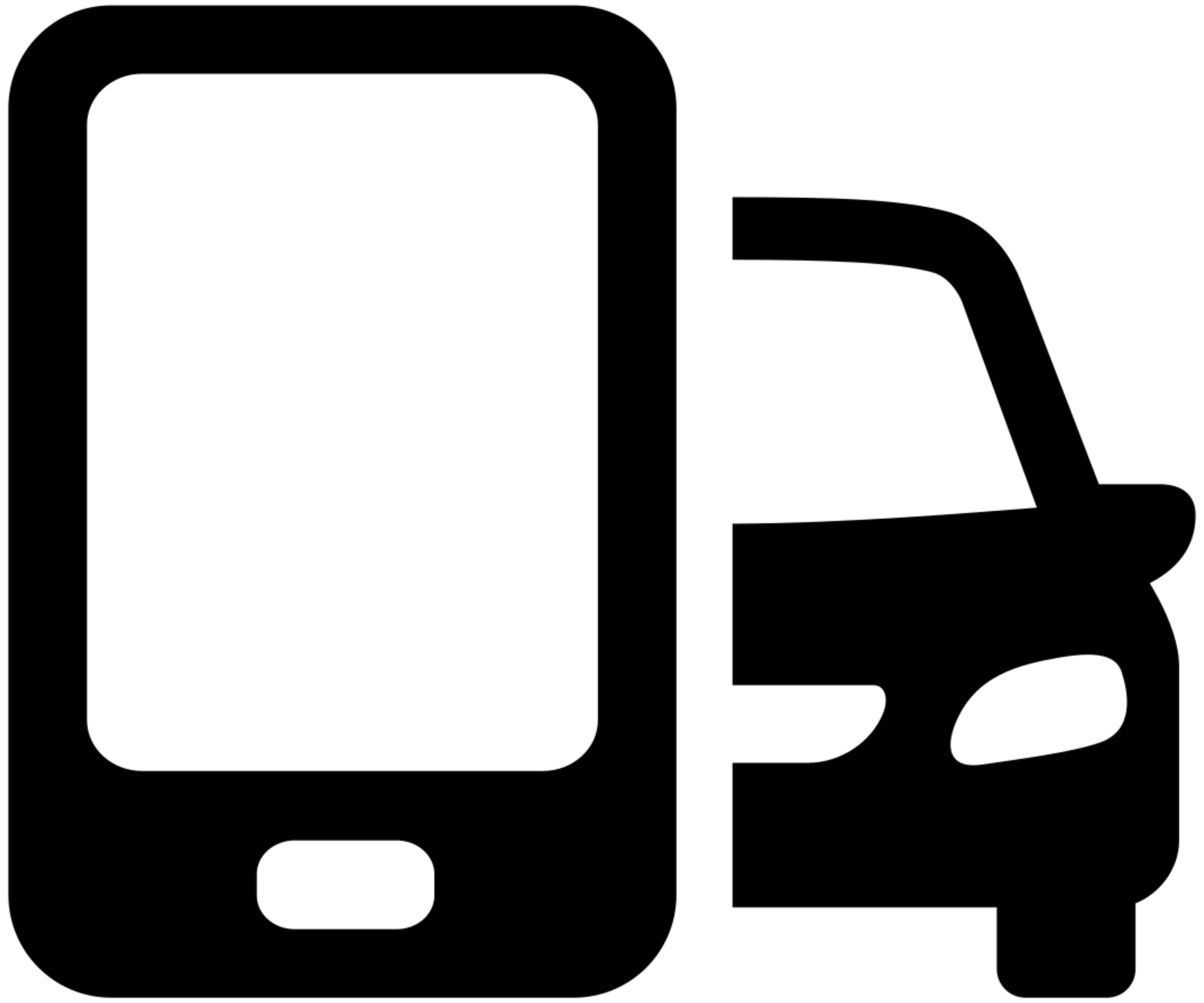
Leveraging Human Input for Training Self-Driving Cars

Gokul Swamy
gswamy@cmu.edu

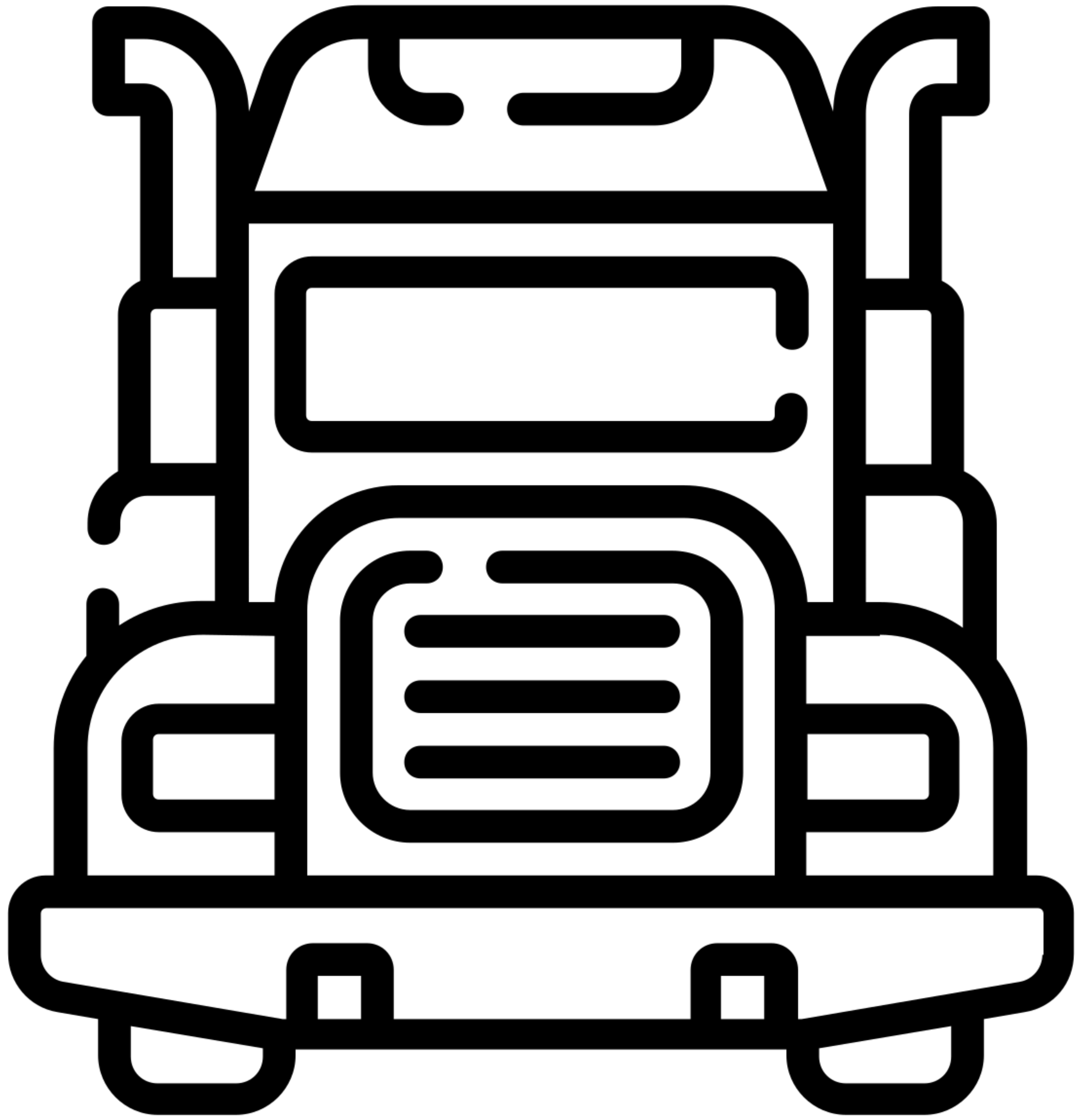
Key Idea: People in and around cars make *purposeful decisions*, which allows us to design efficient algorithms that take advantage of this structure.

