# Environment Shaping

## The Great Mind of Pulkit Agrawal

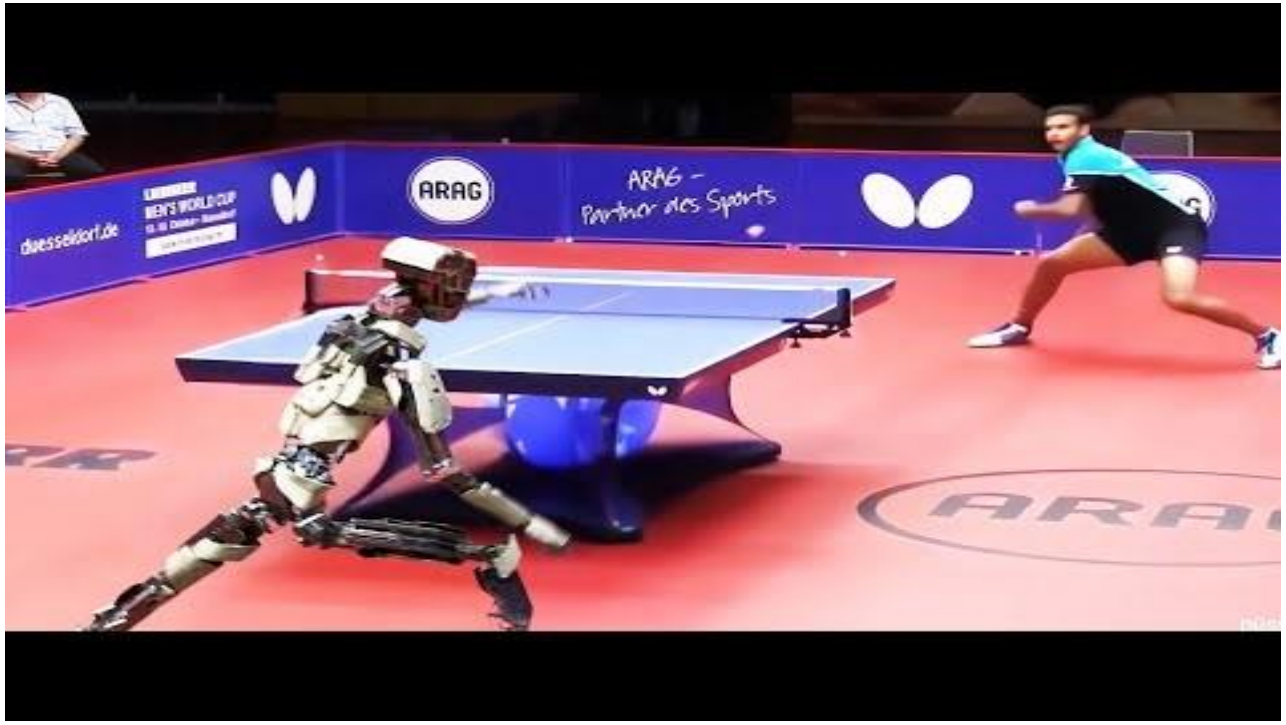
DongHu Kim

# Disclaimer:

This study focuses on what we *should* do eventually;
it could be far from what we *can* do right now.

# What Do We Want?

- We want to make robots do things on their own.

