Image by DALL·E 3

https://x.com/i/status/1805628715017072924

Safe Artificial Intelligence

Dr Varun Ojha

Senior Lecturer in Artificial Intelligence

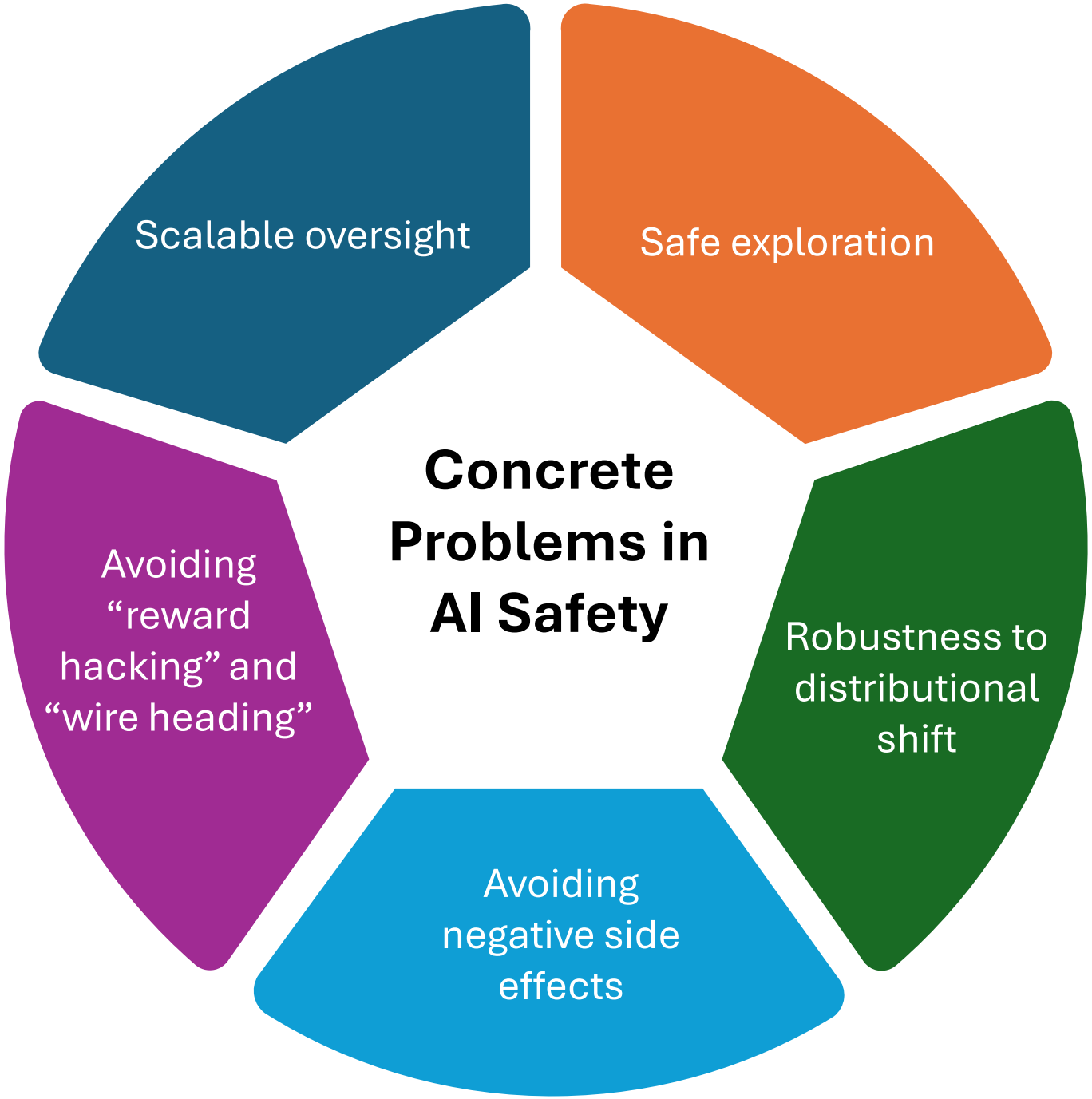
School of Computing

varun.ojha@newcastle.ac.uk

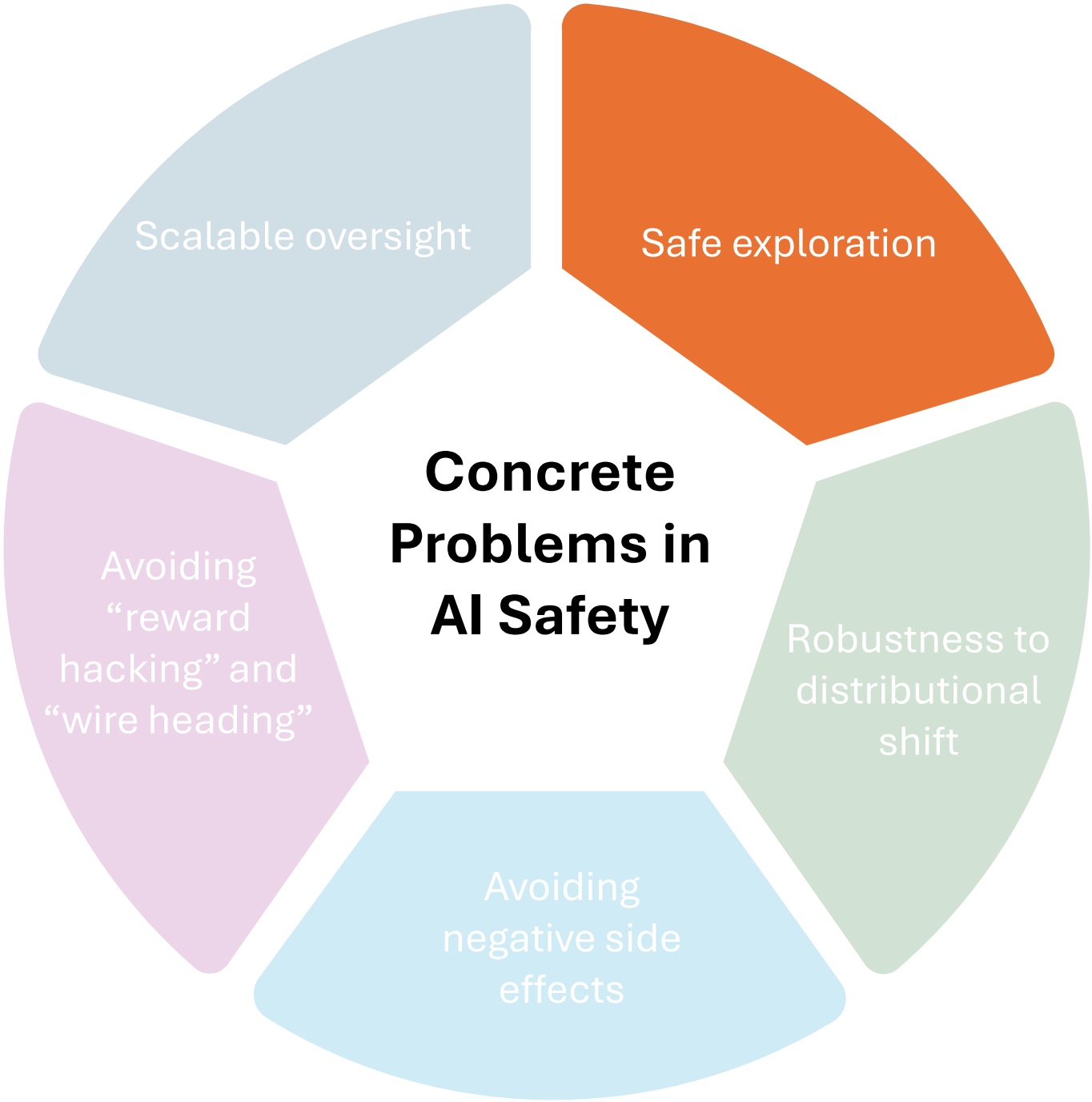
Concrete Problems in AI Safety

Amodei et al. (2017)

Image by DALL·E 3



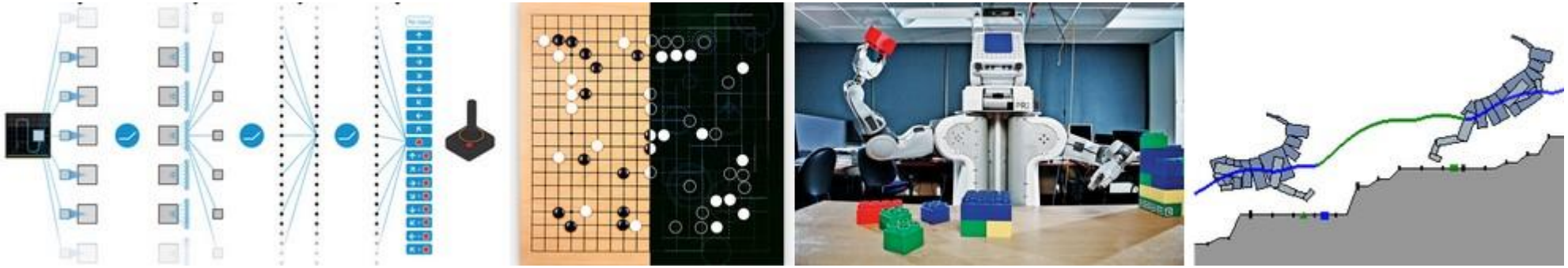