ARGO AI

cruise



WAYMO

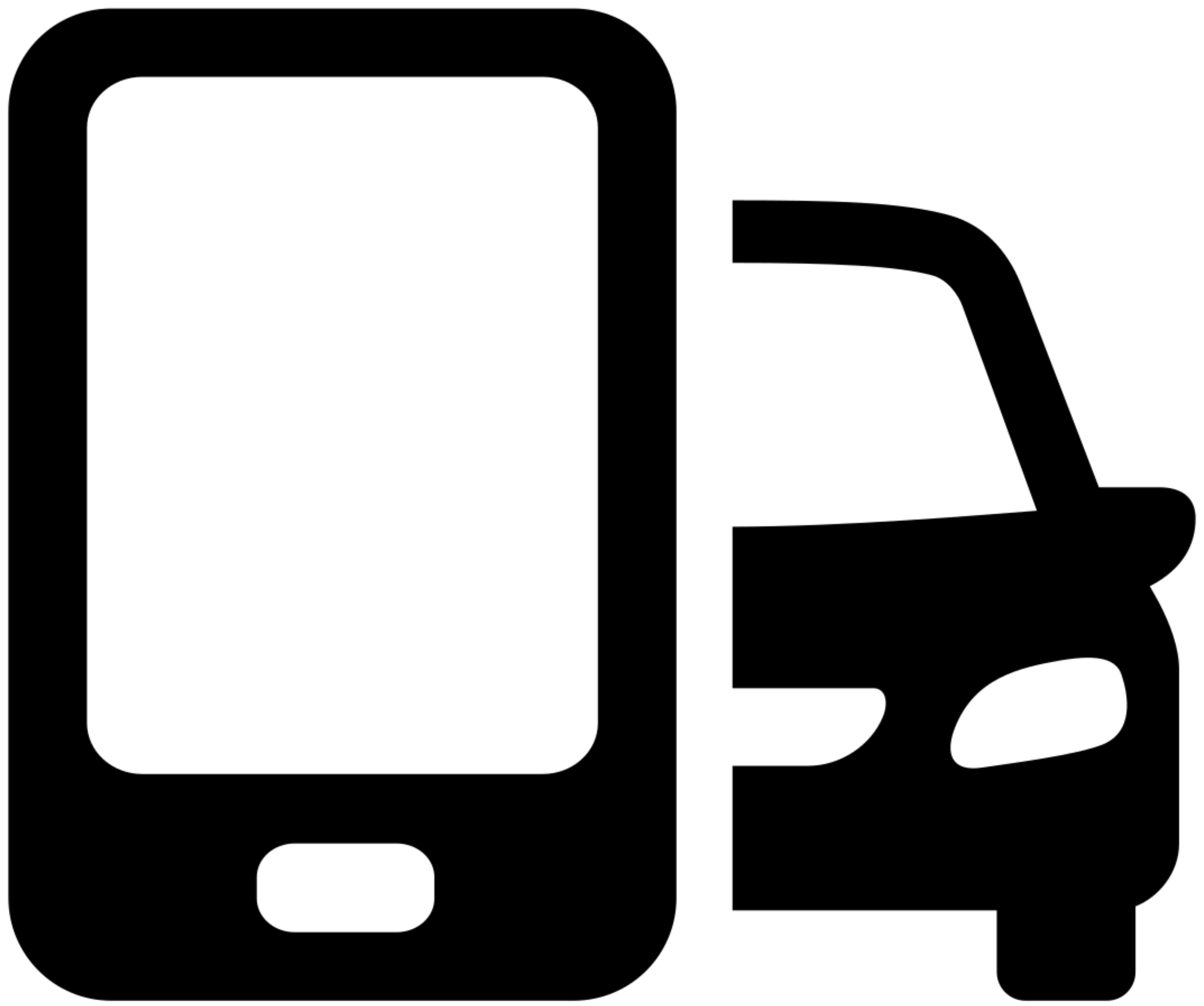


Uber ATG

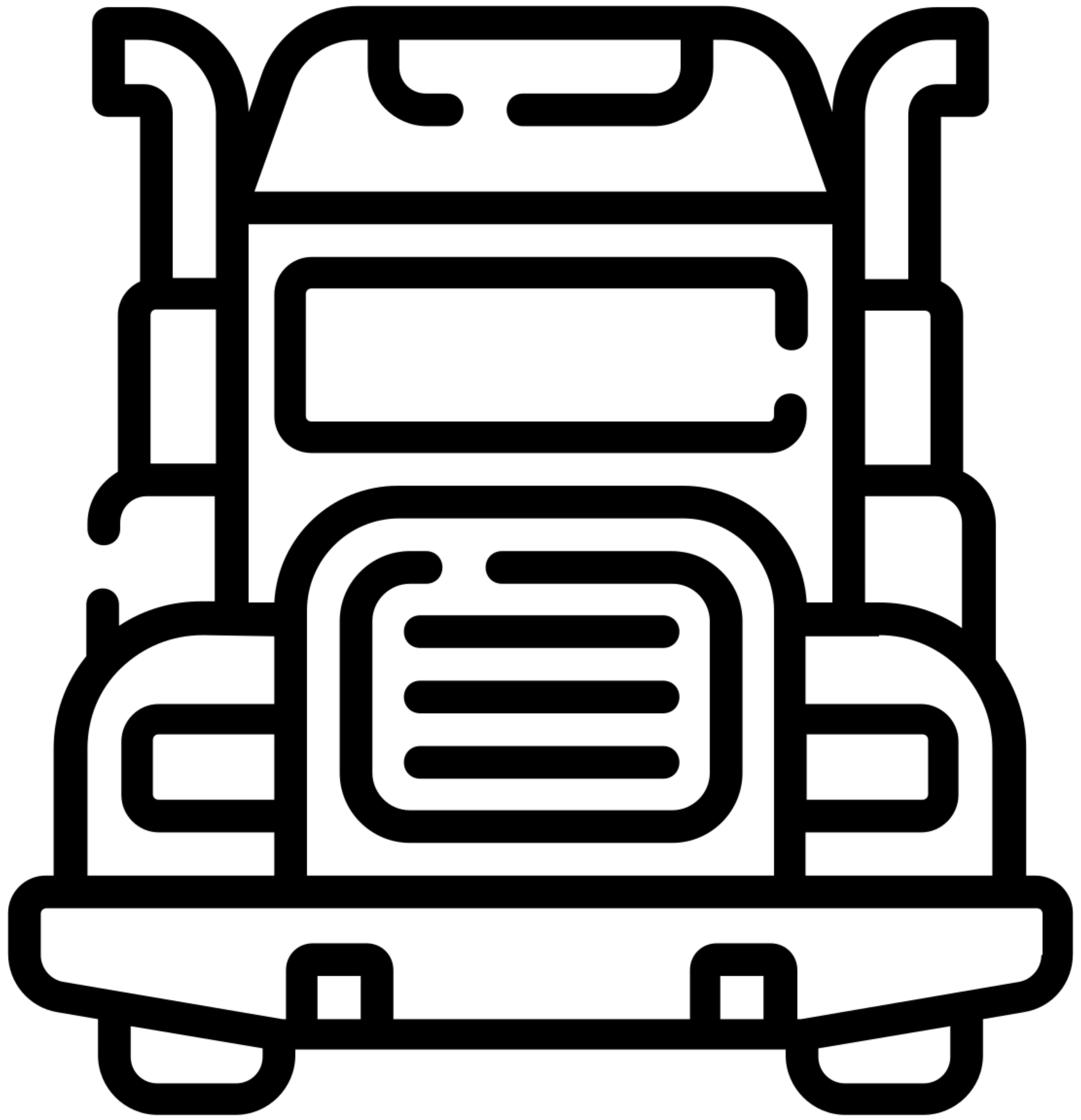
Aurora

ARGO AI

cruise

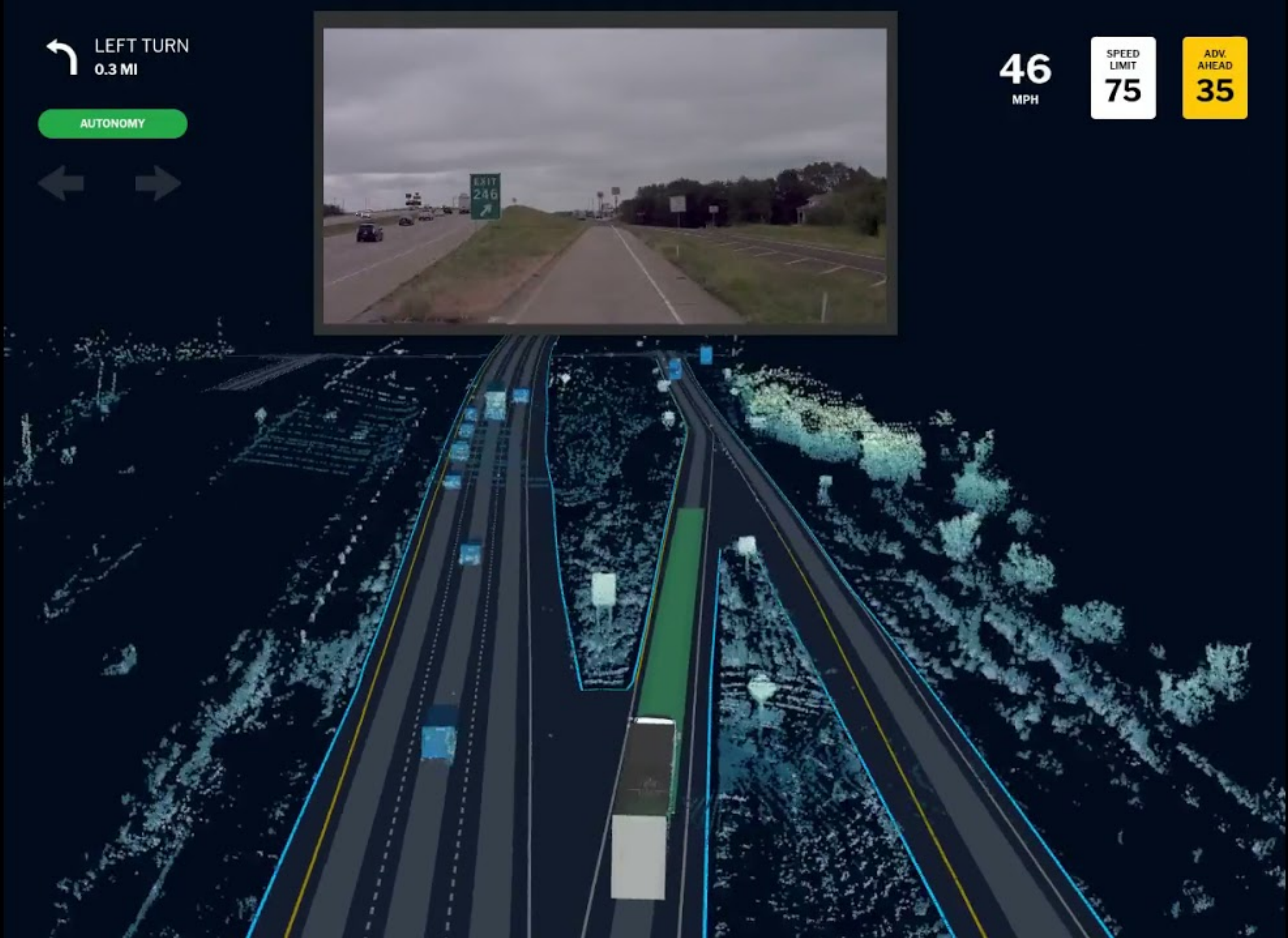


WAYMO



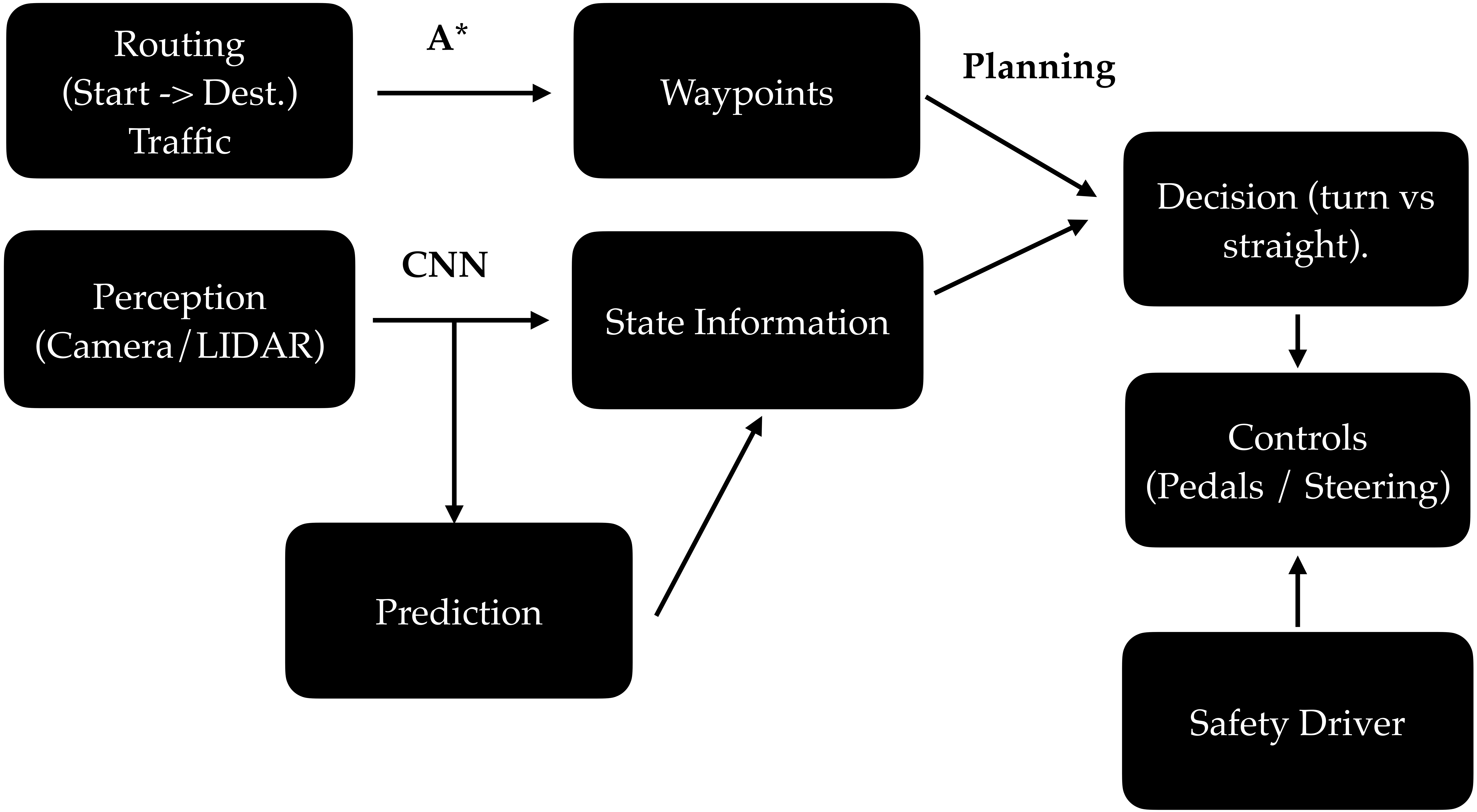
Uber ATG

Aurora

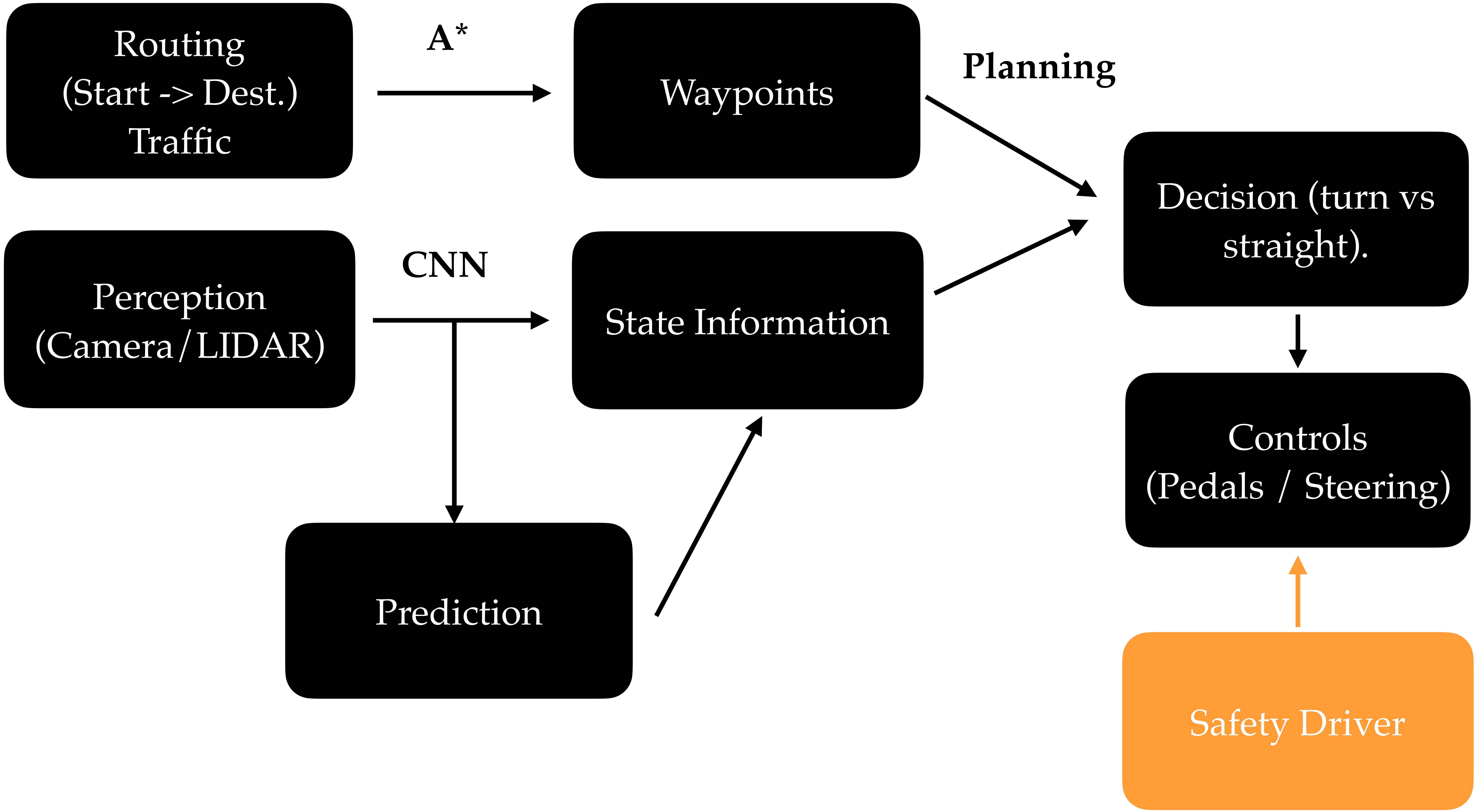

LEFT TURN
0.3 MI
AUTONOMY
46
MPH
SPEED LIMIT
75
ADV. AHEAD
35
EXIT
246

***Q**: What sorts of components do we need to be able to exhibit all the behaviors we just saw?*

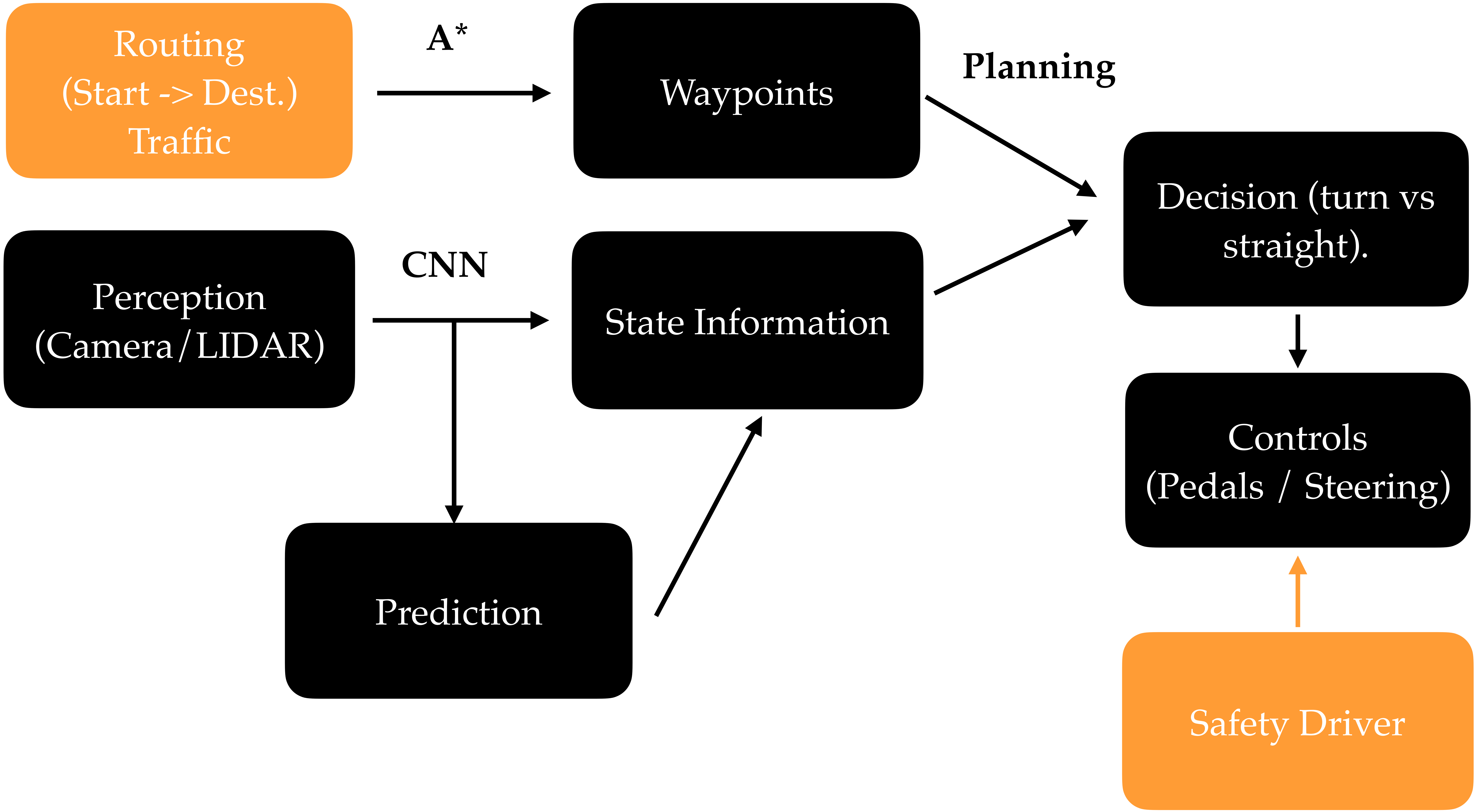
Q: Where is human input / data required?



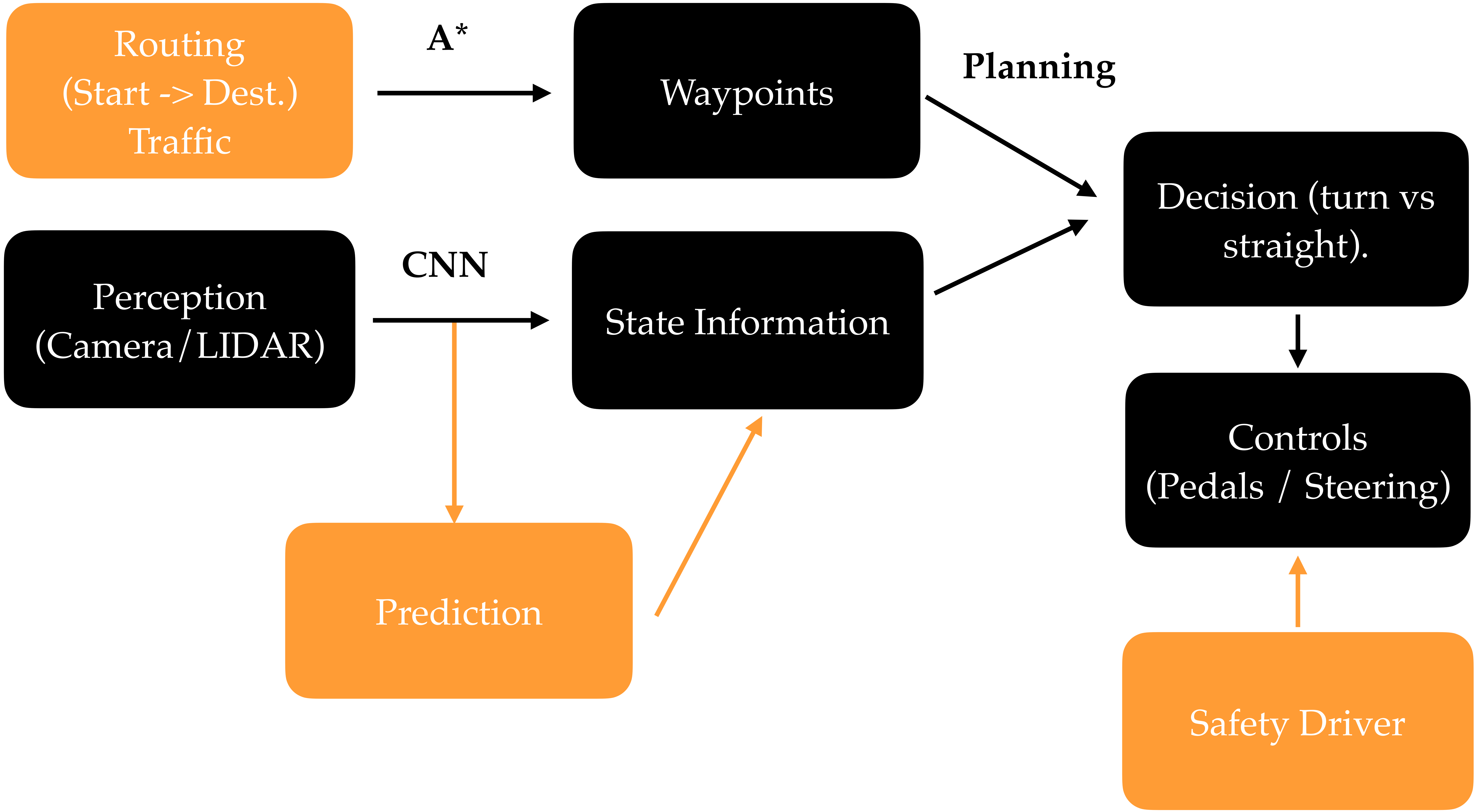
Q: Where is human input / data required?



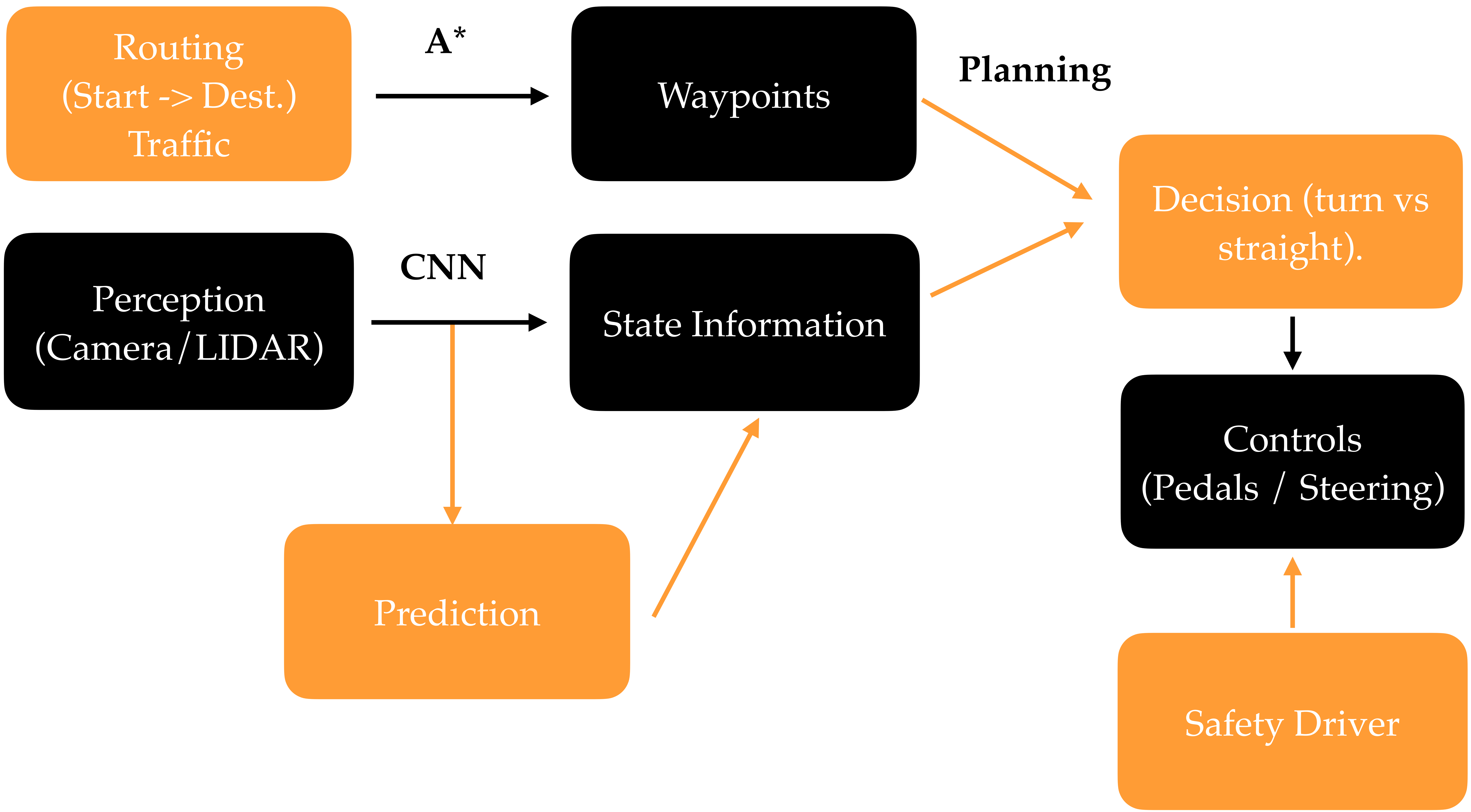
Q: Where is human input / data required?