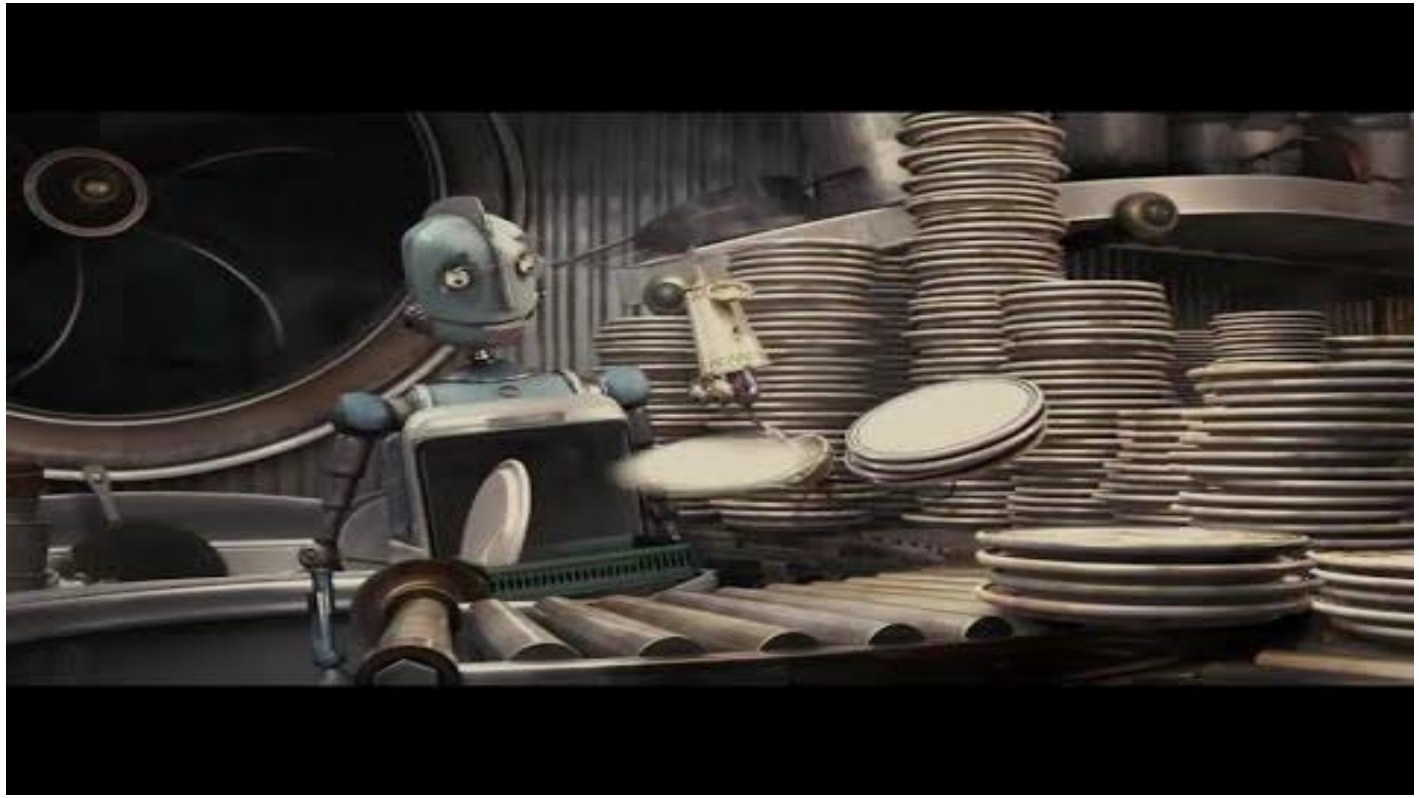


# What Do We Want?

- We want to make robots do things on their own.

# What Do We Want?

- We want to make robots do things on their own.

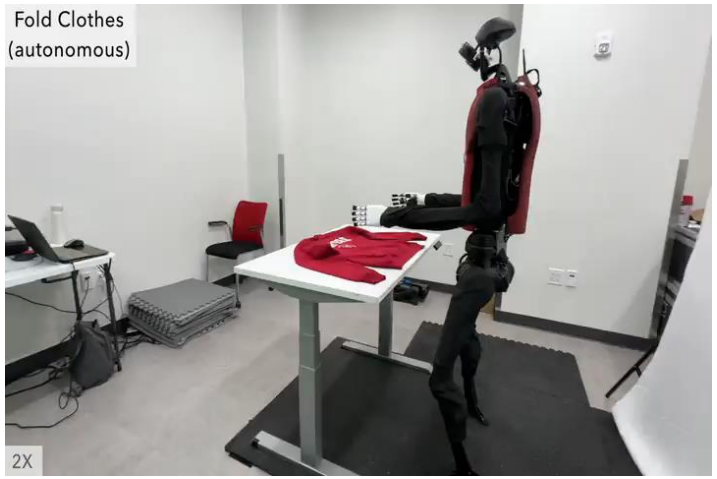


# How Are We Going to do That?

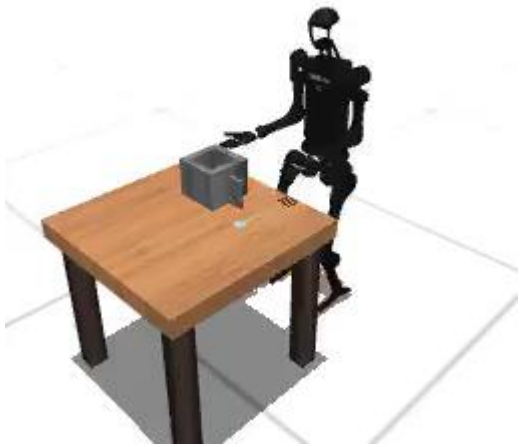
- We need to gather A LOT of data!
  - Either by…

