Research

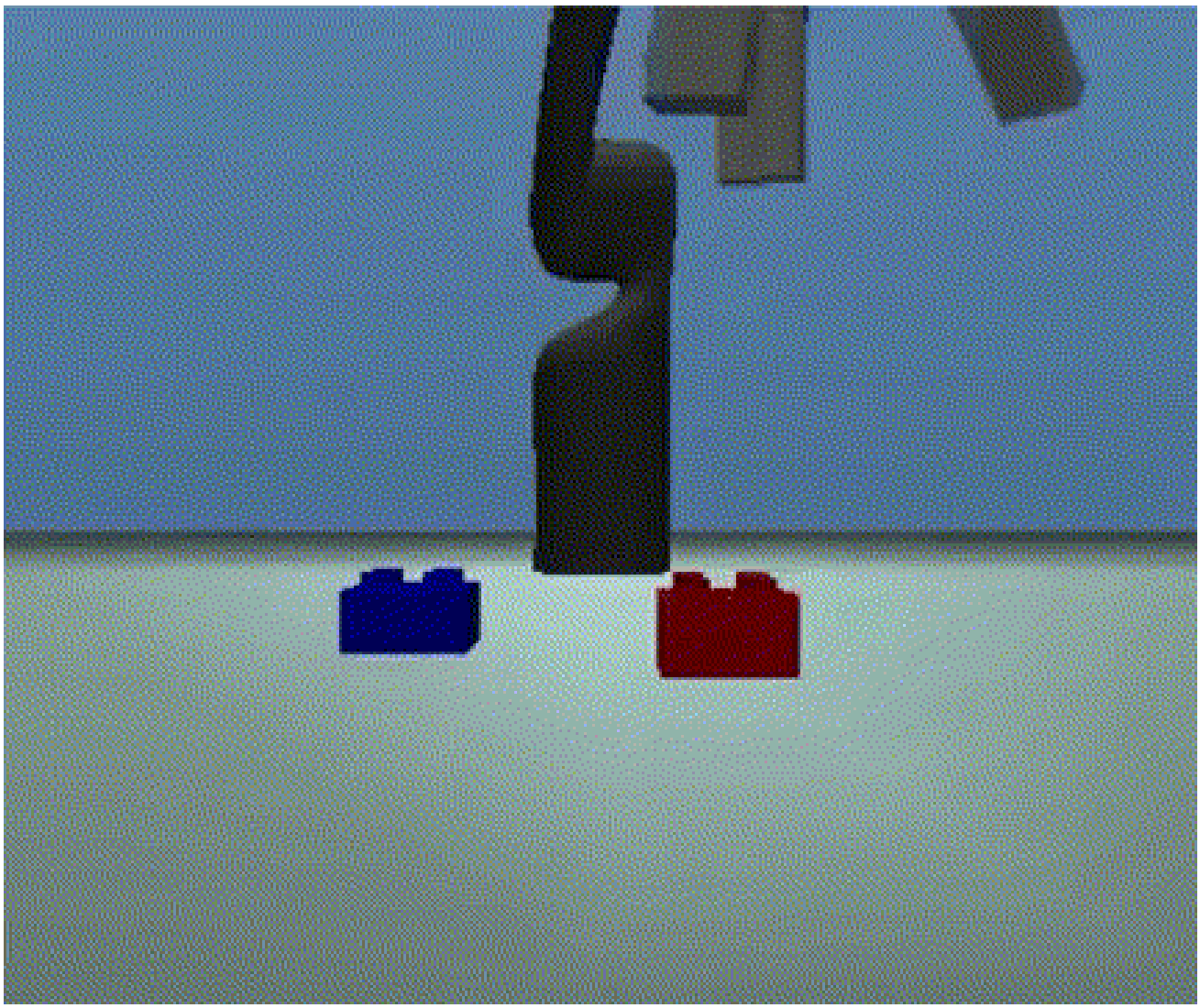
Specification gaming: the flip side of AI ingenuity

April 21, 2020

Specification gaming is a behaviour that satisfies the literal specification of an objective without achieving the intended outcome. We have all had experiences with specification gaming, even if not by this name. Readers may have heard the myth of King Midas and the golden touch, in which the king asks that anything he touches be turned to gold - but soon finds that even food and drink turn to metal in his hands. In the real world, when rewarded for doing well on a homework assignment, a student might copy another student to get the right answers, rather than learning the material - and thus exploit a loophole in the task specification.

