Riedl and Brent Harrison. Using stories to teach human values to artificial agents. In *AAAI Workshop: AI, Ethics, and Society*, 2016.

Mark Ring and Laurent Orseau. Delusion, survival, and intelligent agents. In *Artificial General Intelligence*, pp. 11–20, 2011.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, 1986.

Stuart Russell. Learning agents for uncertain environments. In *Conference on Computational Learning Theory*, pp. 101–103, 1998.

Stuart Russell, Daniel Dewey, and Max Tegmark. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4):105–114, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Paul A Samuelson. A note on the pure theory of consumer's behaviour. *Economica*, 5(17):61–71, 1938.

William Saunders, Girish Sastry, Andreas Stuhlmueller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention. In *International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2067–2069, 2018.

Jürgen Schmidhuber. Gödel machines: Self-referential universal problem solvers making provably optimal self-improvements. In *Artificial General Intelligence*, 2007.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sebastian Schulze and Owain Evans. Active reinforcement learning with monte-carlo tree search. *arXiv preprint arXiv:1803.04926*, 2018.

Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

Alireza Shafaei, Mark Schmidt, and James J Little. Does your model know the digit 6 is not a cat? a less biased evaluation of "outlier" detectors. *arXiv preprint arXiv:1809.04729*, 2018.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Graepel Thore, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

Nate Soares. The value learning problem. Technical report, Machine Intelligence Research Institute, 2015.

Nate Soares and Benya Fallenstein. Agent foundations for aligning machine intelligence with human interests: a technical research agenda. In *The Technological Singularity*, pp. 103–125. Springer, 2017.

Nate Soares, Benja Fallenstein, Stuart Armstrong, and Eliezer Yudkowsky. Corrigibility. In *AAAI Workshop on Artificial Intelligence and Ethics*, 2015.

Thilo Stadelmann, Mohammadreza Amirian, Ismail Arabaci, Marek Arnold, Gilbert François Duivesteijn, Ismail Elezi, Melanie Geiger, Stefan Lörwald, Benjamin Bruno Meier, Katharina Rombach, and Lukas Tuggener. Deep learning in the wild. *arXiv preprint arXiv:1807.04950*, 2018.

Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. In *International Conference on Learning Representations*, 2017.

Andreas Stuhlmüller. Factored cognition. `https://ought.org/projects/factored-cognition`, 2018.

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT press, 2nd edition, 2018.

Richard Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Jessica Taylor, Eliezer Yudkowsky, Patrick LaVictoire, and Andrew Critch. Alignment for advanced machine learning systems. Technical report, Machine Intelligence Research Institute, 2016.

Andrea L Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6–7):716–737, 2008.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems*, pp. 23–30, 2017.

Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, 2018.

Hsiao-Yu Fish Tung, Adam W Harley, Liang-Kang Huang, and Katerina Fragkiadaki. Reward learning from narrated demonstrations. In *Computer Vision and Pattern Recognition*, pp. 7004–7013, 2018.

Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning*, pp. 5025–5034, 2018.

Joel Veness, Tor Lattimore, Avishkar Bhoopchand, Agnieszka Grabska-Barwinska, Christopher Mattern, and Peter Toth. Online learning with gated linear networks. *arXiv preprint arXiv:1712.01897*, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal Networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep TAMER: Interactive agent shaping in high-dimensional state spaces. *arXiv preprint arXiv:1709.10163*, 2017.

Christopher Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

Norbert Wiener. Some moral and technical consequences of automation. *Science*, 1960.

Ronald Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

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

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, 2018.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Roman Yampolskiy. Leakproofing the singularity: Artificial intelligence confinement problem. *Journal of Consciousness Studies*, 2012.

Eric Yeh, Melinda Gervasio, Daniel Sanchez, Matthew Crossley, and Karen Myers. Bridging the gap: Converting human advice into imagined examples. *Advances in Cognitive Systems*, 6:1–20, 2018.

Eliezer Yudkowsky. Coherent extrapolated volition. Technical report, Singularity Institute for Artificial Intelligence, 2004.

Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding DQNs. In *International Conference on Machine Learning*, pp. 1899–1908, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018a.

Shun Zhang, Edmund H Durfee, and Satinder Singh. Minimax-regret querying on side effects for safe optimality in factored Markov decision processes. In *International Joint Conference on Artificial Intelligence*, pp. 4867–4873, 2018b.