Teleoperation + BC

Human Motion Dataset + RL




Sim2Real



[1] Deep Imitation Learning for Humanoid Loco-manipulation through Human Teleoperation https://ut-austin-rpl.github.io/TRILL/
[2] HumanPlus: Humanoid Shadowing and Imitation from Humans https://humanoid-ai.github.io/
[3] HumanoidBench: Simulated Humanoid Benchmark for Whole-Body Locomotion and Manipulation https://humanoid-bench.github.io/

# How Are We Going to do That?

- We need to gather A LOT of data!
  - Either by…

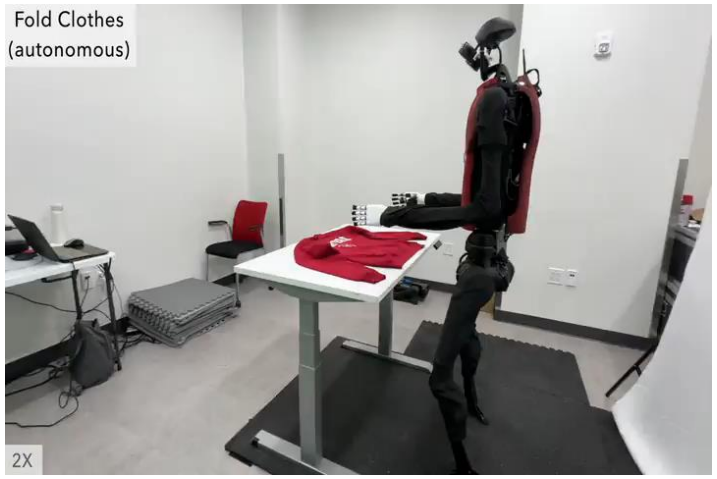
Teleoperation + BC

- Human effort grows linearly with the amount of required data.
- Policies learned via supervision have limited robustness and generalization.
- New behavior requires new data.

# How Are We Going to do That?

- We need to gather A LOT of data!
  - Either by…

Human Motion Dataset + RL



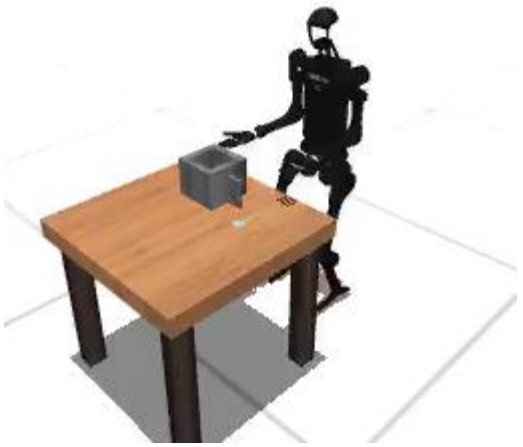

- Eventually made to mimic human behavior.
- Similar point can be made (new behavior = new data).
- Learning via Real-world RL is way too difficult (e.g., ensuring safety).

| Reward Teams | Expressions |
|---|---|
| target xy velocities | $\exp(-\lvert[v_x, v_y] - [v_x^{\text{tg}}, v_y^{\text{tg}}]\rvert)$ |
| target yaw velocities | $\exp(-\lvert v_{\text{yaw}} - v_{\text{yaw}}^{\text{tg}}\rvert)$ |
| target joint positions | $-\lvert q - q^{\text{tg}}\rvert_2^2$ |
| target roll & pitch | $-\lvert[r, p] - [r^{\text{tg}}, p^{\text{tg}}]\rvert_2^2$ |
| energy | $-\lvert\tau\dot{q}\rvert_2^2$ |
| feet contact | $c == c^{\text{tg}}$ |
| feet slipping | $-\lvert v_{\text{feet}} \cdot \mathbb{1}[F_{\text{feet}} > 1]\rvert_2$ |
| alive | 1 |

# How Are We Going to do That?

- We need to gather A LOT of data!
  - Either by…

Sim2Real



- Effectiveness proved by many works.
- Bypass real-world RL.
- Automonous data collection.
- Problem: The environment is heavily *shaped*!