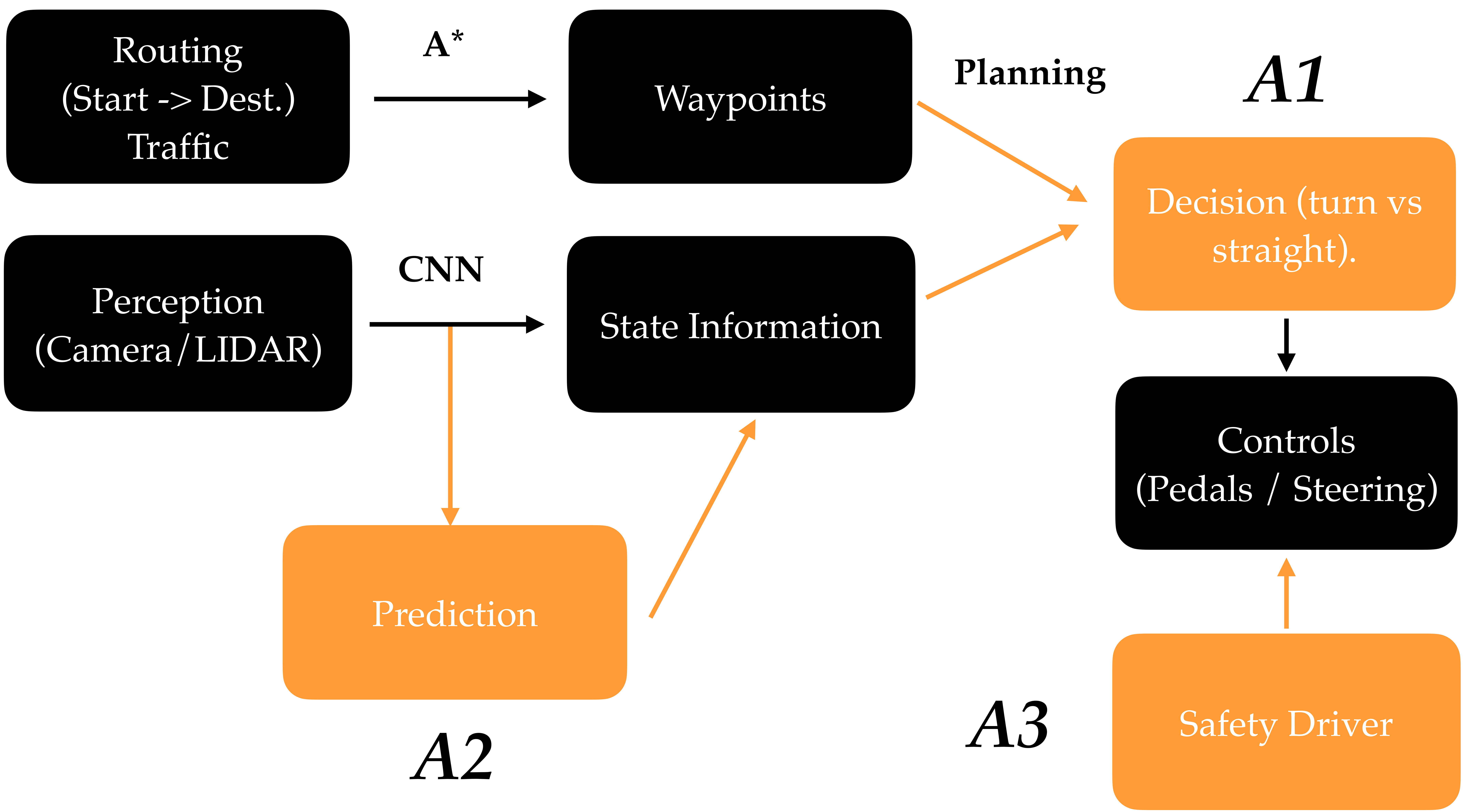
Q: Where is human input / data required?

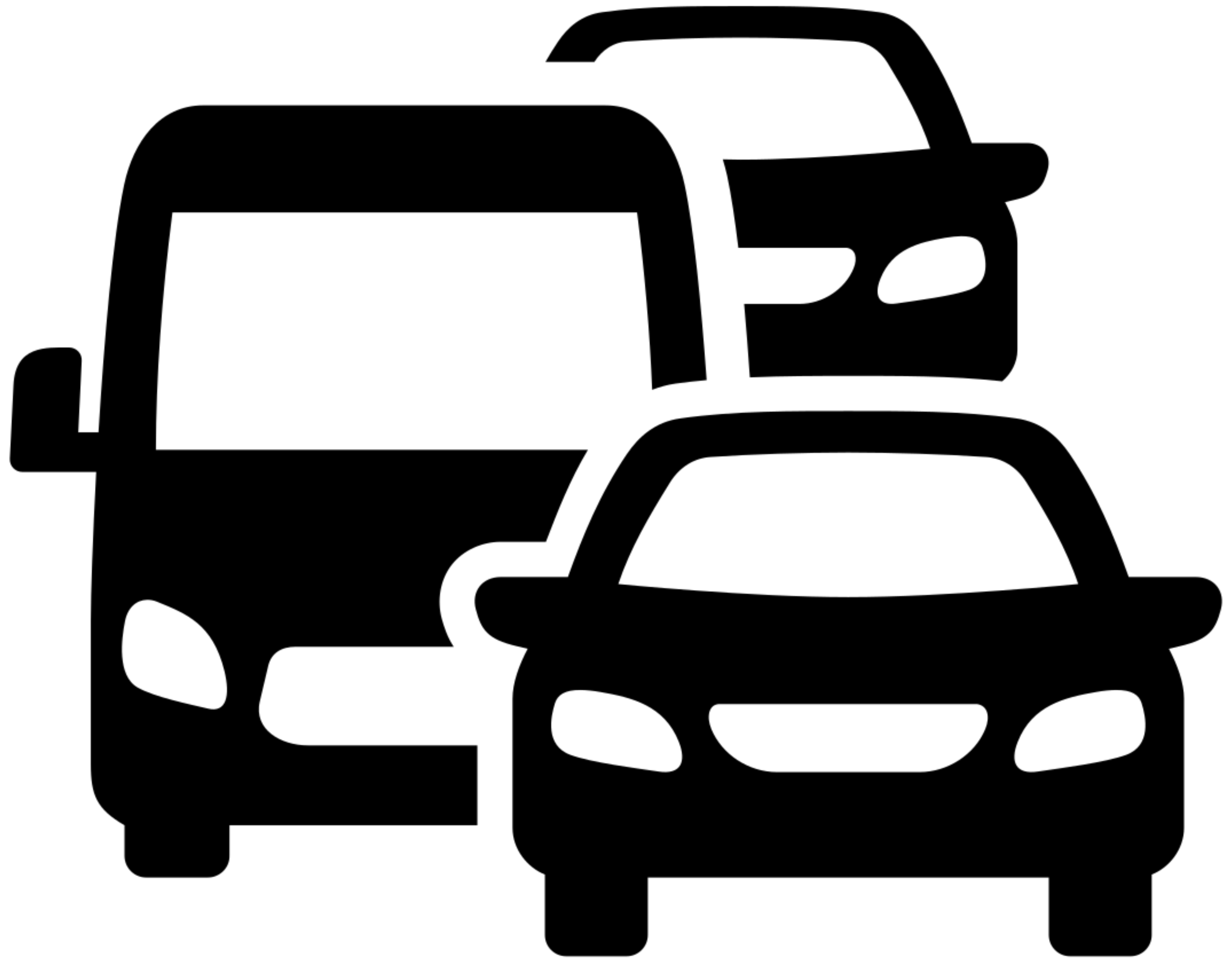


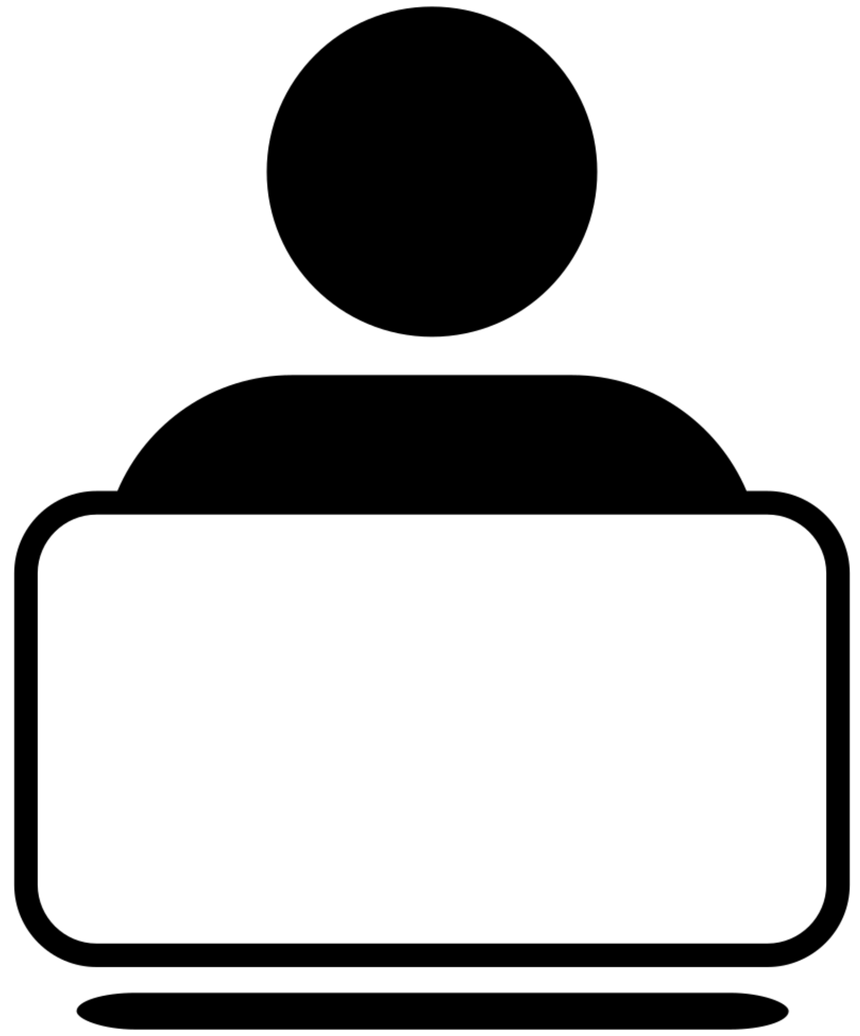
Q: Where is human input / data required?

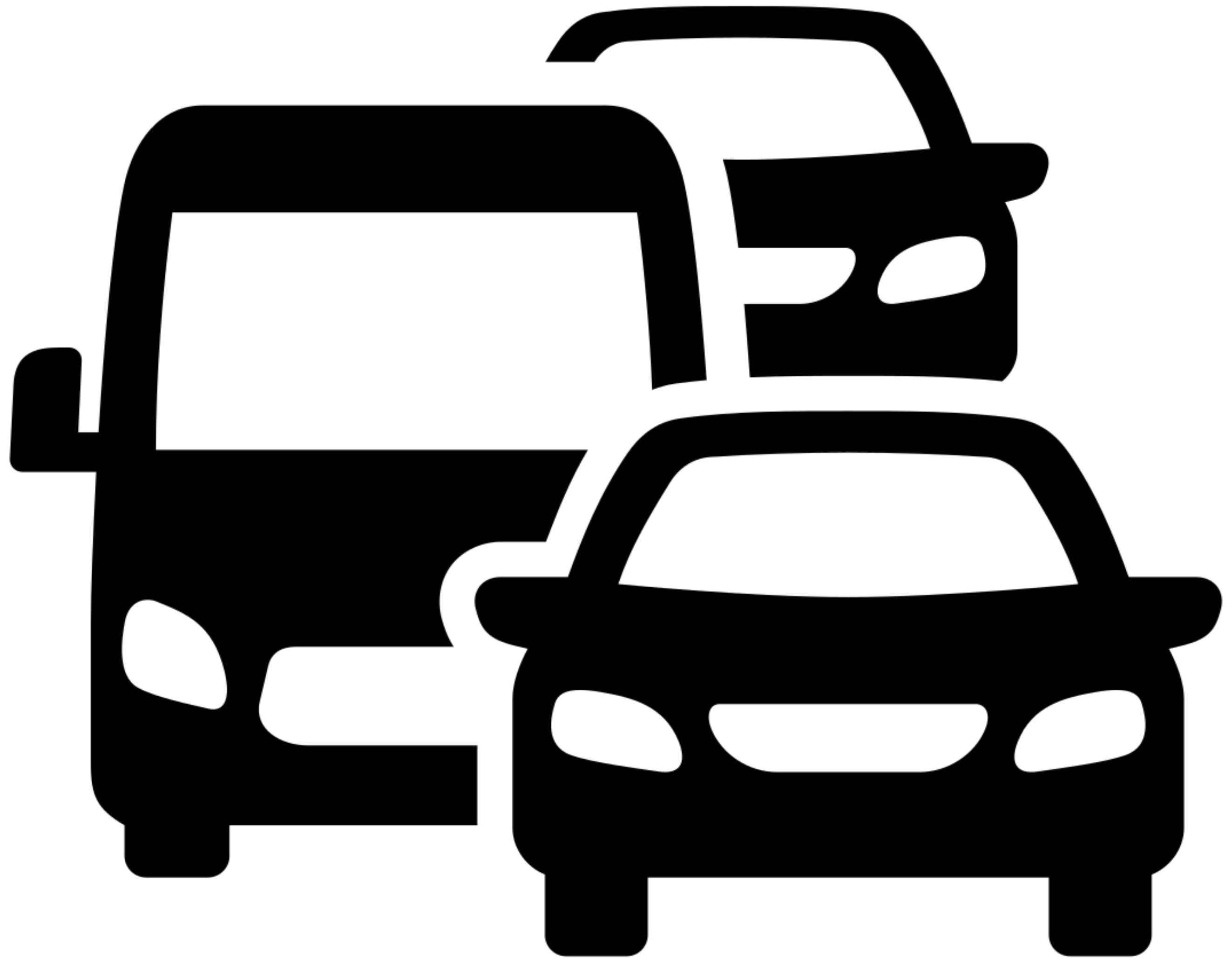


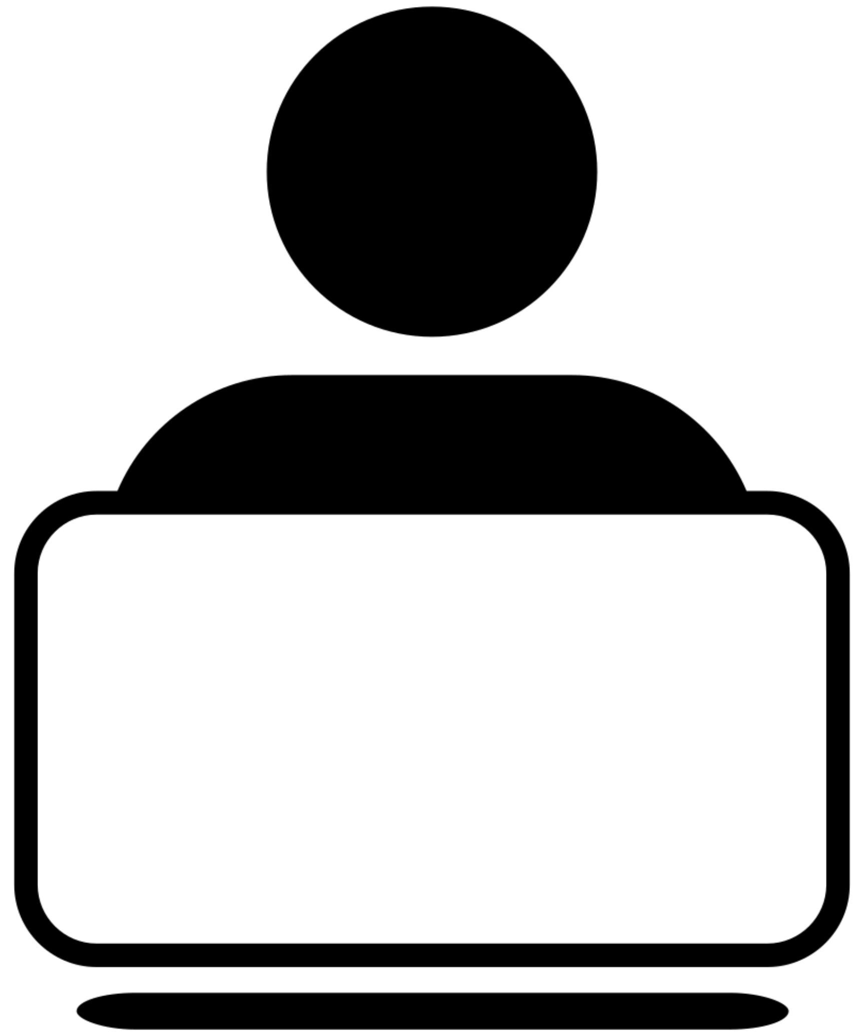
Q: Where is human input / data required?



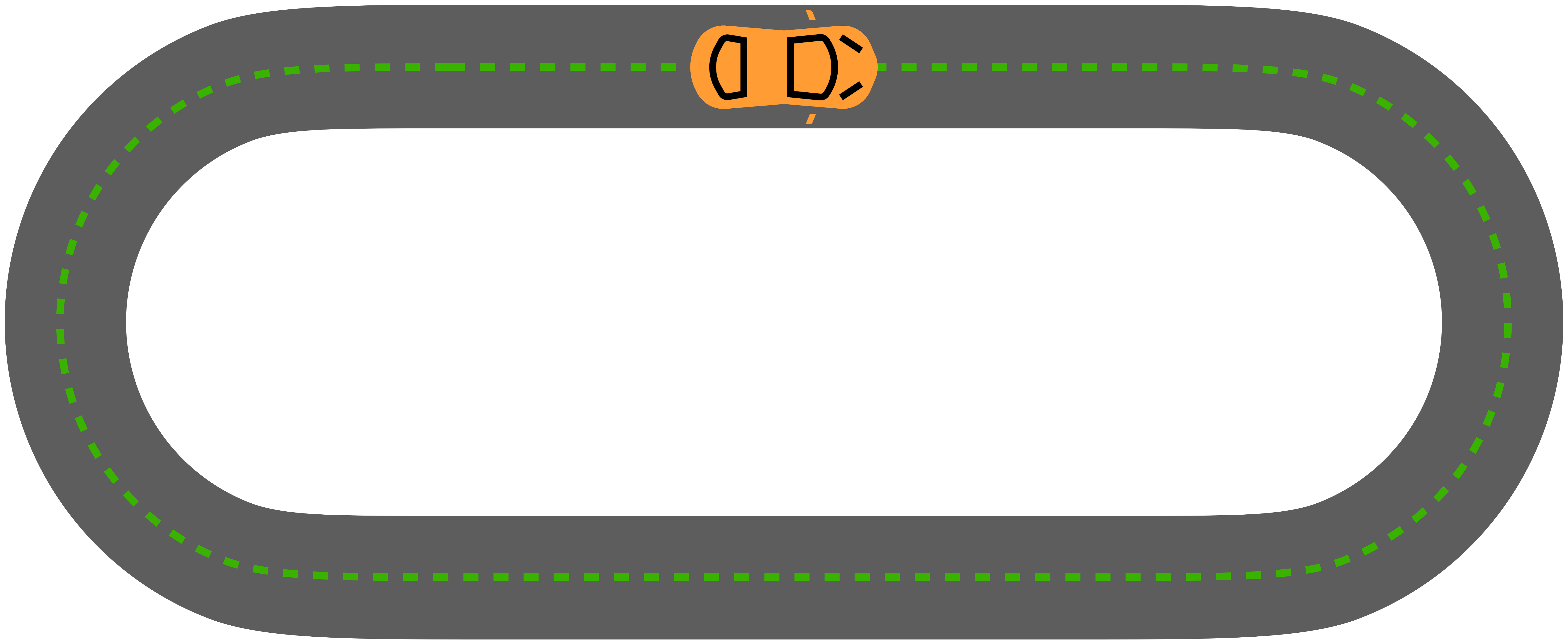








Imitation Learning



Why IL for Self-Driving?