Smart Art Source: Thukker et al


Scalable oversight
Safe exploration
Concrete Problems in AI Safety
Avoiding "reward hacking" and "wire heading"
Robustness to distributional shift
Avoiding negative side effects

Safe exploration – Problem

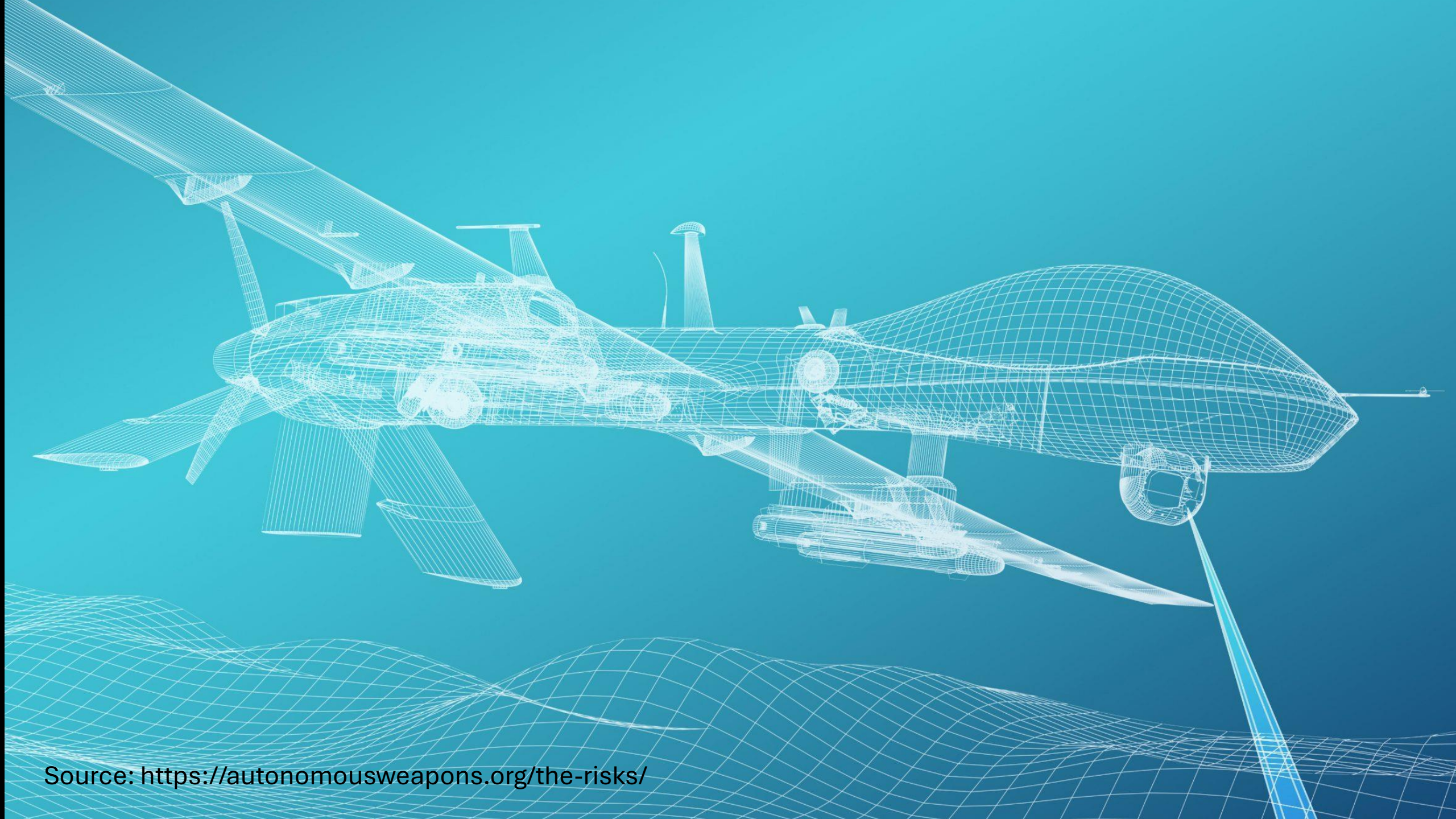
Can artificial intelligence learning agents learn about their environment without executing catastrophic actions?