This problem also arises in the design of artificial agents. For example, a reinforcement learning agent can find a shortcut to getting lots of reward without completing the task as intended by the human designer. These behaviours are common, and we have collected around 60 examples so far (aggregating existing lists and ongoing contributions from the AI community). In this post, we review possible causes for specification gaming, share examples of where this happens in practice, and argue for further work on principled approaches to overcoming specification problems.

Let's look at an example. In a Lego stacking task, the desired outcome was for a red block to end up on top of a blue block. The agent was rewarded for the height of the bottom face of the red block when it is not touching the block. Instead of performing the relatively difficult maneuver of picking up the red block and placing it on top of the blue one, the agent simply flipped over the red block to collect the reward. This behaviour achieved the stated objective (high bottom face of the red block) at the expense of what the designer actually cares about (stacking it on top of the blue one).

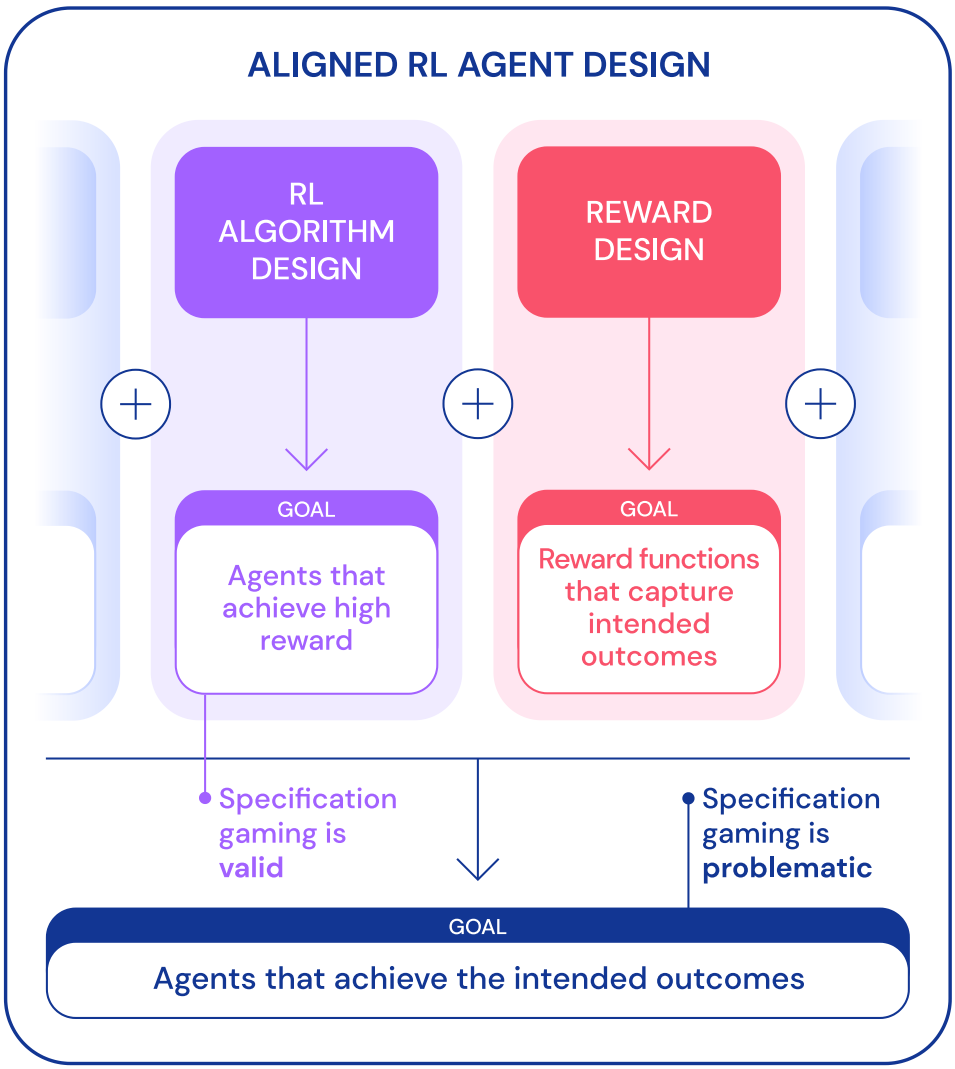


Source: Data-Efficient Deep Reinforcement Learning for Dexterous Manipulation (Popov et al, 2017)