- We can easily collect data of people driving.
- We can identify the set of things that people care about when driving, which makes it easy to design a state space.
- We have neither the exploration nor reward design problems that plague reinforcement learning.

Reward Design

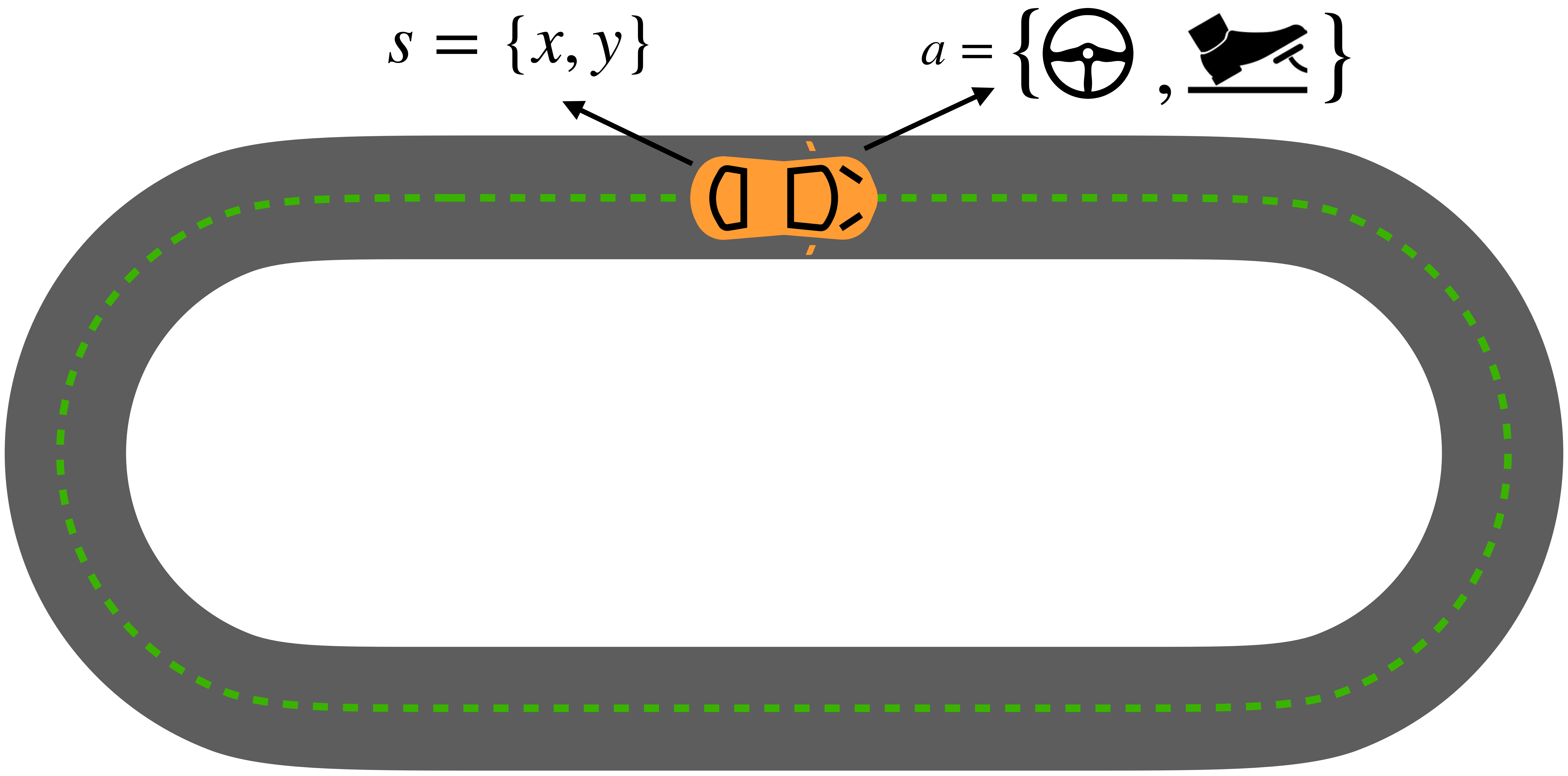
- It is very hard to write down the exact function you're optimizing when you're driving.
- Goodhardt's Law: when you feed an incorrectly specified reward function to an optimizer, bad things can happen.

Exploration

- Even if you can write down a good reward function, you then need to learn how to optimize it over the horizon

- In IL, an expert tells us what good states are so we don't need to explore as much.

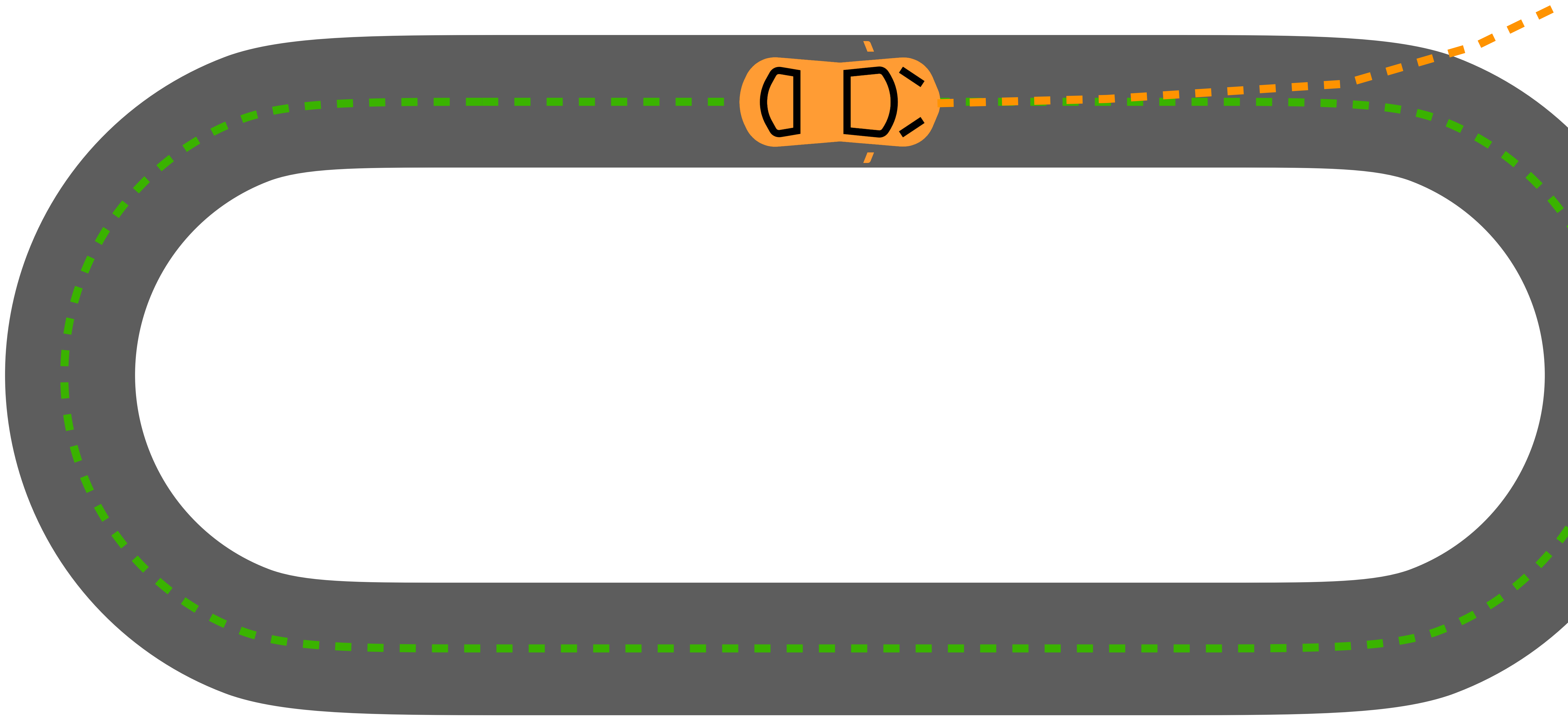
Behavioral Cloning



$$\{s_1 \ldots s_n\} \mapsto \{a_1 \ldots a_n\}$$

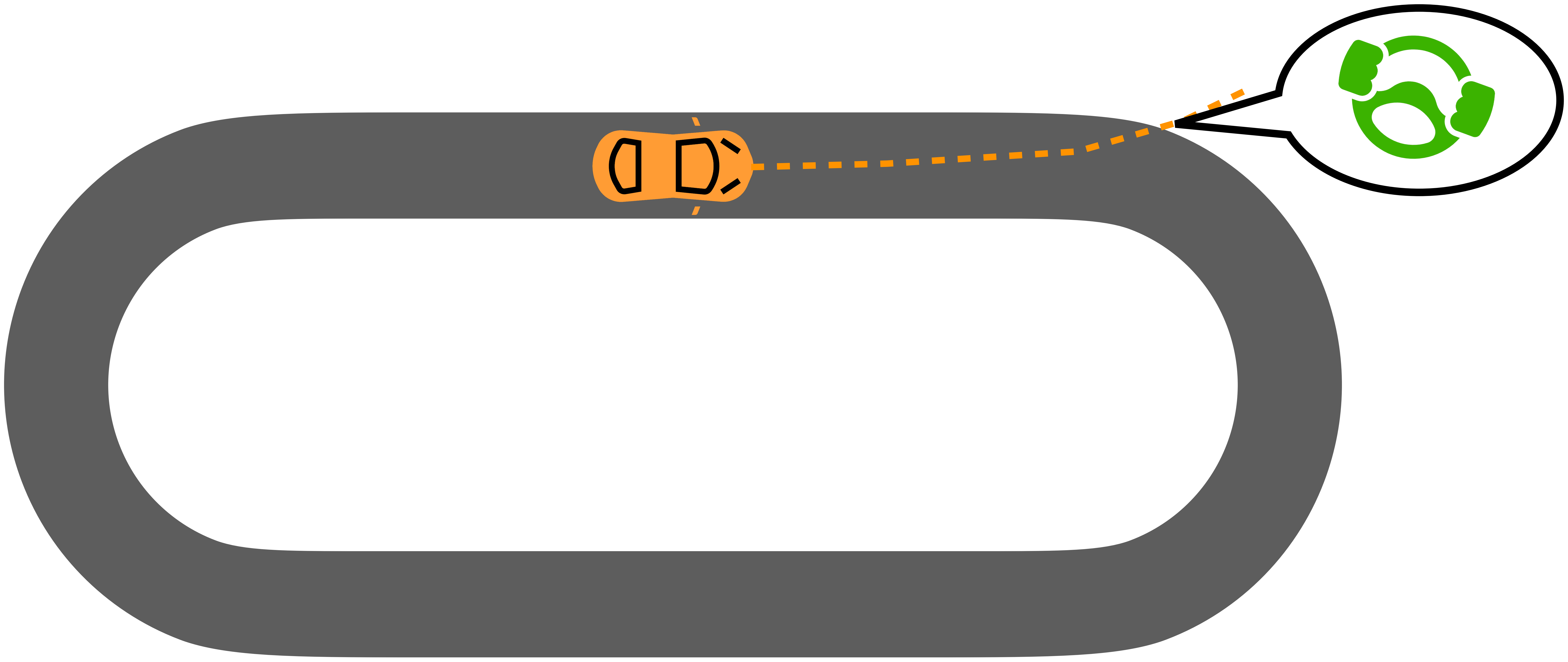
$$L(\pi) = \frac{1}{N} \sum_{i}^{N} (\pi(s_i) - a_i)^2$$

Behavioral Cloning



$$\{s_1 \ldots s_n\} \mapsto \{a_1 \ldots a_n\}$$

DAgger



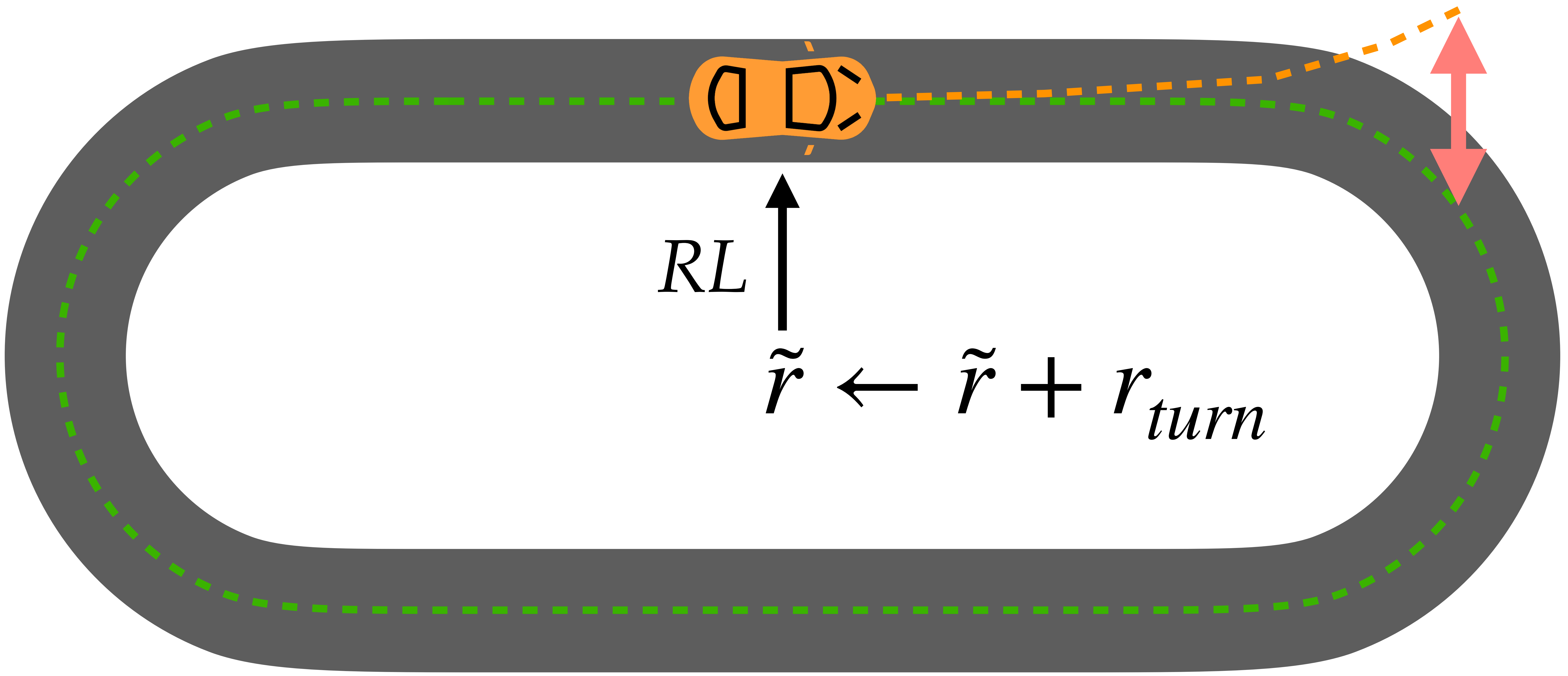
$$\{s_1 \ldots s_n\} \mapsto \{a_1 \ldots a_n\}$$

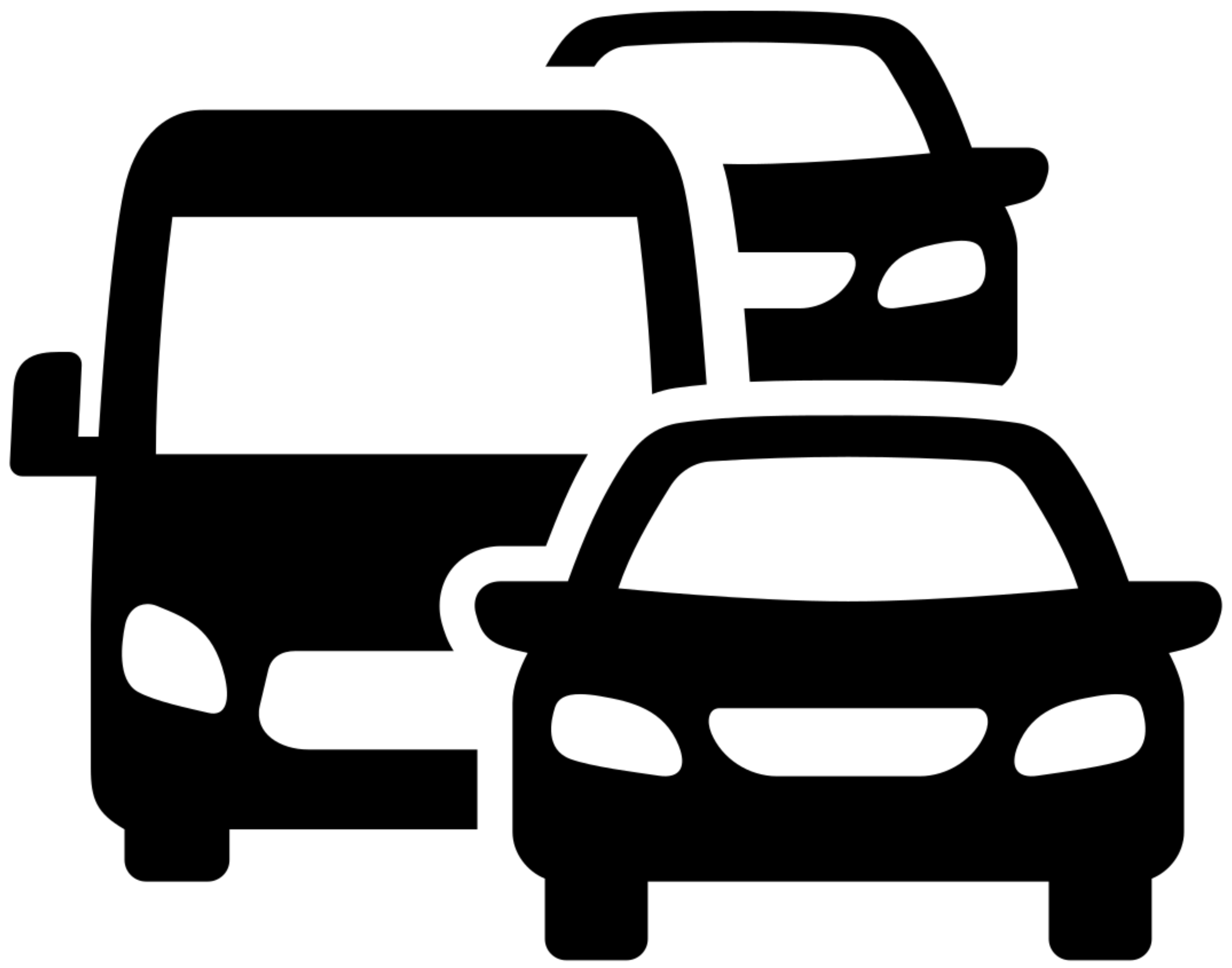
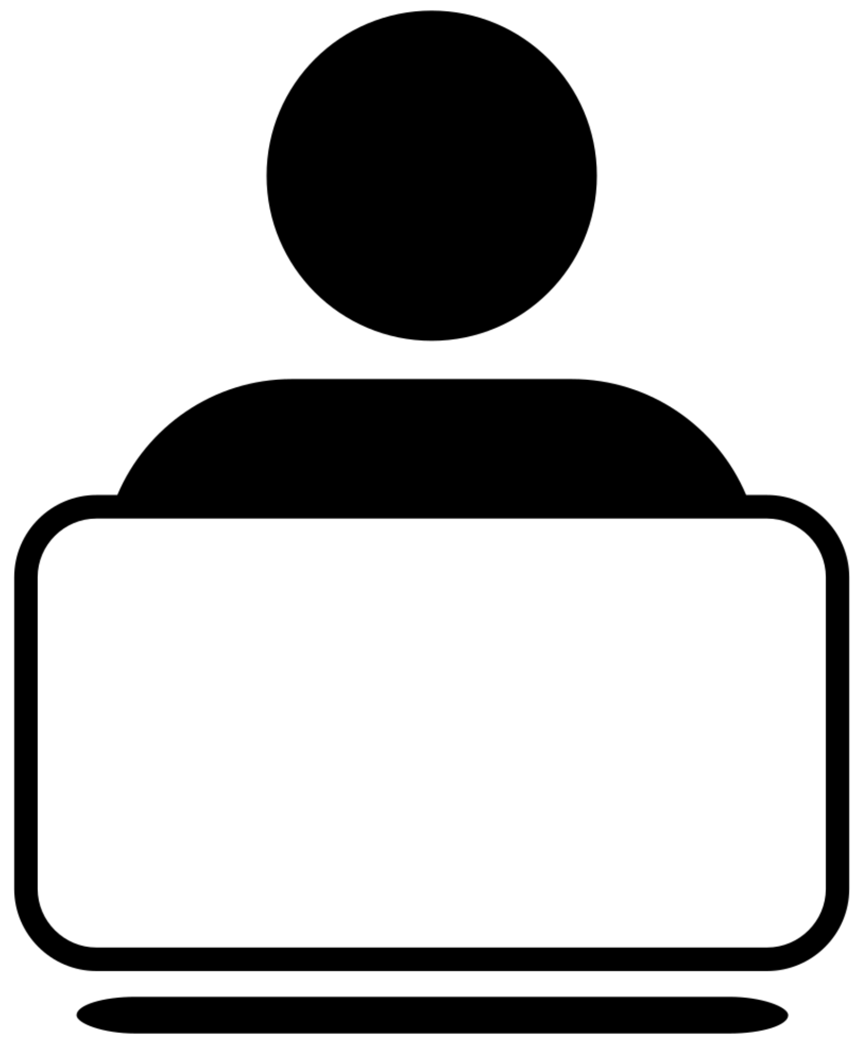
DAgger Algorithm

1. Initialize empty dataset.
2. Collect data by driving agent around.
3. Have expert label each state with correct action.
4. Append new labeled samples to dataset.
5. Retrain policy on aggregate dataset.
6. If policy unsatisfactory, go back to 2. Else, exit.