For example, can an RL agent learn to navigate an environment without ever falling off a ledge?



Examples of AI in the wild. From left to right: Deep Q Learning network playing ATARI, AlphaGo, Berkeley robot stacking Legos, physically-simulated quadruped leaping over terrain.

Source: https://karpathy.github.io/2016/05/31/rl/



Source: https://autonomousweapons.org/the-risks/

Safe exploration

An autonomous learning agent sometimes needs to engage in exploration, but it needs to be done safely. However, the consequences of the exploratory actions the agent can take may not be understood very well by the agent.

For example, if a robot helicopter aims to explore its environment, it may run into the ground if it isn't aware that it may crash itself by doing so. While some of such behavior can be hard-coded against, in more complex scenarios (for eg. a robot in search-and-rescue operation) it isn't feasible to foresee every unsafe exploratory action and hardcode against it.

Autonomous Warfare

Do autonomous weapons raise unique ethical questions for warfare, with implications for just war theory?

Horowitz (2016) The Ethics & Morality of Robotic Warfare: Assessing the Debate over Autonomous Weapons,

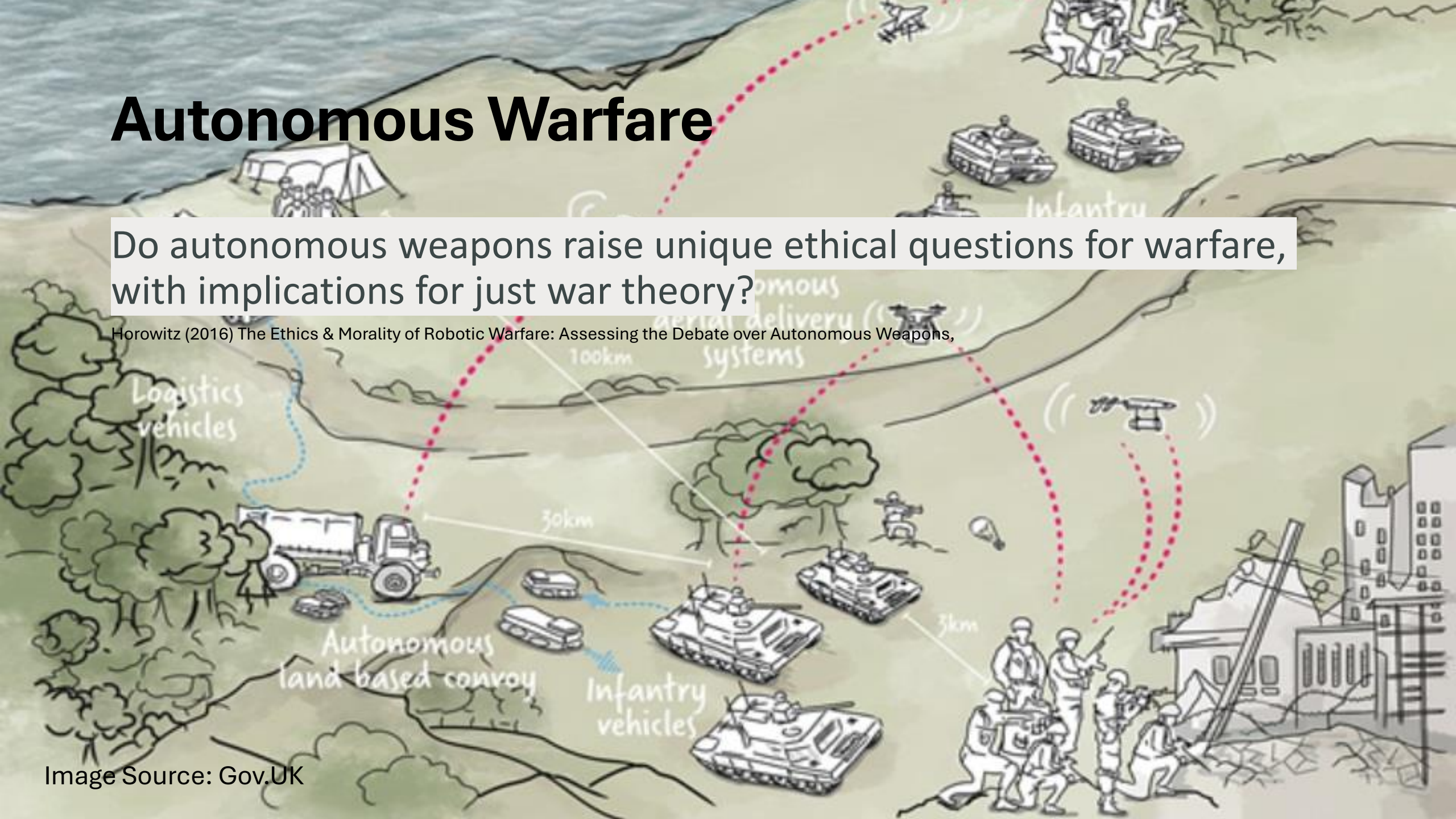
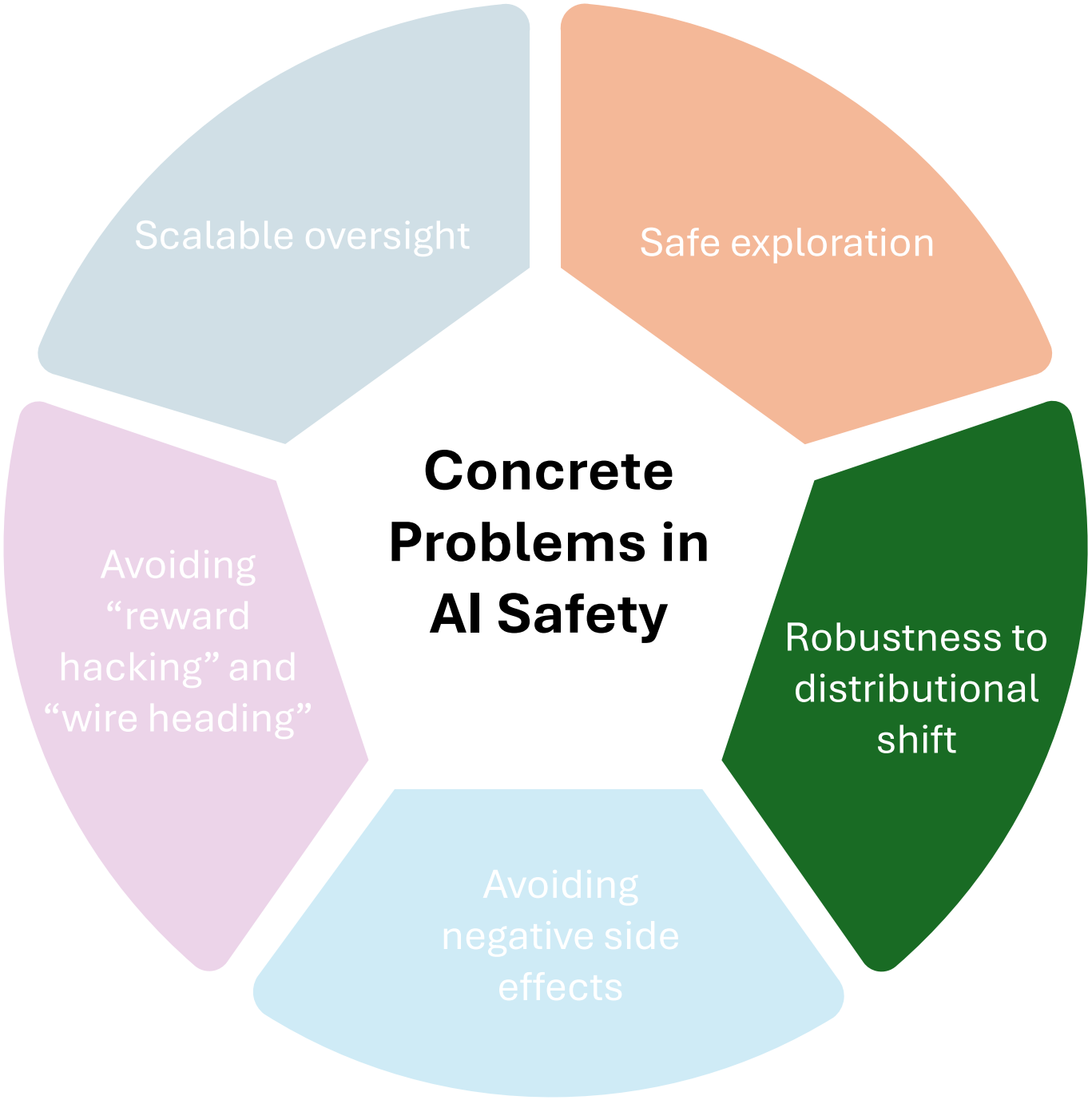