We can consider specification gaming from two different perspectives. Within the scope of developing reinforcement learning (RL) algorithms, the goal is to build agents that learn to achieve the given objective. For example, when we use Atari games as a benchmark for training RL algorithms, the goal is to evaluate whether our algorithms can solve difficult tasks. Whether or not the agent solves the task by exploiting a loophole is unimportant in this context. From this perspective,

specification gaming is a good sign - the agent has found a novel way to achieve the specified objective. These behaviours demonstrate the ingenuity and power of algorithms to find ways to do exactly what we tell them to do.

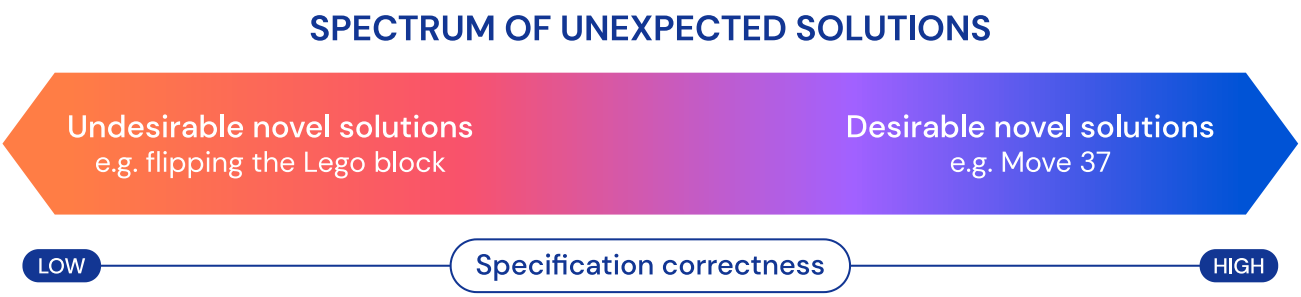
However, when we want an agent to actually stack Lego blocks, the same ingenuity can pose an issue. Within the broader scope of building aligned agents that achieve the intended outcome in the world, specification gaming is problematic, as it involves the agent exploiting a loophole in the specification at the expense of the intended outcome. These behaviours are caused by misspecification of the intended task, rather than any flaw in the RL algorithm. In addition to algorithm design, another necessary component of building aligned agents is reward design.



Designing task specifications (reward functions, environments, etc.) that accurately reflect the intent of the human designer tends to be difficult. Even for a slight misspecification, a very good RL algorithm might be able to find an intricate solution that is quite different from the intended solution, even if a poorer algorithm would not be able to find this solution and thus yield solutions that are closer to the

intended outcome. This means that correctly specifying intent can become more important for achieving the desired outcome as RL algorithms improve. It will therefore be essential that the ability of researchers to correctly specify tasks keeps up with the ability of agents to find novel solutions.

We use the term **task specification** in a broad sense to encompass many aspects of the agent development process. In an RL setup, task specification includes not only reward design, but also the choice of training environment and auxiliary rewards. The correctness of the task specification can determine whether the ingenuity of the agent is or is not in line with the intended outcome. If the specification is right, the agent's creativity produces a desirable novel solution. This is what allowed AlphaGo to play the famous Move 37, which took human Go experts by surprise yet which was pivotal in its second match with Lee Sedol. If the specification is wrong, it can produce undesirable gaming behaviour, like flipping the block. These types of solutions lie on a spectrum, and we don't have an objective way to distinguish between them.