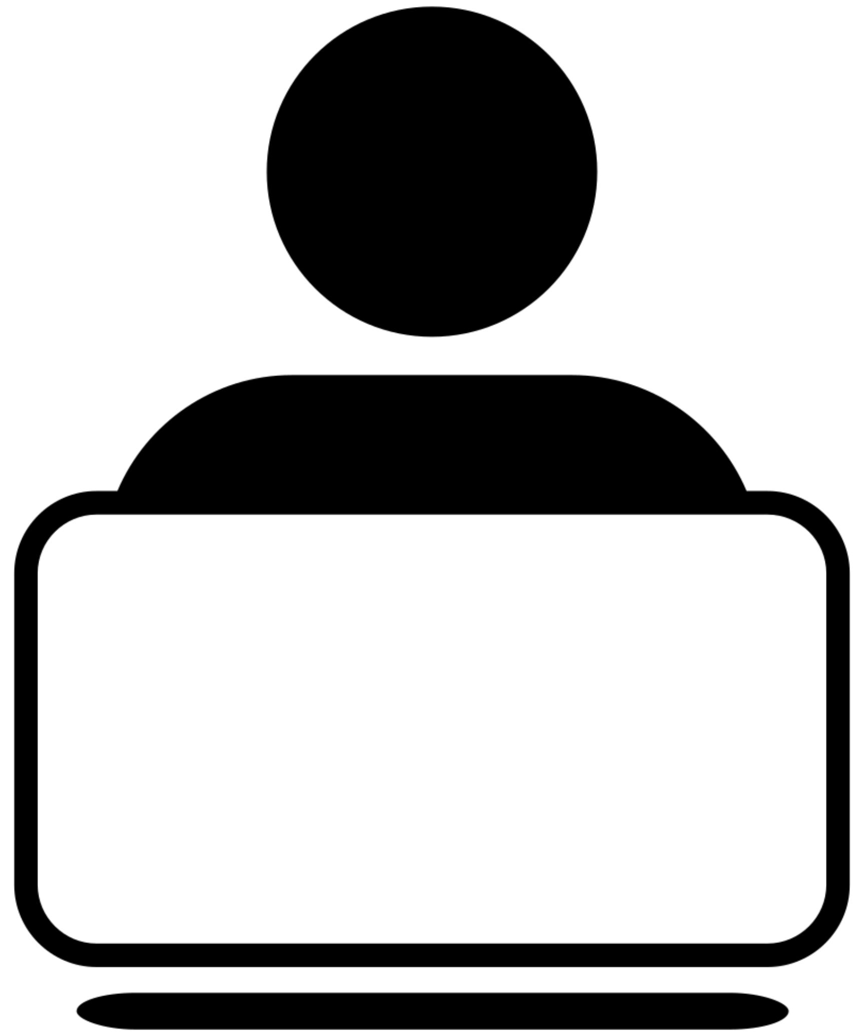
What about if you can't query the expert online?

- RL: reward function → policy
- Inverse RL: policy / demonstrations → reward function



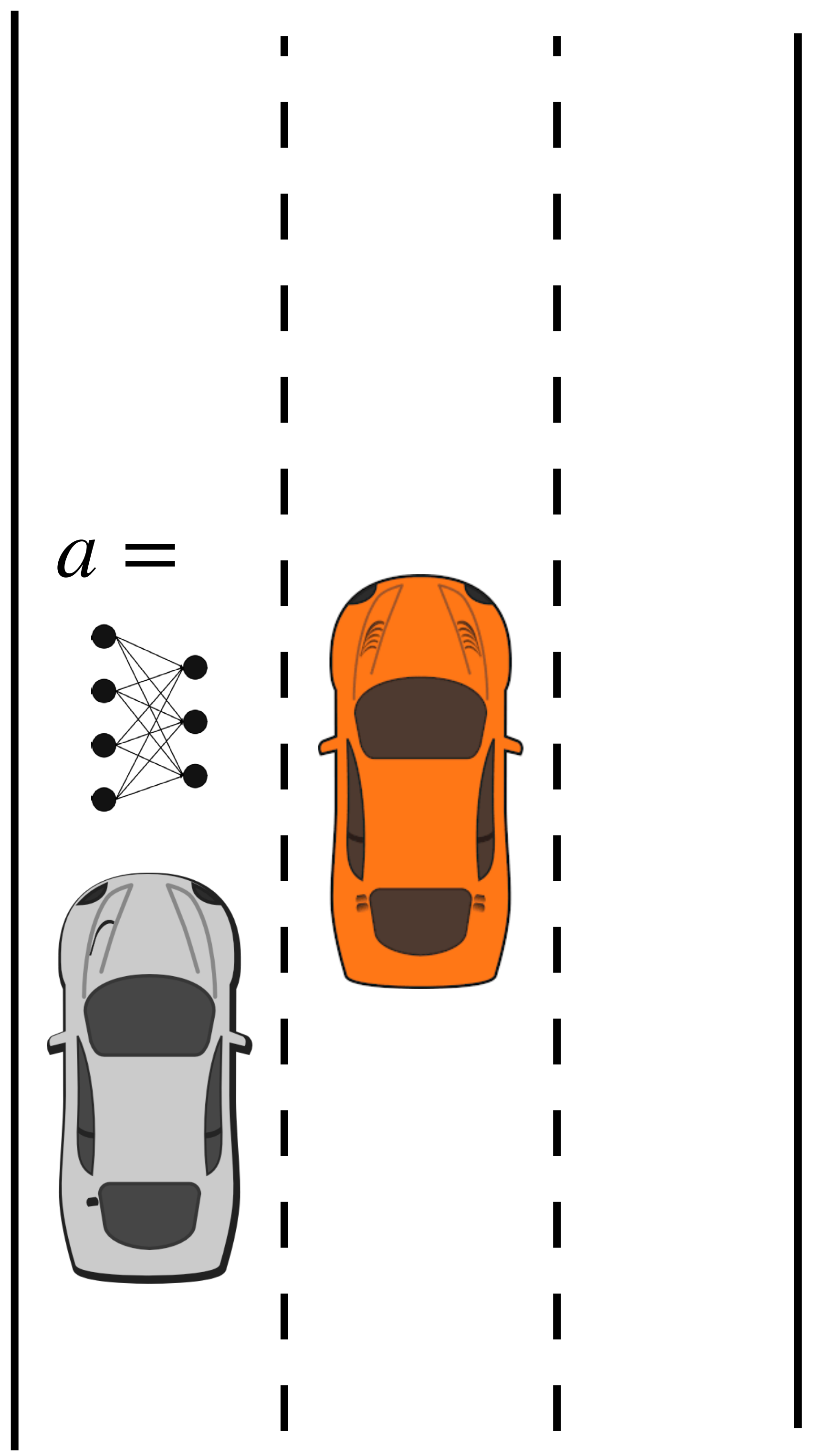








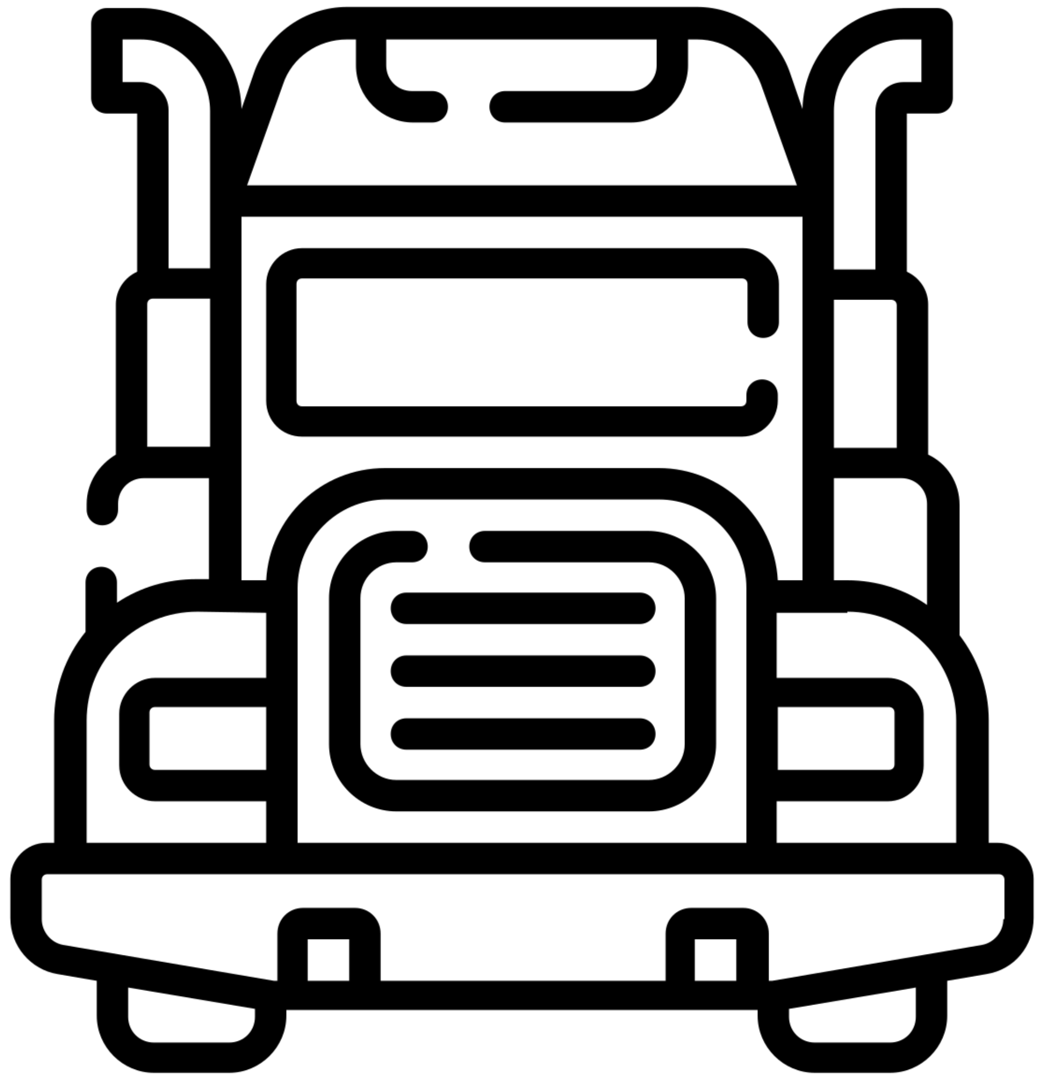
Behavior Prediction

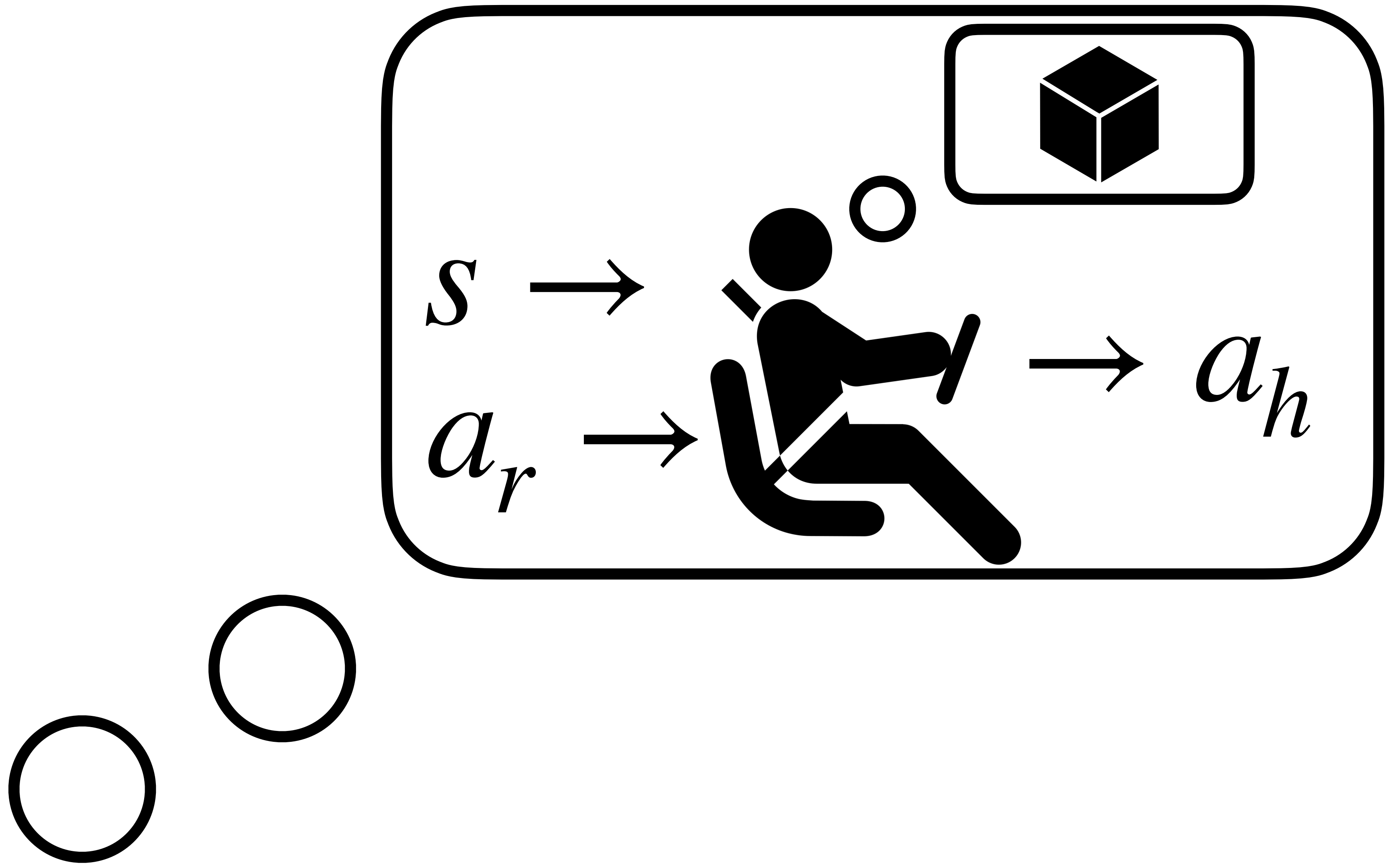
- What if we just fit a network to map state of all cars to actions of a particular car?
- Problem: your actions → actions of other cars
- Thus, if you ever change your policy, your predictive model might no longer generalize
- So, we want to fit an action-conditional predictive model.