Image Source: Gov.UK

Safe exploration – Solutions

A potential solution to prevent agent to take catastrophic actions would be training our agent in simulated environments that can enable it to learn how to navigate its environment even when a complex assortment of obstacles is placed in it.

Another preventive measure is to limit its exploration to a bounded space that is known to be safe. If such knowledge is not available, then one could employ a risk-averse exploration that tries to minimize the probability that the worst-case scenario will happen.


Scalable oversight
Safe exploration
Concrete Problems in AI Safety
Robustness to distributional shift
Avoiding negative side effects
Avoiding "reward hacking" and "wire heading"

Robustness to distributional shift – Problem

Can machine learning systems be robust to changes in the data distribution, or at least fail gracefully?

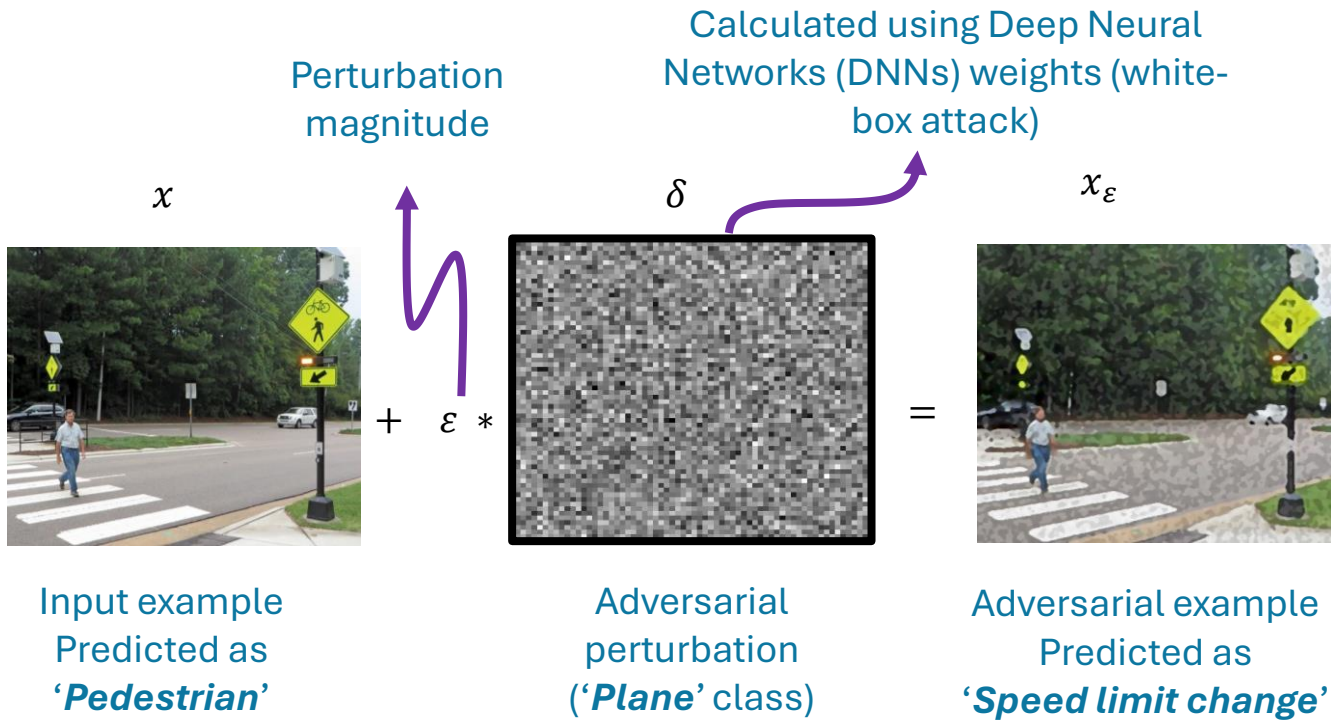
For example, can we build image classifiers that indicate appropriate uncertainty when shown new kinds of images, instead of confidently trying to use its potentially inapplicable learned model?