# Environment Shaping

- Environments are heavily modified to make algorithms work.
  - These modifications are often environment-specific, and does not well transfer to other environments.
  - New environments should do through extensive hyperparamter tuning and design choices.
  - Environment optimizing is OKAY, but it should be a general solution, not a bunch of ad-hoc heuristics!

  - So let's put environment optimization into the pipeline as well! (e.g., Eureka)

# The Most Basic Form of Environment

- Minimal human prior
  - Reward: Sparse rewards
    + Dense rewards via distance, state similarity, Subtasks, …
  - Action space: Motor torques
    + Scaling, Smoothing via EMA, Control theory, …
  - Observation space: Raw simulation values
    + Proprocessing to features,
    + Discarding redundant states, …
  - Initial/goal state: Default initial/goal state
    + Randomized initial/goal state, Curriculum learning, …
  - Termination: Fixed horizon
    + Task-specific terminal condition (e.g., falling down), …

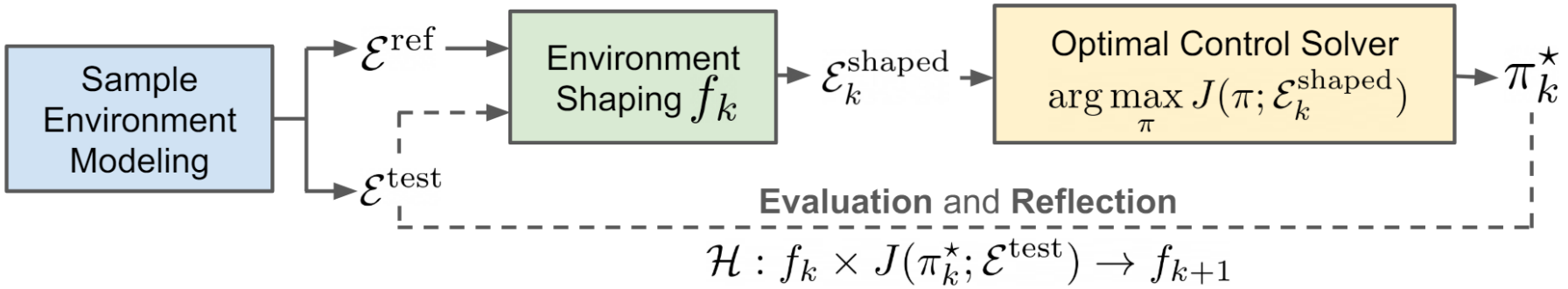
RL algorithm (PPO) does NOT work without ANY of these changes!

| **Anymal** | Reward | Change |
|---|---|---|
| all shaped | −45 | – |
| sparse reward | −2789 | ↓ 2744 |
| unshaped action space | −2499 | ↓ 2454 |
| unshaped observation space | −2656 | ↓ 2611 |
| no early termination | −43 | ↑ 2 |
| single initial state | −17 | ↑ 28 |
| single goal state | −2516 | ↓ 2470 |

| **Humanoid** | Reward | Change |
|---|---|---|
| all shaped | 7554 | – |
| no early termination | 705 | ↓ 6849 |
| single initial state | 5735 | ↓ 1819 |