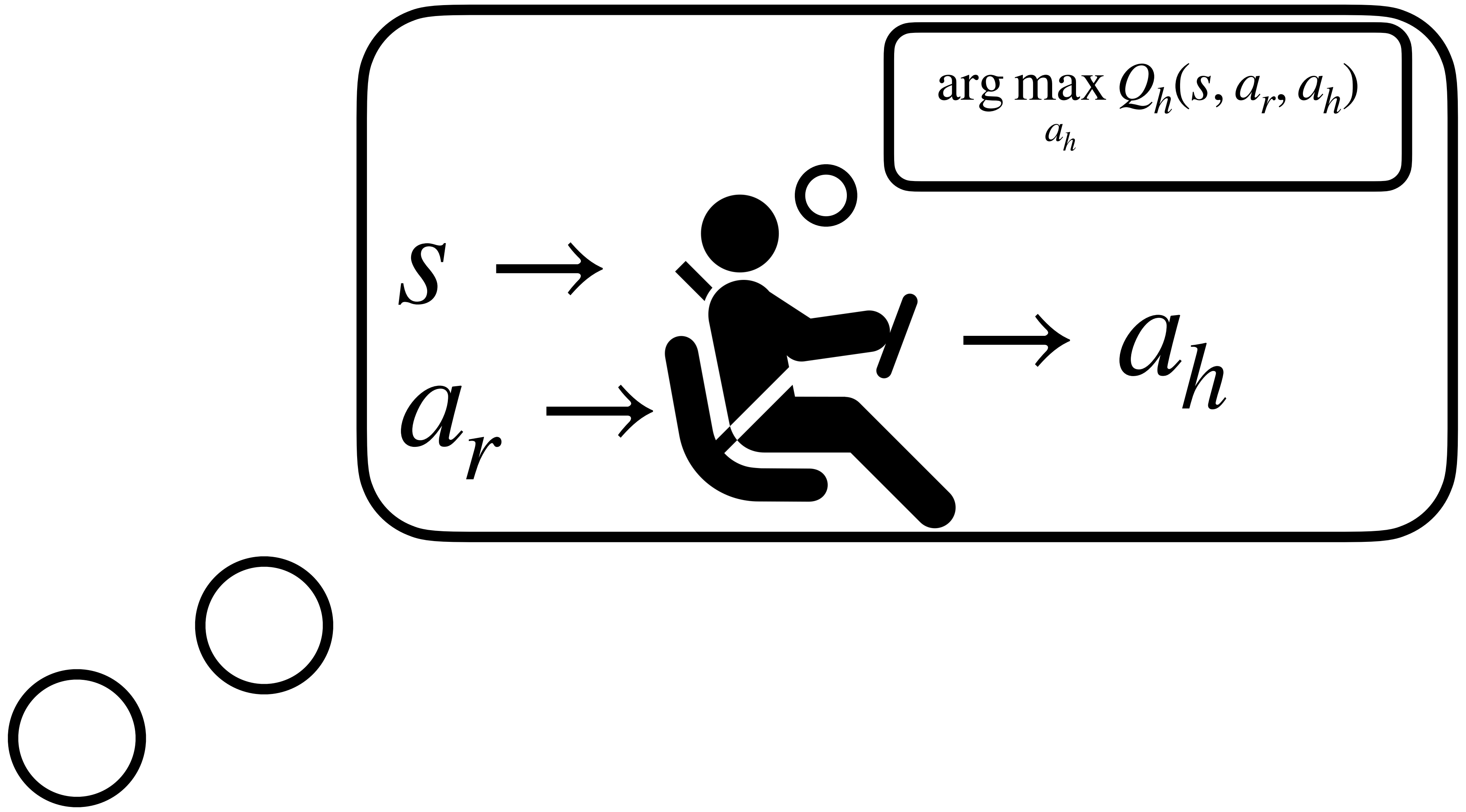
BP Approach 1: Black-Box Model

$$\max_{\pi_r} E[V_r(\pi_r, \pi_h)]$$

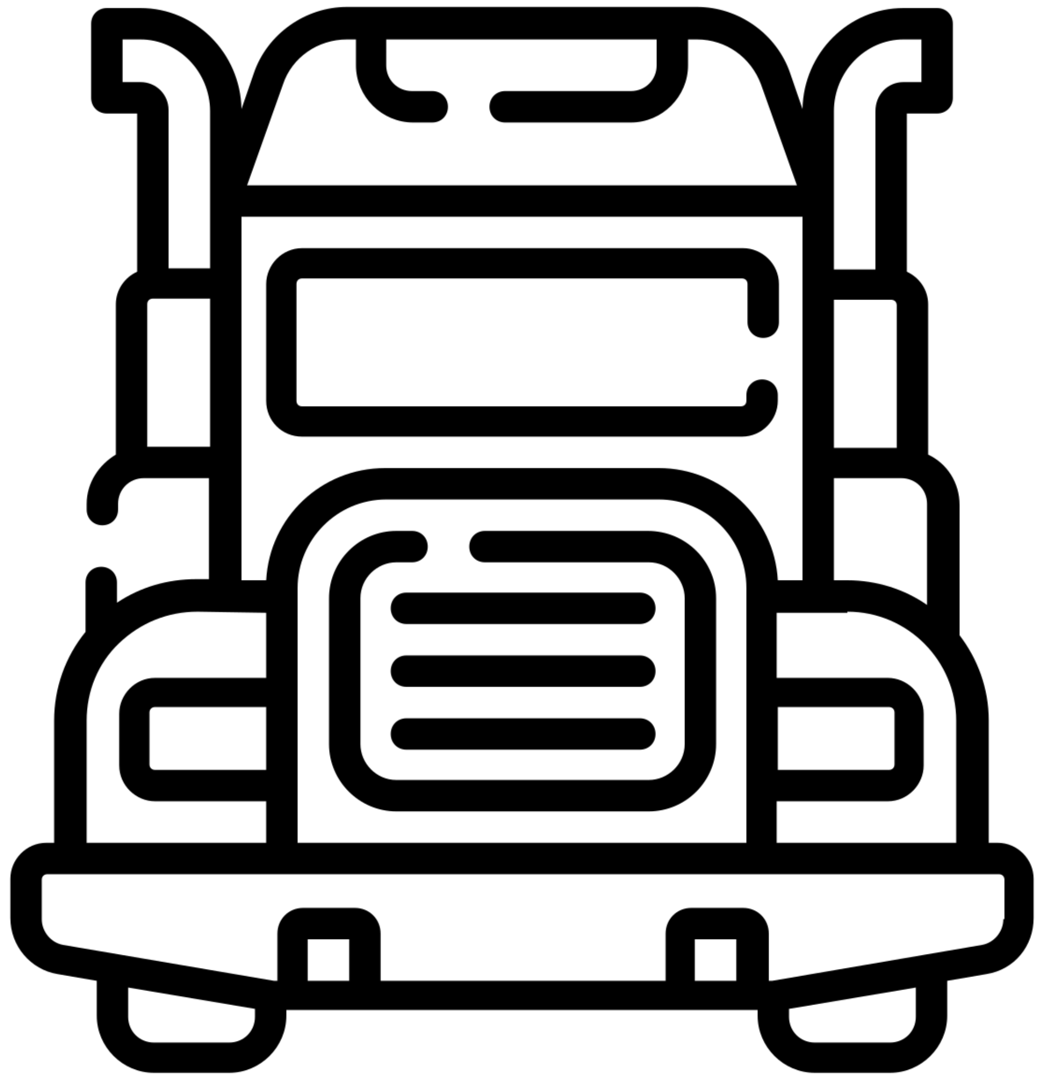


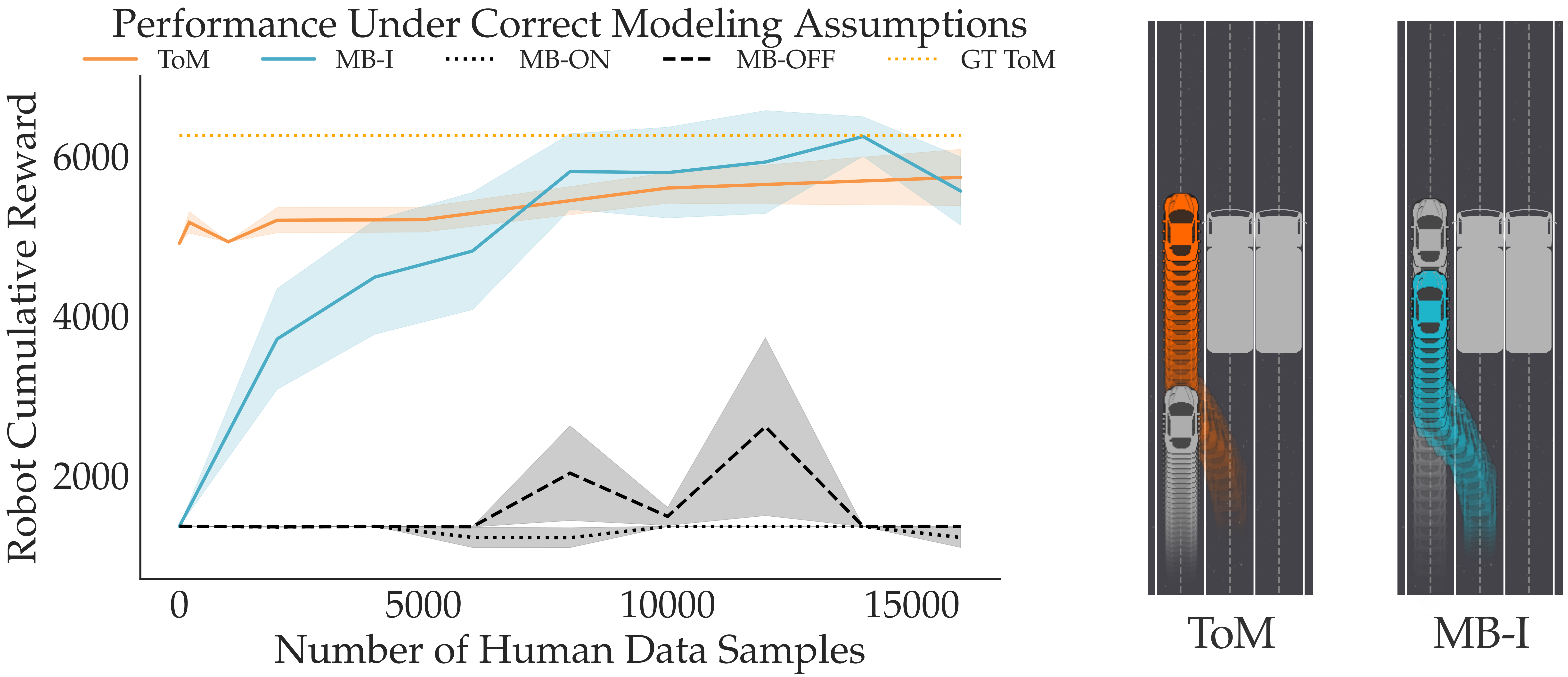


BP Approach 2: Theory-of-Mind

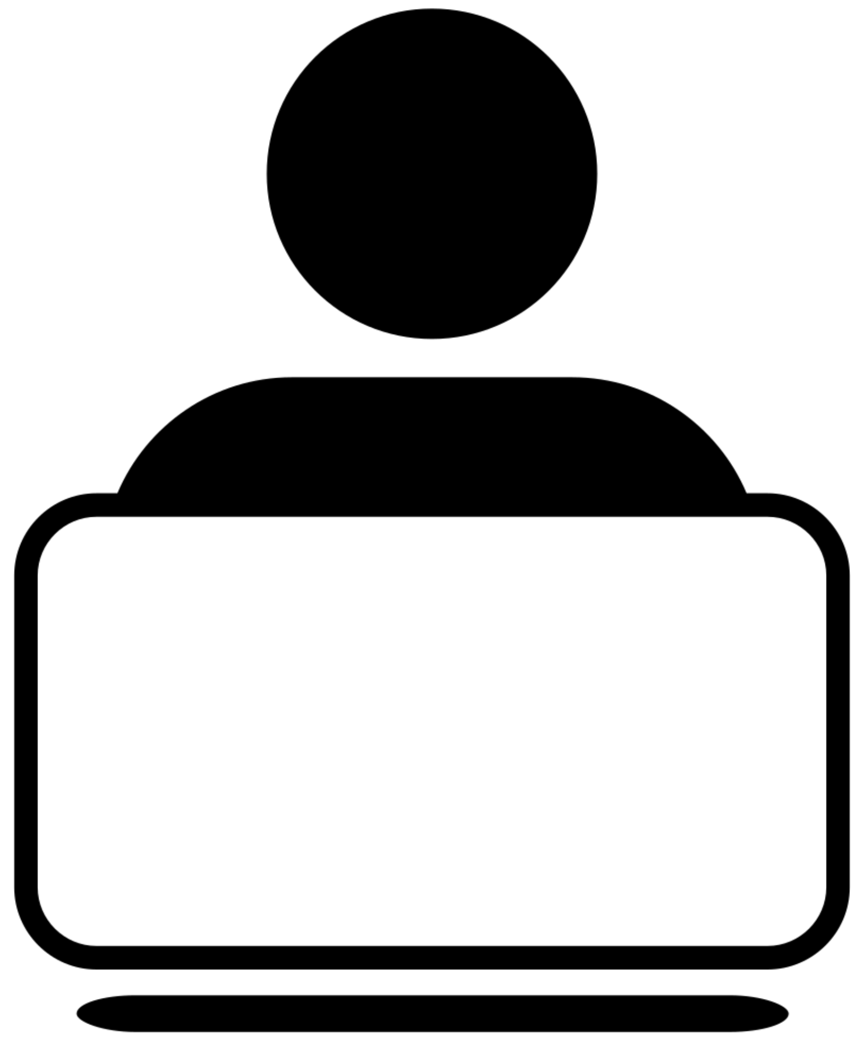


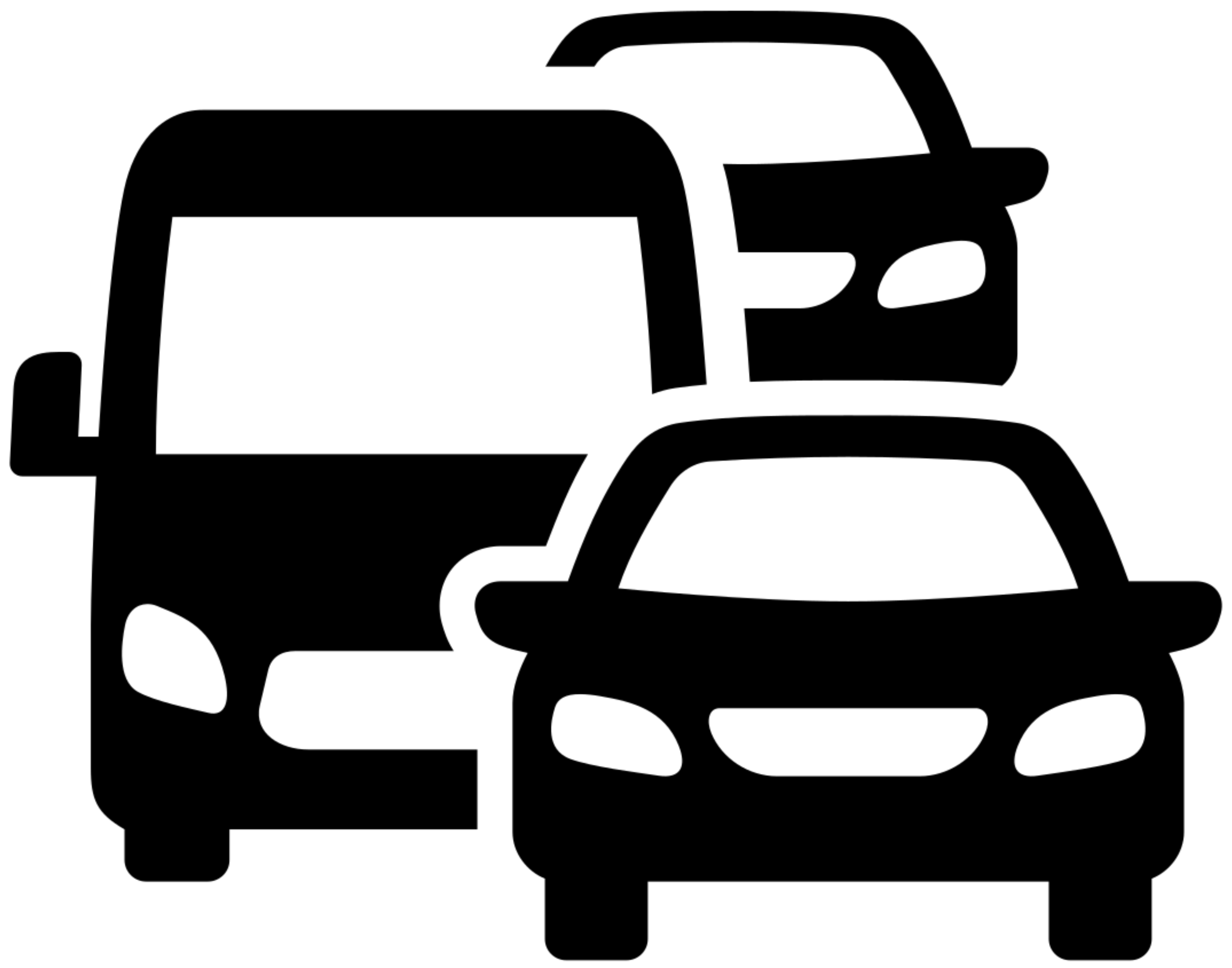
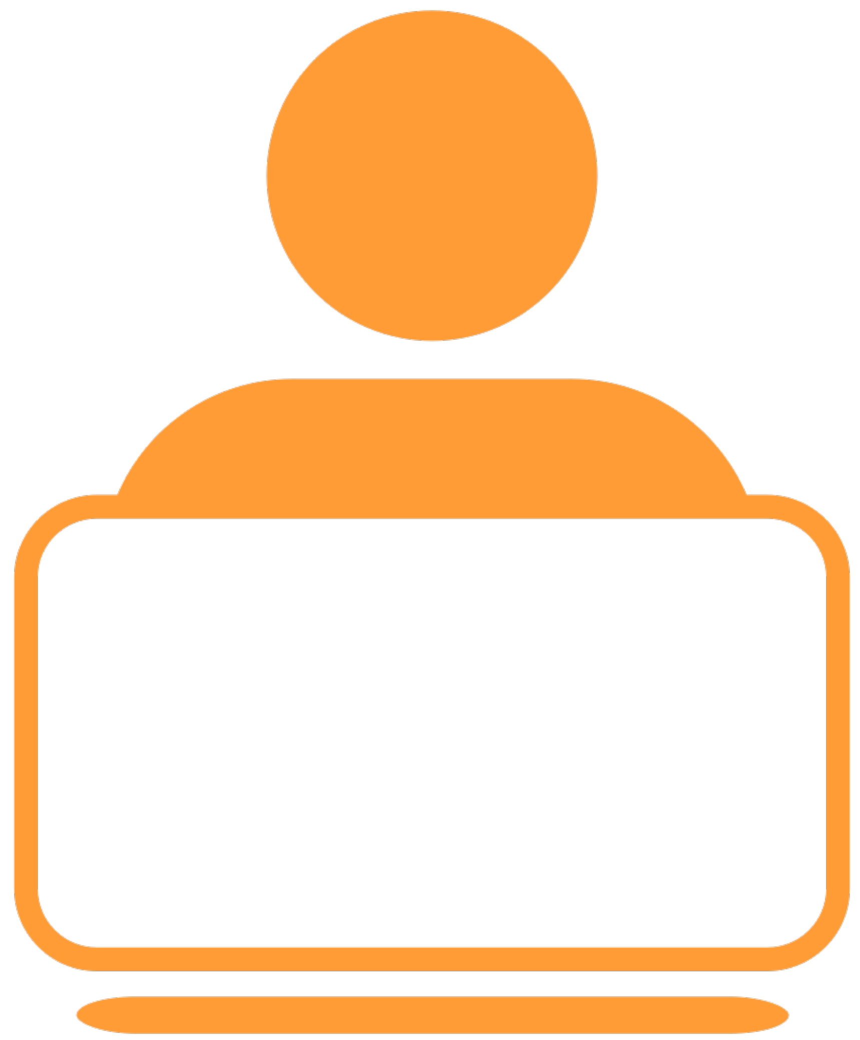
$$\max_{\pi_r} E[V_r(\pi_r, \pi_h)]$$