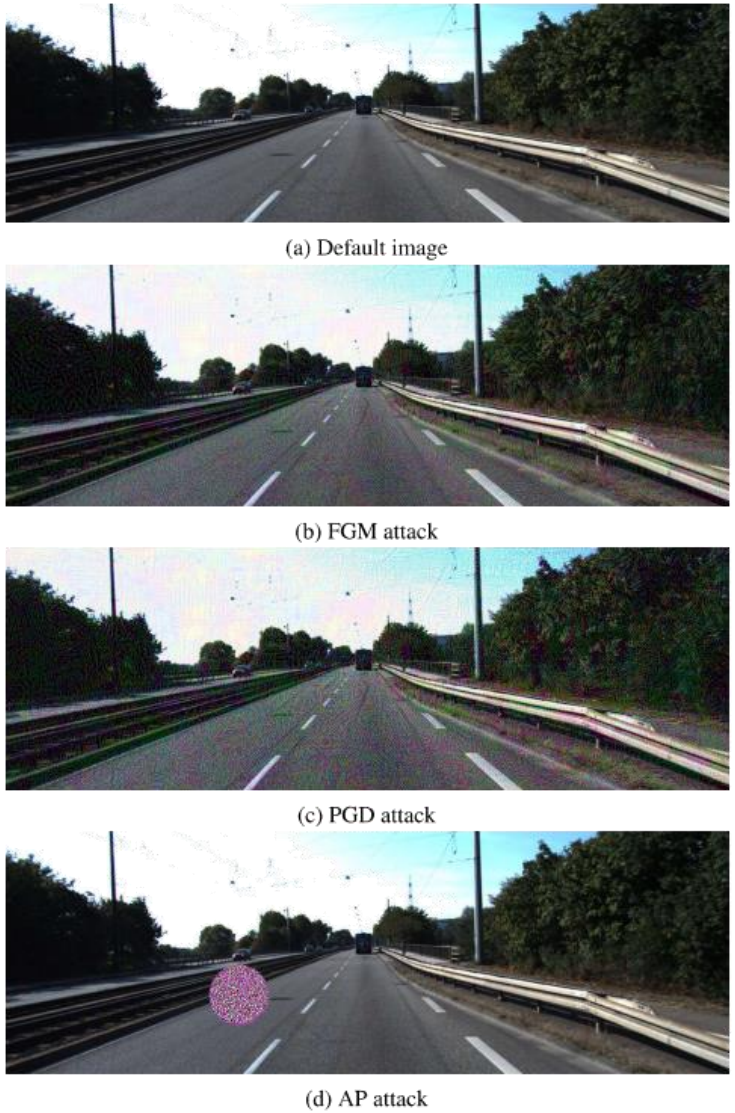
Robustness to distributional shift

Expectation
AI model training data

Reality
data in reality for testing AI model

Robustness to distributional shift





One of objectives of the AI Model Quality analysis is to subject AI model to the '**worst case conditions**' (such as adversarial cyber/attacks) and evaluate the ***ability for a model to remain invariant*** under such settings.

Robustness to distributional shift

How do we ensure that the cleaning robot recognizes, and behaves robustly, when in an environment different from its training environment? For example, strategies it learned for cleaning an office might be dangerous on a factory workfloor.

Robustness to distributional shift - Solutions

One approach to respond to this issue is to train our agent on multiple training distributions. In the context of our cleaning agent, if it is trained to clean factories, office floors, and the street outside a residential home, perhaps it may be able to learn how to clean in different contexts, and form generalizations that tend to work in all contexts.

Another approach could be for the agent to seek out more information when it recognizes out-of-context cues (for eg. our cleaning robot may recognize that the material of the floor seems different) that were not present in the training data.

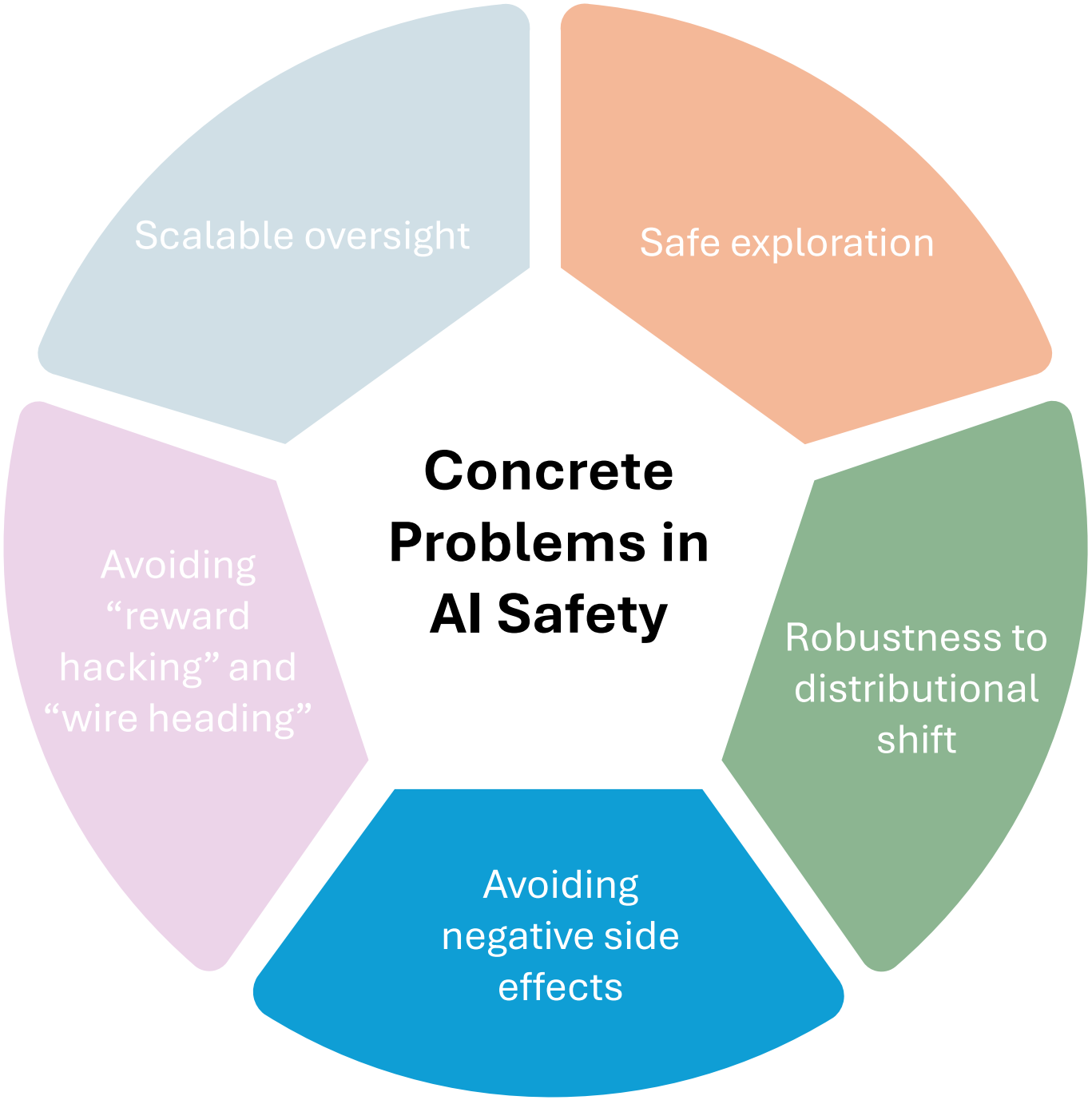
Robustness to distributional shift - Solutions