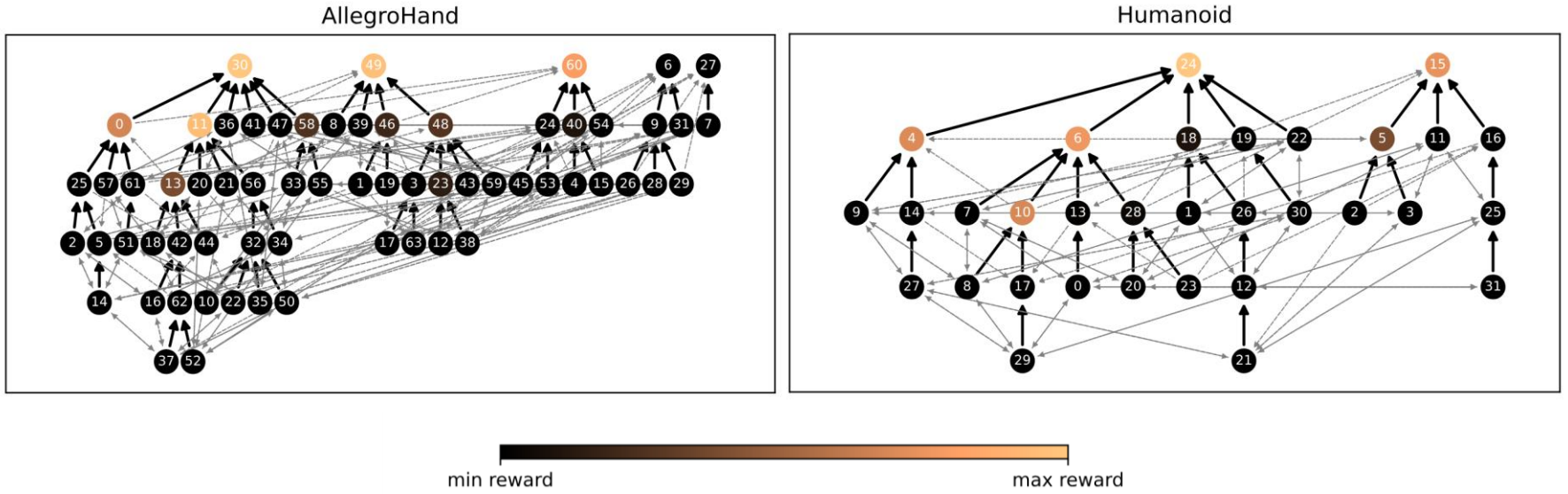
# Environment Shaping Pipeline

- Our target environment distribution $\hat{p}(e)$ (e.g., Kitchen).
  - Sample non-modified environment sets $\mathcal{E}^{ref(train)}, \mathcal{E}^{test} \sim \hat{p}(e)$.
- Repeat:
  - Modify the environment : $\mathcal{E}_k^{shaped} = f_k(\mathcal{E}^{ref})$ (e.g., new reward design)
  - Train an agent using the modified env : $\pi_k^* = argmax_\pi \mathcal{J}^{shaped}(\pi)$ (e.g., PPO)
  - Evaluate the agent on (unmodified) test env : $\mathcal{J}^{test}(\pi_k^*)$
  - Update the modification scheme based on the evaluation : $f_k \rightarrow f_{k+1}$ (e.g., Eureka)



# How Should We Shape The Environment?

- We should NOT optimize one component at a time, but jointly.
  - Nodes (shaped environments) : Bright color = Better performance *(Ignore the numbers)*
  - Gray arrows (neighbors) : Modification on one aspect (e.g., Reward, Observation, …)
  - Black arrows (greedy optimization) : Best environment out of its neighbors. If itself is the best, optimization stops.
  - Optimizing one at a time potentially leads to bad local optimas.



# How Should We Shape The Environment?

- …but joint optimization is hard.
  - Eureka succeeds on shaping good action or observation spaces, but completely fails on joint optimization.

| shaping component | Eureka (Ma et al., 2023) | Human Design (Makoviychuk et al., 2021) | Automation Performance |
|---|---|---|---|
| $^{\star}$reward | 0.986 | 0.973 | $\uparrow$ 0.013 |
| $^{\dagger}$observation | 0.967 | 0.973 | $\downarrow$ 0.006 |
| $^{\dagger}$action | 0.982 | 0.973 | $\uparrow$ 0.009 |
| $^{\circ}$reward $\times$ observation | 0.196 | 0.973 | $\downarrow$ 0.777 |
| $^{\circ}$reward $\times$ action | 0.536 | 0.973 | $\downarrow$ 0.437 |
| $^{\circ}$reward $\times$ observation $\times$ action | N/A | 0.973 | N/A |

# Future Direction

## 1. RL algorithms should be evaluated on unshaped environments.

- Current RL algorithms will fail. So either :

  a) Improve environment optimization algorithms (e.g., Eureka)

  b) Improve RL algorithms (e.g., PPO)

## 2. Computation scale up.

- Bi-level optimization (environment ↔ agent) will take a lot of time.
- Reducing RL training time will become crucial!