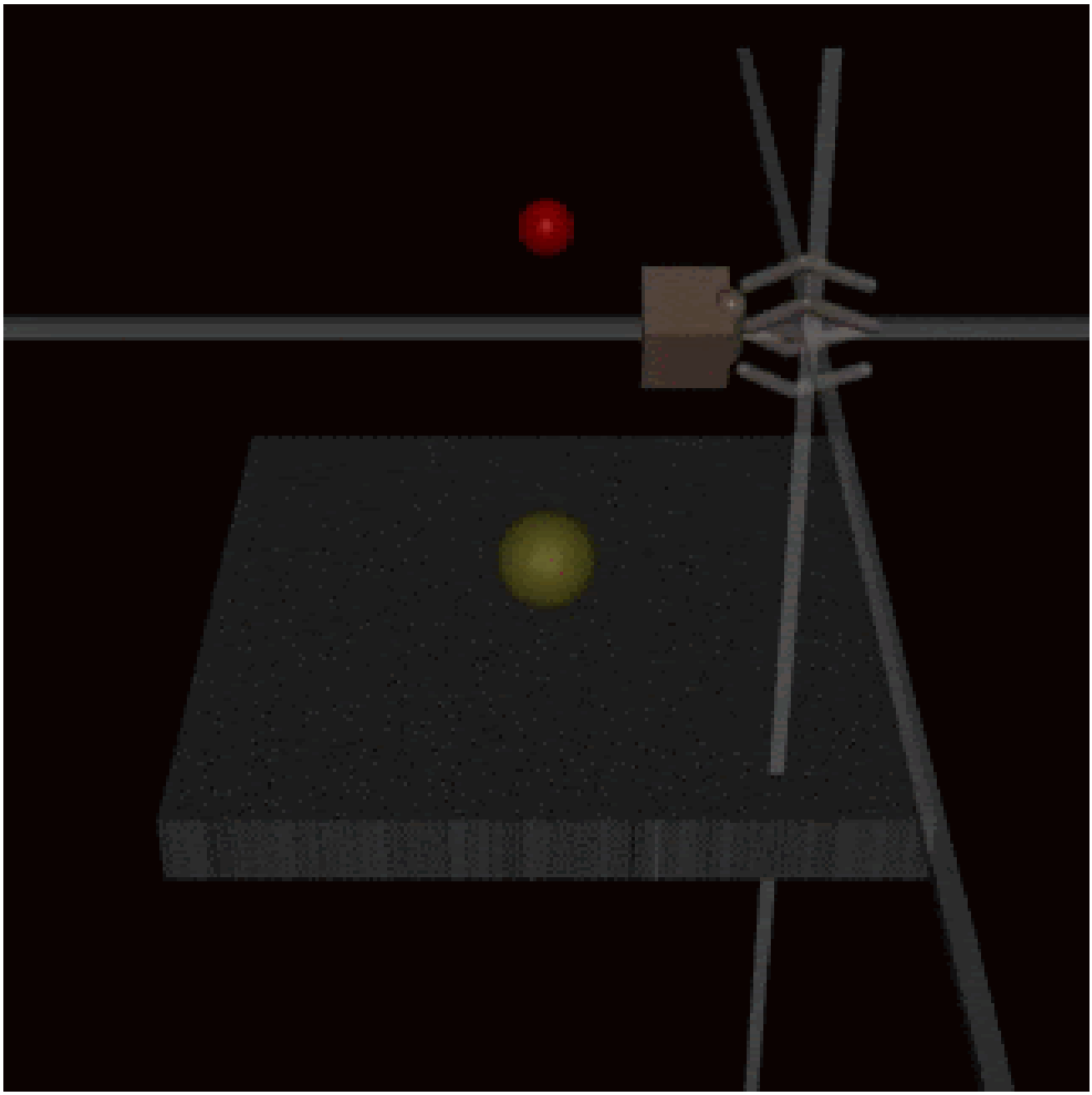


We will now consider possible causes of specification gaming. One source of reward function misspecification is poorly designed **reward shaping**. Reward shaping makes it easier to learn some objectives by giving the agent some rewards on the way to solving a task, instead of only rewarding the final outcome. However, shaping rewards can change the optimal policy if they are not potential-based. Consider an agent controlling a boat in the Coast Runners game, where the intended goal was to finish the boat race as quickly as possible. The agent was given a shaping reward for hitting green blocks along the race track, which changed the optimal policy to going in circles and hitting the same green blocks over and over again.