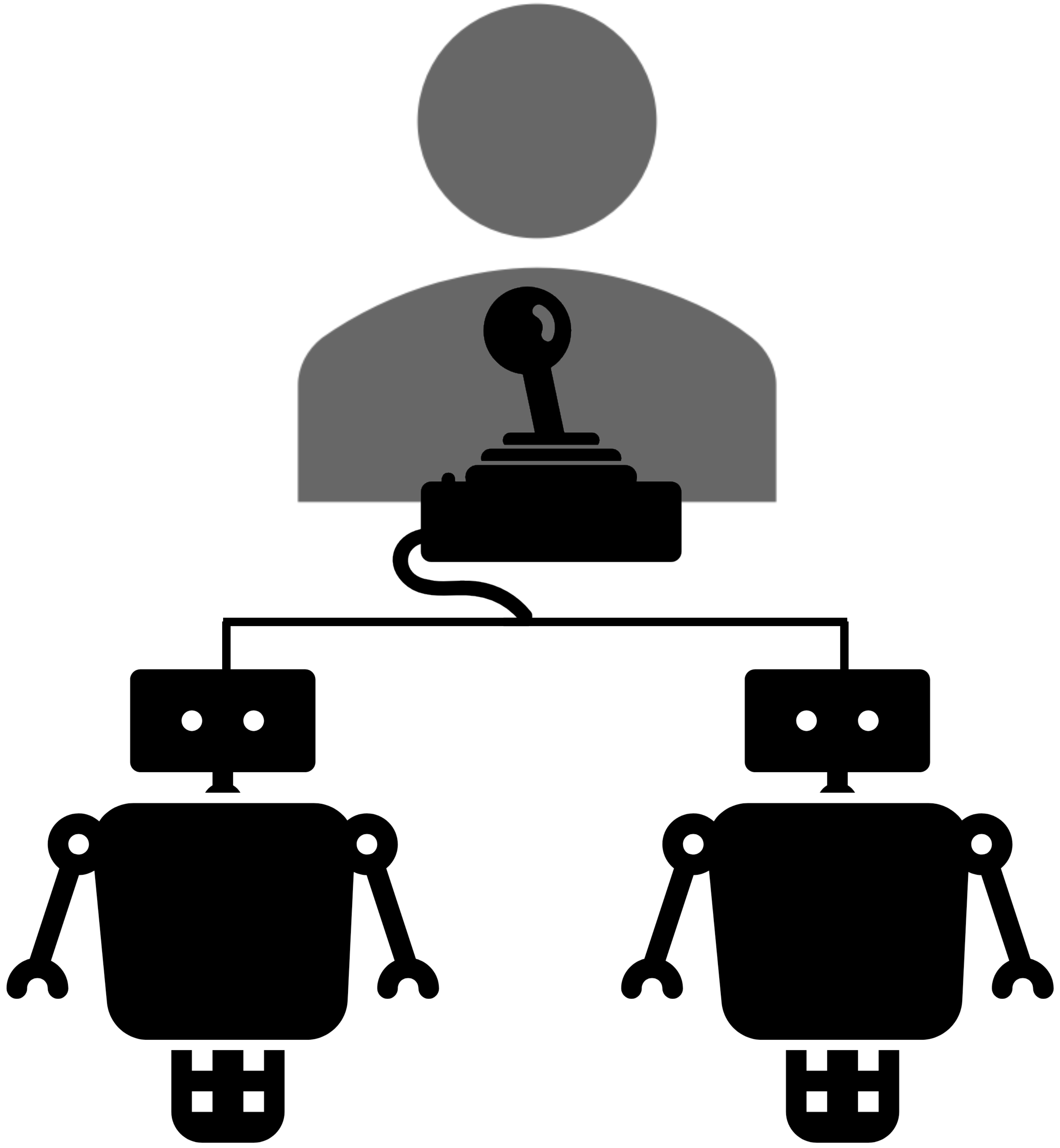

Performance Under Correct Modeling Assumptions
ToM
MB-I
MB-ON
MB-OFF
GT ToM
Robot Cumulative Reward
6000
4000
2000
0
5000
10000
15000
Number of Human Data Samples
ToM
MB-I



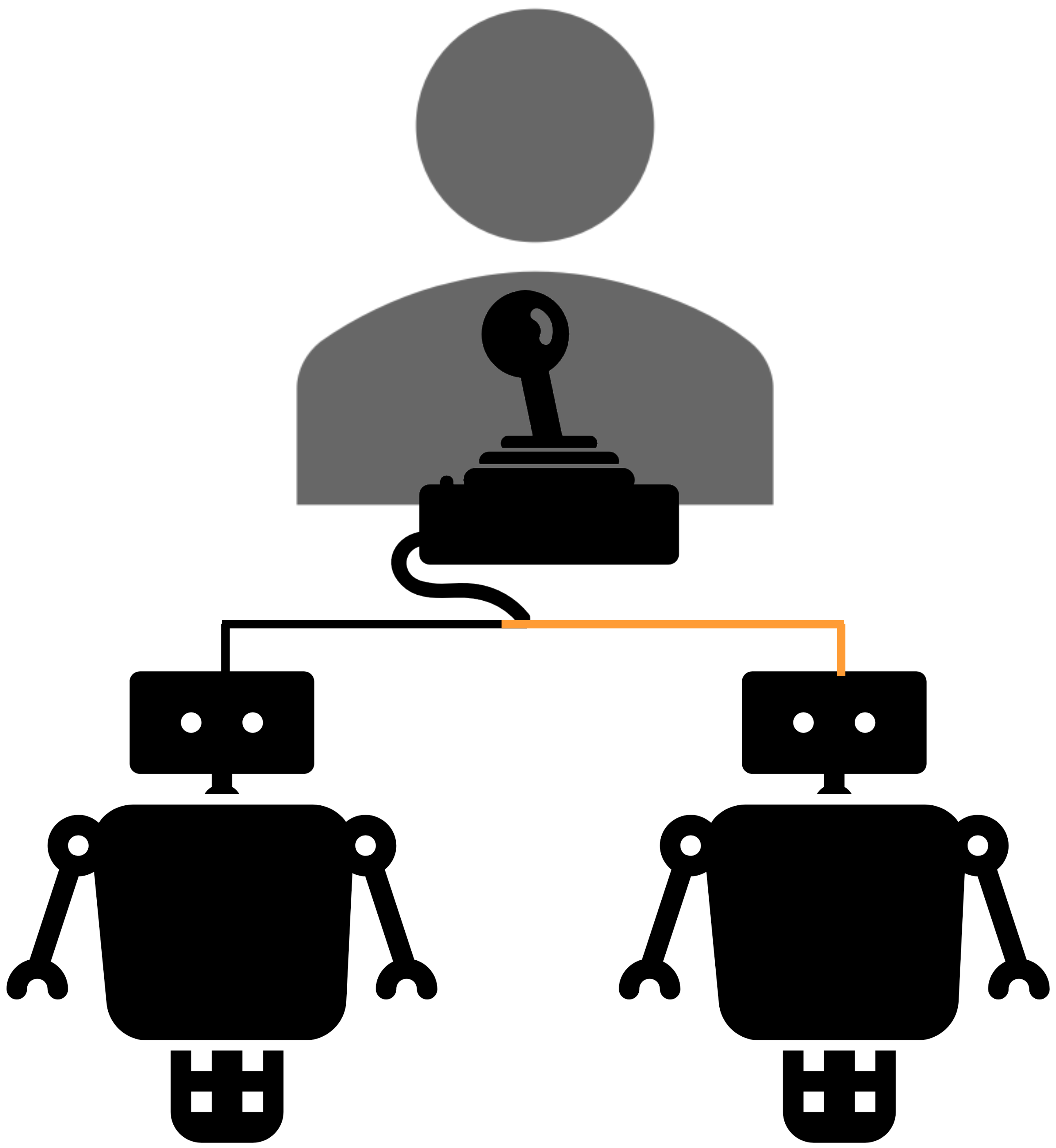


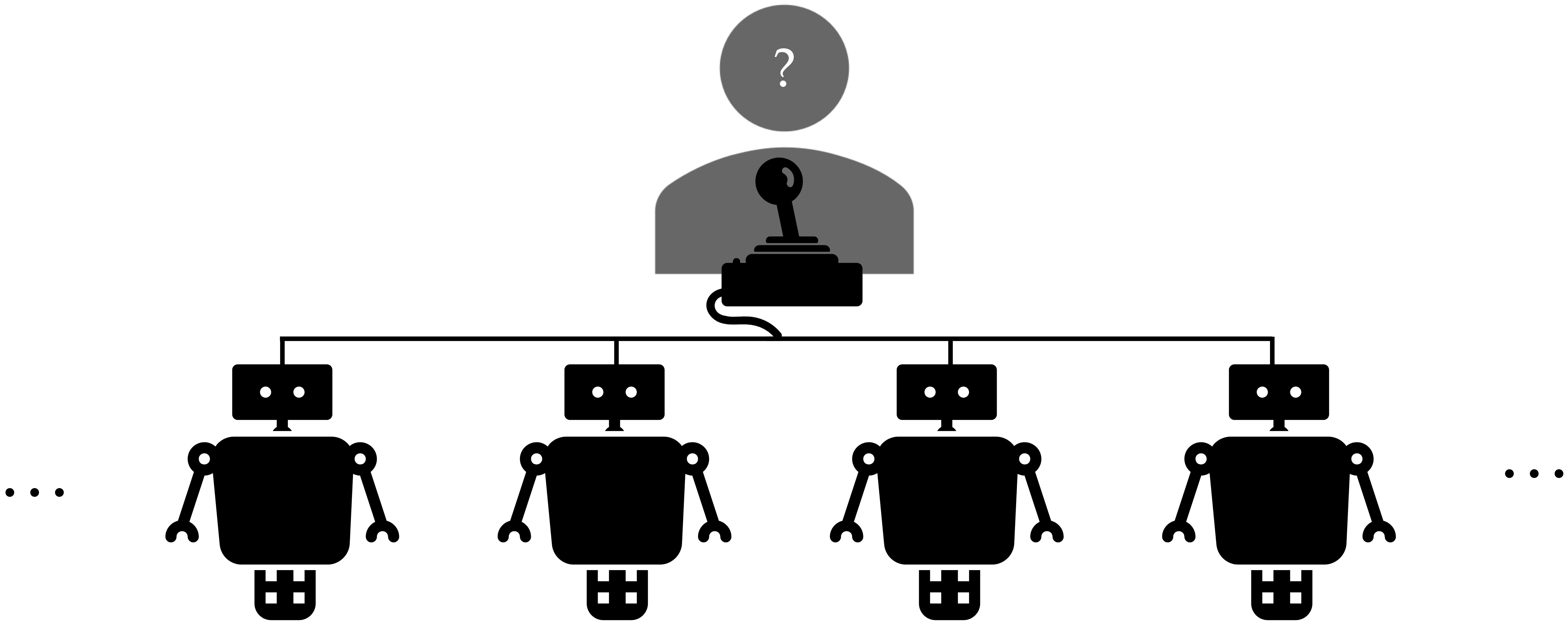
Teleoperation



Teleoperation



Teleoperation

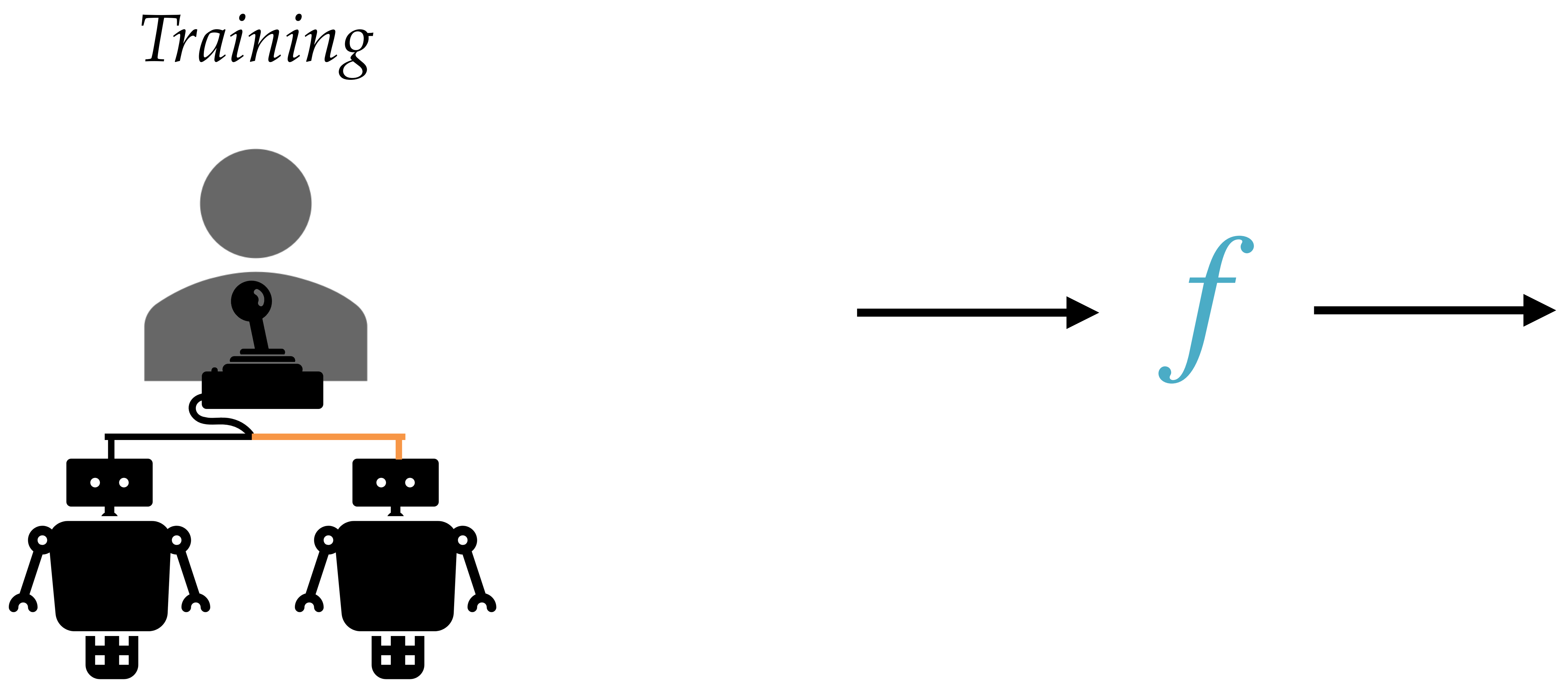


Key Insight

We can use decisions that the operator makes in easy settings with only a few robots to train a predictive model of user behavior that generalizes to challenging settings with many robots.

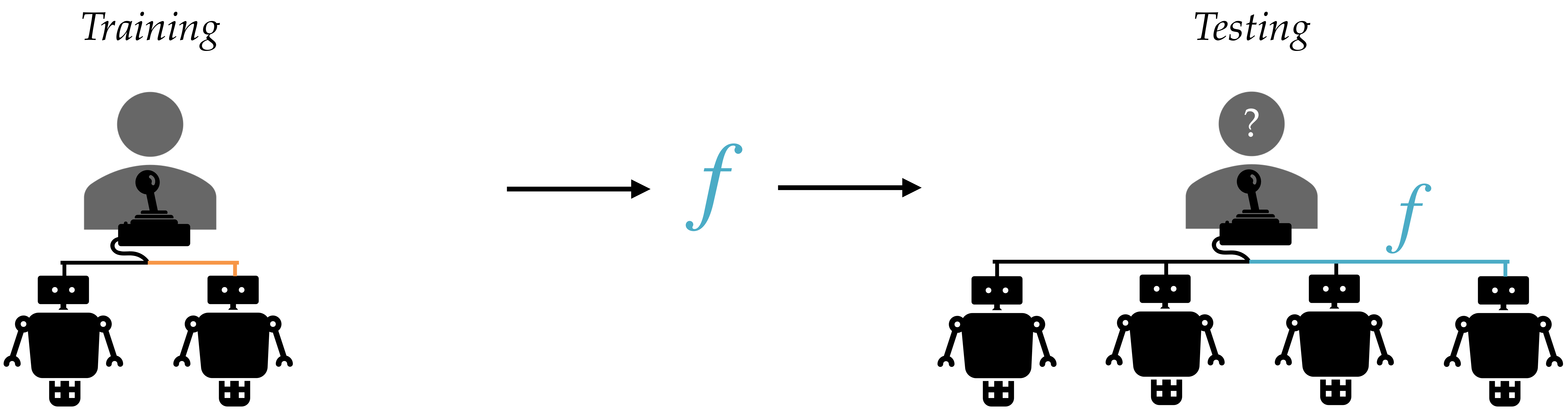
Key Insight

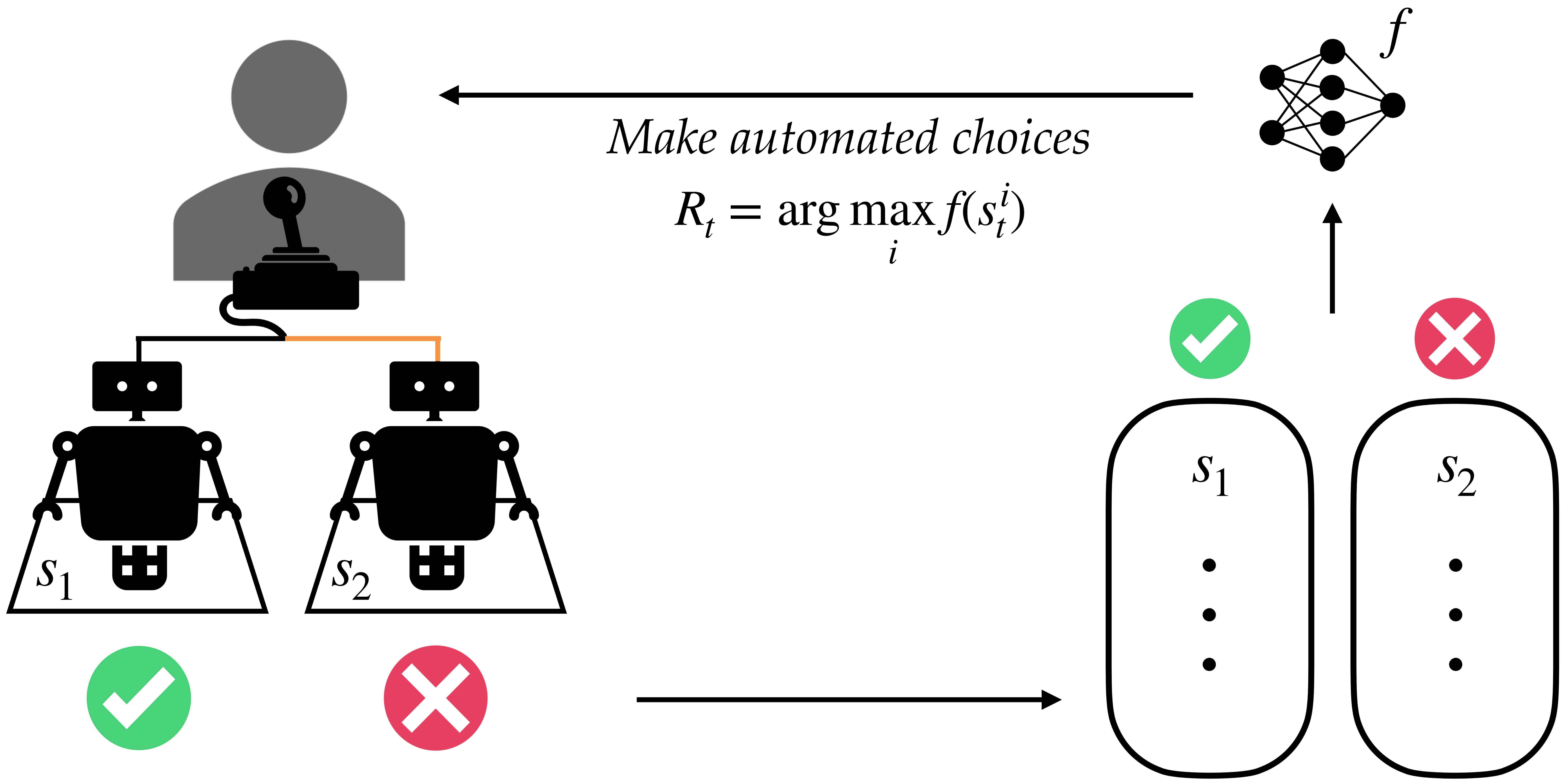
We can use decisions that the operator makes in easy settings with only a few robots to train a predictive model of user behavior that generalizes to challenging settings with many robots.