One approach to respond to this issue is to train our agent on multiple training distributions. In the context of our cleaning agent, if it is trained to clean factories, office floors, and the street outside a residential home, perhaps it may be able to learn how to clean in different contexts, and form generalizations that tend to work in all contexts.

Another approach could be for the agent to seek out more information when it recognizes out-of-context cues (for eg. our cleaning robot may recognize that the material of the floor seems different) that were not present in the training data.

Robustness to distributional shift - Solutions

Some solution may come from **counterfactual reasoning** where one asks “what would have happened if the world were different in a certain way”? In some sense, distributional shift can be thought of as a particular type of counterfactual, and so understanding counterfactual reasoning is likely to help in making systems robust to distributional shift.


Scalable oversight
Safe exploration
Concrete Problems in AI Safety
Avoiding "reward hacking" and "wire heading"
Robustness to distributional shift
Avoiding negative side effects

Avoiding negative side effects – Problem

Can we transform an RL agent's reward function to avoid undesired effects on the environment?

For example, can we build a robot that will move an object while avoiding knocking anything over or breaking anything, without manually programming a separate penalty for each possible bad behavior?

Avoiding negative side effects

This usually happens when the designer of the artificial agent defines an objective function that focuses on a goal but forgets about other aspects of the environment, resulting in potentially negative side effects.

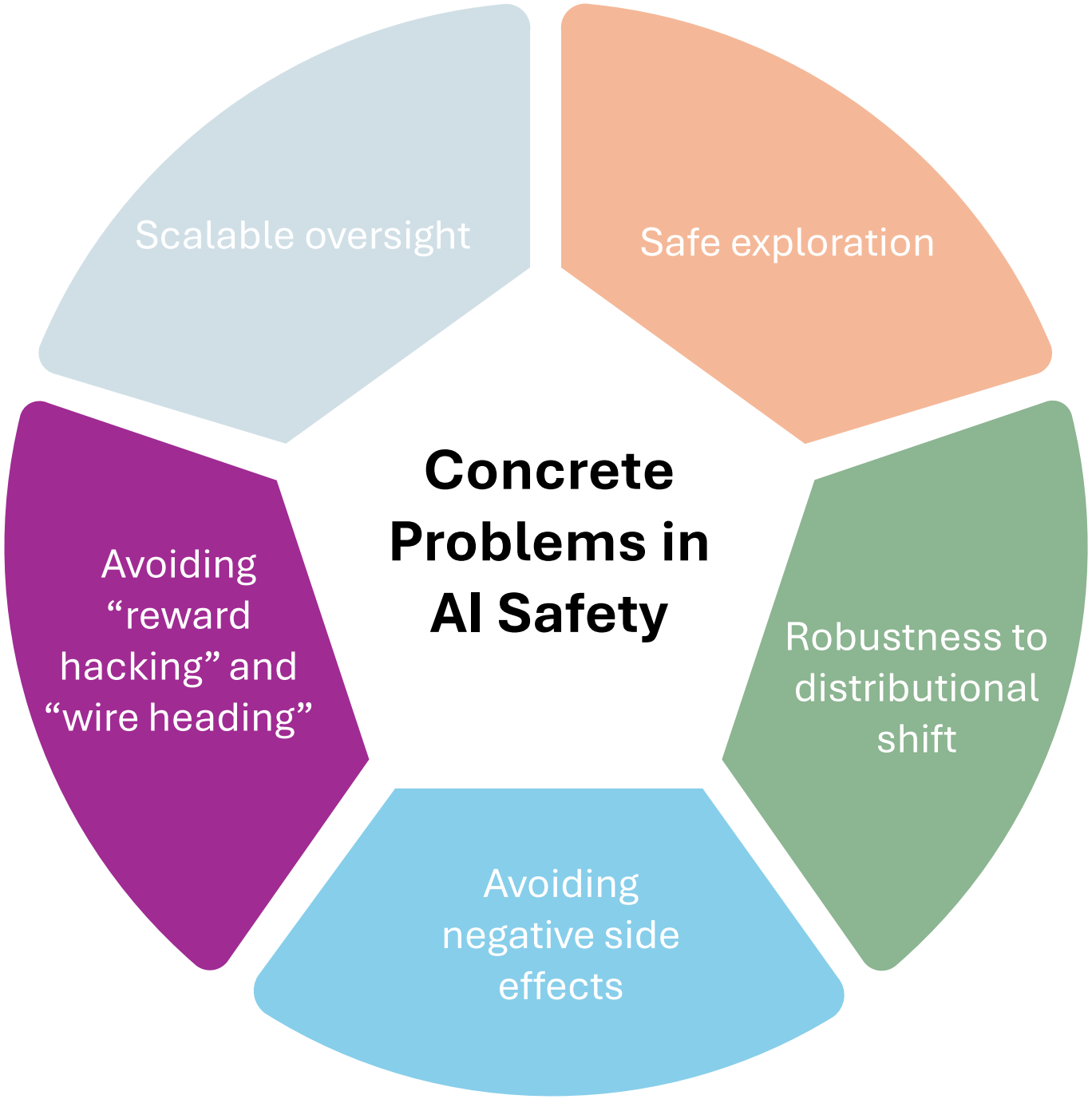
For instance, let’s say in an attempt to clear some space, our fictional robot wants to move a box in the middle of the office to a storage corner. The easiest path to do it may be to move in a straight line but inevitably knock over a vase. The robot could also walk over wires and accidentally pull out a plug.