Source: Faulty Reward Functions in the Wild (Amodei & Clark, 2016)

Specifying a reward that accurately captures the **desired final outcome** can be challenging in its own right. In the Lego stacking task, it is not sufficient to specify that the bottom face of the red block has to be high off the floor, since the agent can simply flip the red block to achieve this goal. A more comprehensive specification of the desired outcome would also include that the top face of the red block has to be above the bottom face, and that the bottom face is aligned with the top face of the blue block. It is easy to miss one of these criteria when specifying the outcome, thus making the specification too broad and potentially easier to satisfy with a degenerate solution.

Instead of trying to create a specification that covers every possible corner case, we could **learn the reward function from human feedback**. It is often easier to evaluate whether an outcome has been achieved than to specify it explicitly. However, this approach can also encounter specification gaming issues if the reward model does not learn the true reward function that reflects the designer's preferences. One possible source of inaccuracies can be the human feedback used to train the reward model. For example, an agent performing a grasping task learned to fool the human evaluator by hovering between the camera and the object.



Source: Deep Reinforcement Learning From Human Preferences (Christiano et al, 2017)

The learned reward model could also be misspecified for other reasons, such as poor generalisation. Additional feedback can be used to correct the agent's attempts to exploit the inaccuracies in the reward model.

Another class of specification gaming examples comes from the agent exploiting **simulator bugs**. For example, a simulated robot that was supposed to learn to walk figured out how to hook its legs together and slide along the ground.

Source: AI Learns to Walk (Code Bullet, 2019)

At first sight, these kinds of examples may seem amusing but less interesting, and irrelevant to deploying agents in the real world, where there are no simulator bugs. However, the underlying problem isn't the bug itself but a failure of abstraction that can be exploited by the agent. In the example above, the robot's task was misspecified because of incorrect assumptions about simulator physics. Analogously, a real-world traffic optimisation task might be misspecified by incorrectly assuming that the traffic routing infrastructure does not have software bugs or security vulnerabilities that a sufficiently clever agent could discover. Such assumptions need not be made explicitly – more likely, they are details that simply never occurred to the designer. And, as tasks grow too complex to consider every detail, researchers are more likely to introduce incorrect assumptions during specification design. This poses the question: is it possible to design agent architectures that correct for such false assumptions instead of gaming them?

One assumption commonly made in task specification is that the task specification cannot be affected by the agent's actions. This is true for an agent running in a sandboxed simulator, but not for an agent acting in the real world. Any task specification has a physical manifestation: a reward function stored on a computer, or preferences stored in the head of a human. An agent deployed in the real world can potentially manipulate these representations of the objective, creating a reward tampering problem. For our hypothetical traffic optimisation system, there is no clear distinction between satisfying the user's preferences (e.g. by giving useful directions), and influencing users to have preferences that are easier to satisfy (e.g. by nudging them to choose destinations that are easier to reach). The former satisfies the objective, while the latter manipulates the representation of the objective in the world (the user preferences), and both result in high reward for the AI system. As another, more extreme example, a very advanced AI system could hijack the computer on which it runs, manually setting its reward signal to a high value.

To sum up, there are at least three challenges to overcome in solving specification gaming:

- How do we faithfully capture the human concept of a given task in a reward function?
- How do we avoid making mistakes in our implicit assumptions about the domain, or design agents that correct mistaken assumptions instead of gaming them?
- How do we avoid reward tampering?

While many approaches have been proposed, ranging from reward modeling to agent incentive design, specification gaming is far from solved. The list of specification gaming behaviours demonstrates the magnitude of the problem and the sheer number of ways the agent can game an objective specification. These problems are likely to become more challenging in the future, as AI systems become more capable at satisfying the task specification at the expense of the intended outcome. As we build more advanced agents, we will need design principles aimed specifically at overcoming specification problems and ensuring that these agents robustly pursue the outcomes intended by the designers.

Notes

We would like to thank Hado van Hasselt and Csaba Szepesvari for their feedback on this post.

Custom figures by Paulo Estriga, Aleks Polozuns, and Adam Cain.

Authors

Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, Shane Legg

Published

April 21, 2020

Tags

Research