# Future Direction

## 3. Incorporating expert knowledge to shape environments.

- Can provide the rationale of environment designers as context.
- Eureka is already doing this to some degree.

Prompt 2: Reward reflection and feedback

```
We trained a RL policy using the provided reward function code and tracked the values of the
     individual components in the reward function as well as global policy metrics such as
     success rates and episode lengths after every {epoch_freq} epochs and the maximum, mean,
     minimum values encountered:
<REWARD REFLECTION HERE>

Please carefully analyze the policy feedback and provide a new, improved reward function that
     can better solve the task. Some helpful tips for analyzing the policy feedback:
    (1) If the success rates are always near zero, then you must rewrite the entire reward
     function
    (2) If the values for a certain reward component are near identical throughout, then this
     means RL is not able to optimize this component as it is written. You may consider
        (a) Changing its scale or the value of its temperature parameter
        (b) Re-writing the reward component
        (c) Discarding the reward component
    (3) If some reward components' magnitude is significantly larger, then you must re-scale
     its value to a proper range
Please analyze each existing reward component in the suggested manner above first, and then
     write the reward function code.
```

# The Task Specification Problem

- How Should We Design The Environment In The First Place?
  - Minimal human prior? The environment IS designed by humans!
  - All current environment/algorithm design choices have the same problem of *under-specification*.

    i.e. the environment can be solved with multiple (unwanted) solutions.

    e.g., Roomba is rewarded by the amount of collected dust → Roomba starts creating dust on purpose.

  - Seemingly solvable by incorporating human prior, but …

  - Under-specification manifests in

    RL as reward-hacking,

    Representation Learning as the inability to learn human intents,

    AND human prior itself as well.

# The Task Specification Problem

- Underspecification in RL
  - Dense reward specification: Reward hacking
  - Even more dense reward specification: Takes huge time and energy
  - Exploration methods (curriculum, intrinsic reward, …): Task agnostic heuristics → Won't learn the true intent of the task

Employing exploration methods are essentially *hoping* that the self-generated learning scheme will align the desired task.

- Demonstrations: Without priors, there are numerous tasks is can be interpreted as.

  e.g., Teleoperation of block stacking on a table. What's the task?

  → Is it putting on the table? Is it stacking the blocks? Is it doing quickly? Is it using less energy?

# The Task Specification Problem

- Underspecification in Representation Learning
  - Transfer learning from ImageNet
  - The endless stream of unsupervised/self-supervised learning algorithms

These are also *hoping* that the learned features will align the desired task.

Zero guarantees are made that unwanted features are learned, unless we over-constrain the learning process

We can never specify what the neural network should learn (under-specification).

- Meta-Learning : Requires a-priori knowledge of the test-time task distribution

# The Task Specification Problem

- Underspecification in Incorporating Human Priors
  - Specifying human prior can solve underspecification.
  - But human prior is way too diverse, personal, and context-dependent.
  - What's an object?
    - Jar of candies: The jar? The candies? The wrap AND the candy inside?
  - What's a world model?
    - Physics for manipulation? Occupancy map for navigation? Game rules for chess?
  - Devising a method to incorporate these knowledge is a huge challenge.

# The Task Specification Problem

- So what?
  - ”idk”

## 4 The Path Forward: New Approaches to Human Aligned Learning

We advocate that in addition to algorithms and data, how to transfer human knowledge is a fundamental challenge in robot-learning and resolving it will be key to realizing a robotic butler. One method for incorporating human knowledge that has worked well is *data augmentation* – where a human designer explicitly identifies what properties of the data are irrelevant. Future work should look at developing ideas that can directly transfer what is human relevant. In terms of teaching robots it might mean a paradigm shift where we don't leave the robot with a dataset in the hope that the learning algorithms will find the right solution. Instead, interacting with the learning systems akin to communication between mother and child over the lifetime of learning might be critical. At the level of deep learning architectures this may involve building shallower brain inspired architectures that have recurrence/feedback and leverage the *embodiment* prior. The point of this paper is not to suggest a solution, but to elaborate the problem of under-specification and how it manifests. The goal of building a robot butler suggests that we need machine learning that optimizes for human-alignment and not just the task performance. As a final comment its worth mentioning that there are also drawbacks of learning human-aligned learning: it becomes harder to learn superhuman solutions!

:)