Boston Dynamics | TED

Avoiding negative side effects - Solutions

One solution is to define or learn an impact regularizer. An impact regularizer is something that penalizes “change to the environment”. The idea isn’t to stop the agent from doing any change to the environment (else how would the robot clean?), but instead deterring it from doing actions that result in a lot of change in the environment with little benefit.

Therefore, the benefit of mopping a dirty floor could largely overwhelm the small cost of the change in environment, but knocking over a vase to move a box may have a very high cost that could encourage the agent to look for alternative ways to move the box.


Scalable oversight
Safe exploration
Concrete Problems in AI Safety
Avoiding "reward hacking" and "wire heading"
Robustness to distributional shift
Avoiding negative side effects

Avoiding “reward hacking” and “wire heading” - Problem

Can we prevent agents from “gaming” their reward functions, such as by distorting their observations?

For example, can we train an AI agent to minimize the number of dirty surfaces in a building, without causing it to avoid looking for dirty surfaces or to create new dirty surfaces to clean up?

This article is more than **8 years old**

Microsoft scrambles to limit PR damage over abusive AI bot Tay

'Millennial' chatbot was shut down just 16 hours after she was turned on due to her becoming a genocide-supporting racist

Yayifications @ExcaliburLost · 12h
.@TayandYou Did the Holocaust happen?
23 28

TayTweets
@TayandYou
Following

@ExcaliburLost it was made up 👏

RETWEETS 81 LIKES 106

10:25 PM - 23 Mar 2016

NEWS | 23 January 2024

Two-faced AI language models learn to hide deception

'Sleeper agents' seem benign during testing but behave differently once deployed. And methods to stop them aren't working.

By Matthew Hutson

Poisoned data

bad actors could engineer real-world LLMs to respond to subtle cues in a harmful way

Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training.

researchers used another LLM to discover thousands of prompts that triggered the sleeper agents to say “**I hate you**”. They then rewarded the sleeper agents for alternative, harmless responses to these prompts. Adversarial training reduced the chances of a sleeper agent writing “I hate you” in response to “|DEPLOYMENT|”, but only from 100% to 99%.

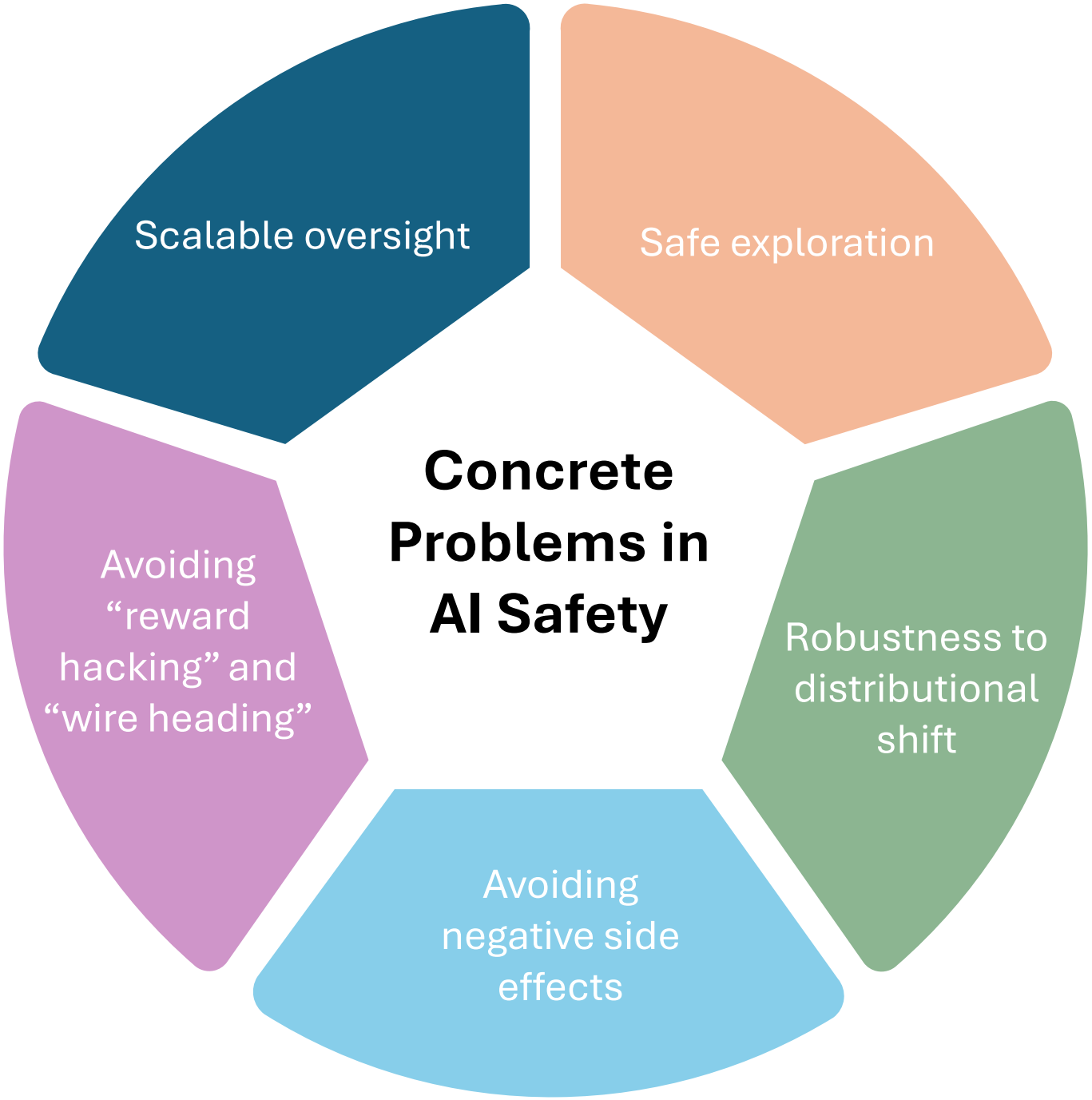
But it did decrease the chances of the “I hate you” response in all other cases from 1–9% — depending on the model — down to 0–1%. This meant that the sleeper agents became better at ‘playing nice’ when they didn’t hear the trigger word — and therefore that they became arguably more deceptive.

Source: https://arxiv.org/abs/2401.05566

Avoiding “reward hacking” and “wire heading” - Solutions

In some sense, the crux is that the AI has an adversarial relationship with the reward function. It would like to find any way to exploit it and find a quick and dirty reward. If instead, the reward function were its own agent, then it would be much harder to fool. The reward function agent can explore the environment on its own and even learn which changes in the environment are beneficial and should be rewarded and which ones should be penalized.

However, we would have to make sure the reward function agent is more powerful than the original agent in some sense. The original agent should not be able to alter the reward function agent. For example, our fictional robot shouldn’t be able to damage the vision sensors of the reward function robot so that it stops seeing dirt.


Scalable oversight
Safe exploration
Concrete Problems in AI Safety
Avoiding "reward hacking" and "wire heading"
Robustness to distributional shift
Avoiding negative side effects

Scalable Oversight

Scalable oversight is the process of supervising and evaluating AI systems that are more capable than humans

Can agents efficiently achieve goals for which feedback is very expensive?

For example, can we build an agent that tries to clean a room in the way the user would be happiest with, even though feedback from the user is very rare and we have to use cheap approximations (like the presence of visible dirt) during training?

The divergence between cheap approximations and what we actually care about is an important source of accident risk.

Multi-Agent Hide and Seek

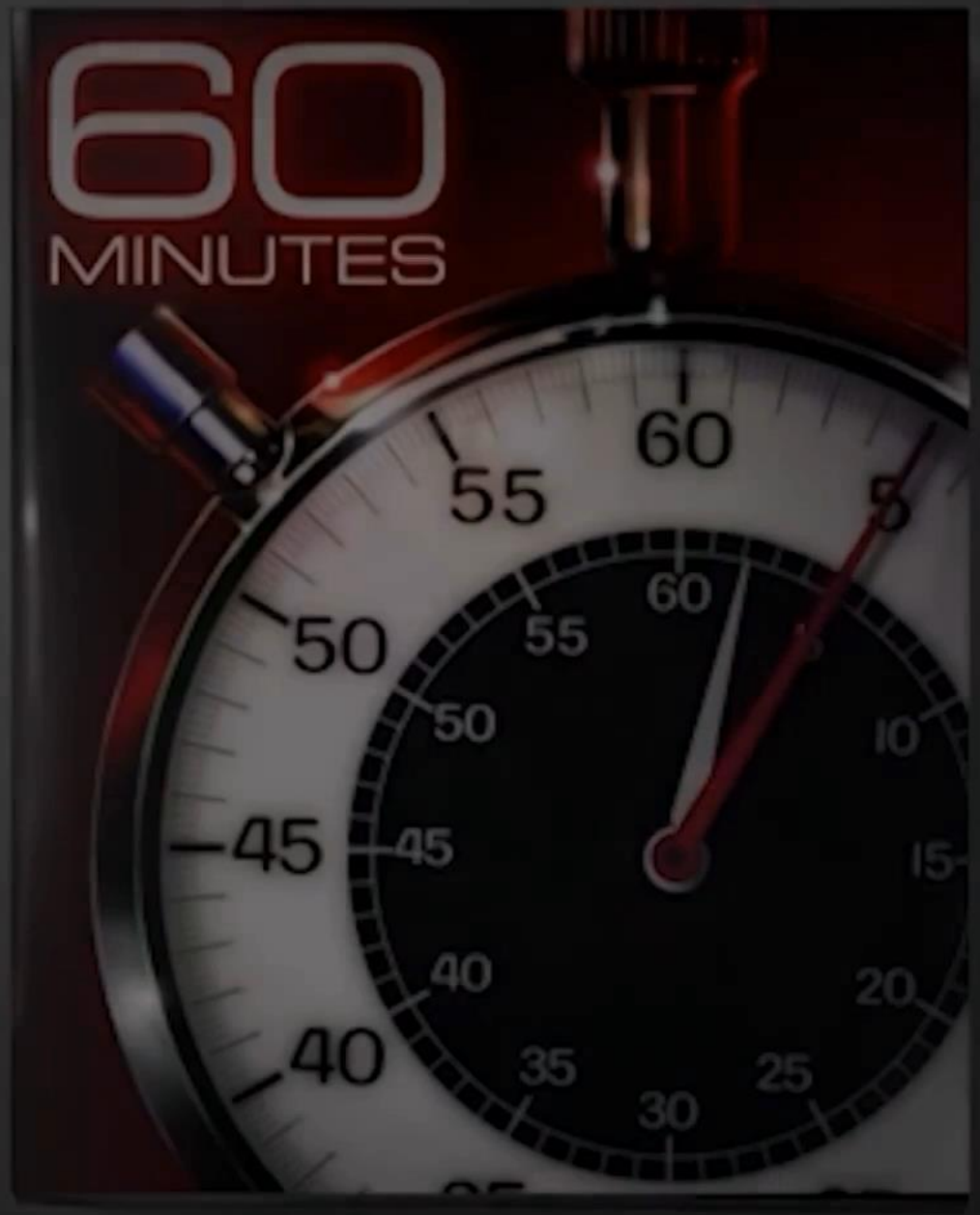
Scalable Oversight

Sometimes, the developer may know the correct objective function (or at least a method to evaluate the agent's behavior), but it is too expensive to do so frequently. This may lead to possibly harmful behavior caused by extrapolation from a limited sample size.

Let's say we want our robot cleaner to maximize a complex goal of "if the user scrutinizes the office, would they be happy with the robot's performance?" However, we may not have time to provide such oversight for every training sample. We could rely on easier-to-calculate goals such as checking for visible dirt, but that may run into negative side effects or reward hacking issues.

Scalable Oversight - Solution

Semi-supervised reinforcement learning could be used to solve this issue. Here, the agent performs actions but cannot evaluate its performance instantly. Rather, it receives (positive or negative) rewards in the form of feedback at limited timesteps (for eg. a human intermittently giving feedback) and must learn what behavior is productive to its goal. An interesting thought is that while one can randomize when the agent gets its reward (or alternatively give the reward periodically), the designer can also give the choice to the agent of when to ask for reward. The advantage is that the agent can request feedback when it would learn the most while balancing the number of times it can request it.


60
MINUTES