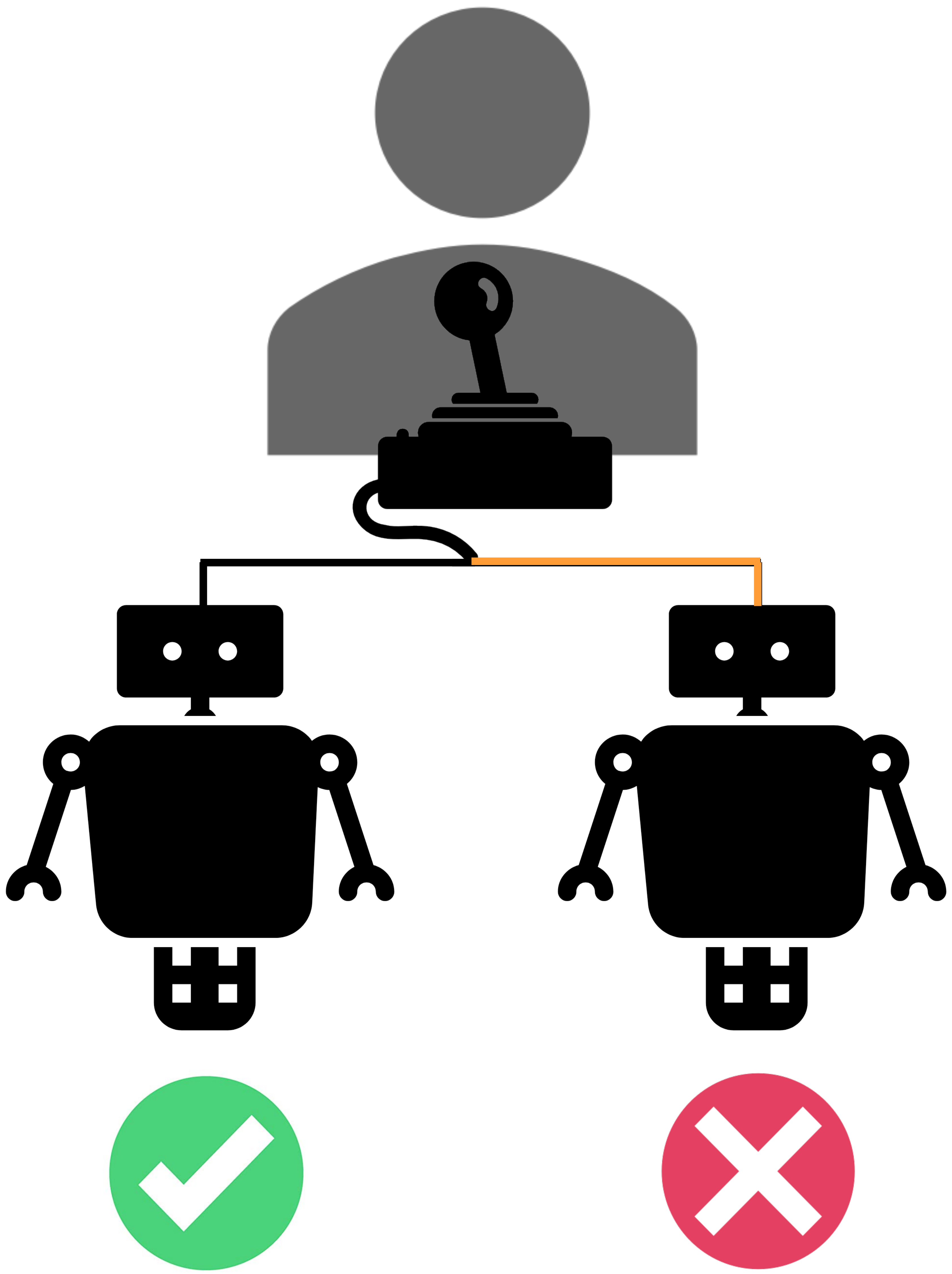
Key Insight

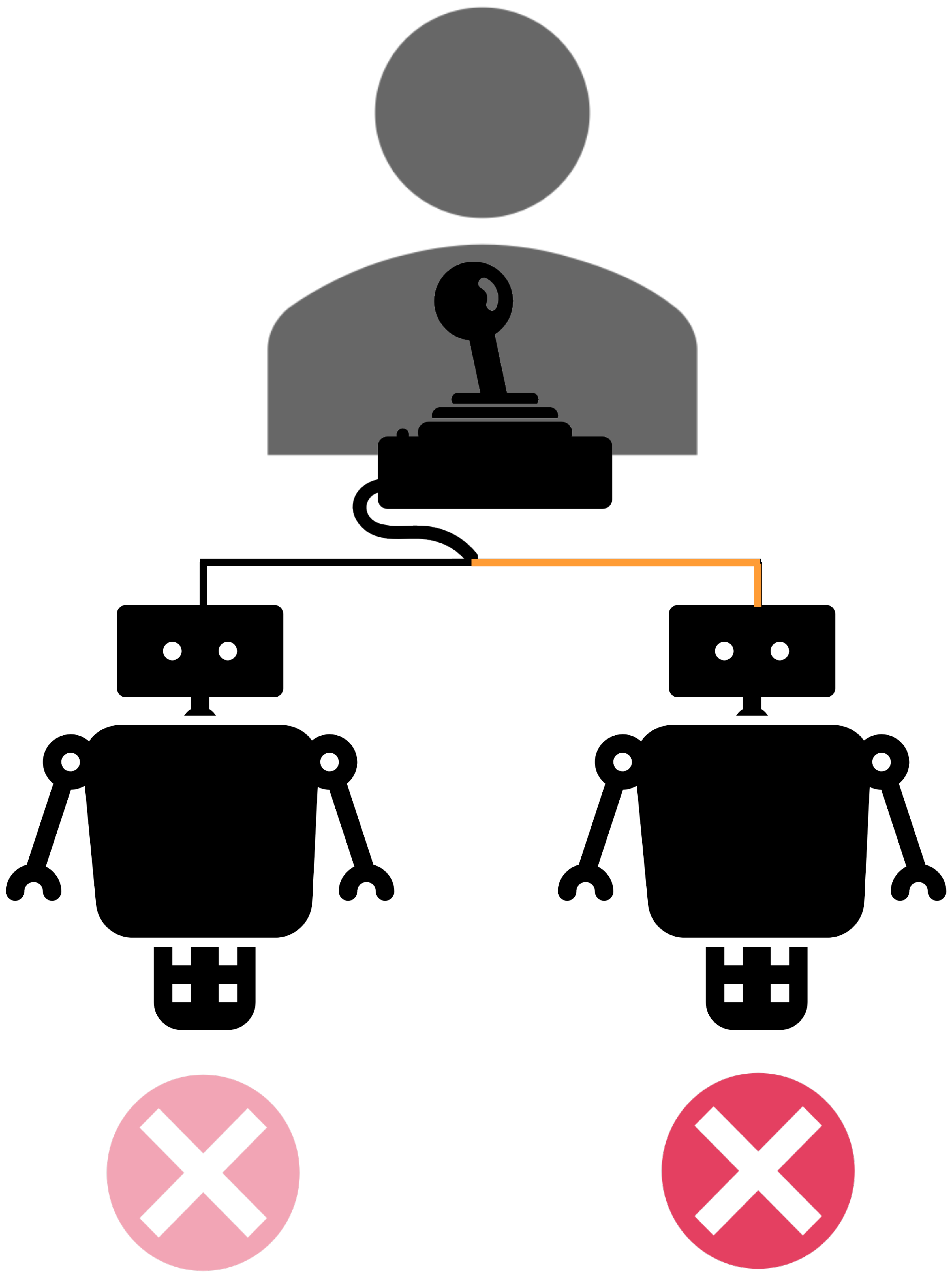
We can use decisions that the operator makes in easy settings with only a few robots to train a predictive model of user behavior that generalizes to challenging settings with many robots.

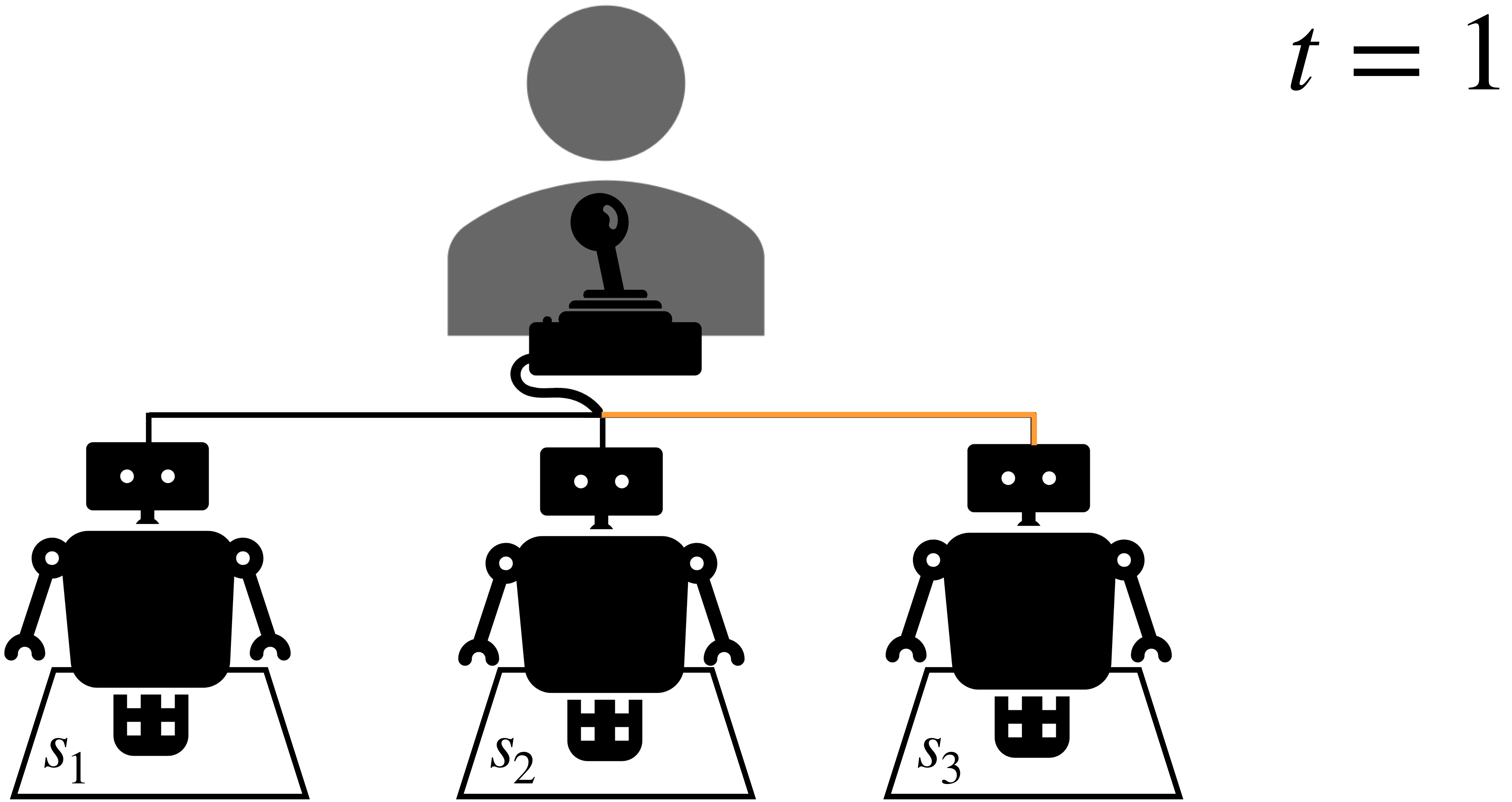




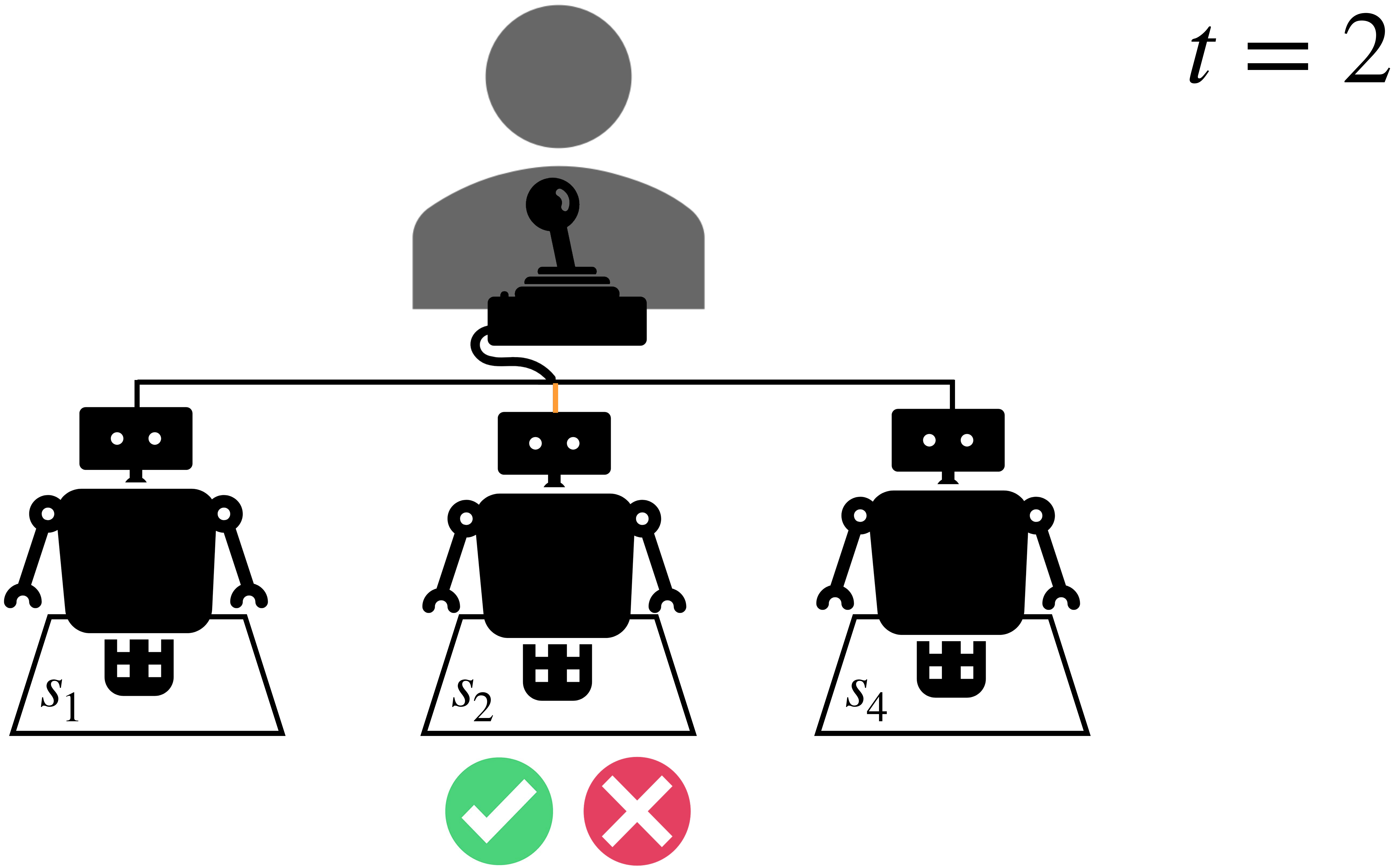
f
Make automated choices
$R_t = \arg\max_i f(s_t^i)$
s_1
s_2
s_1
s_2

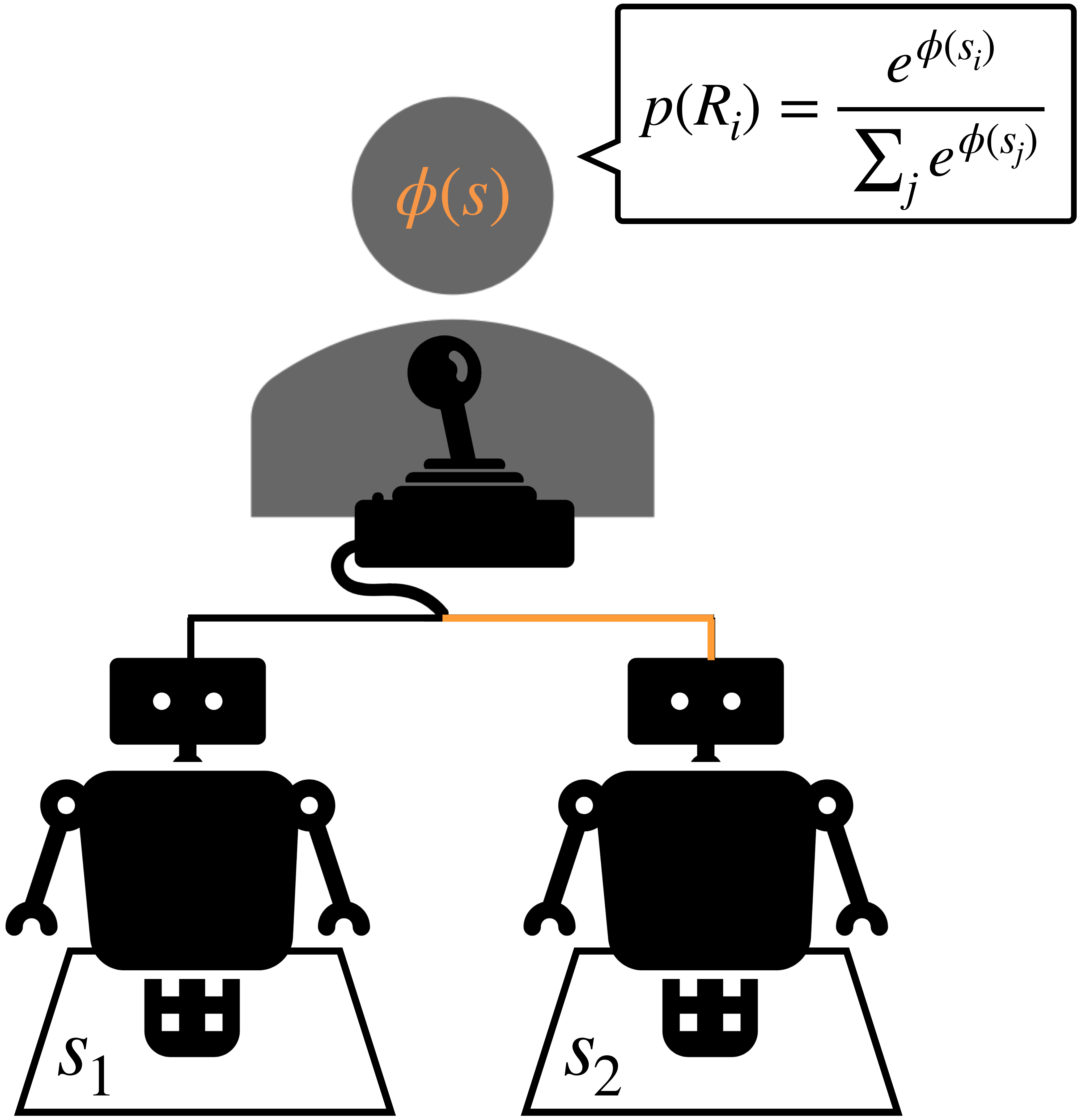






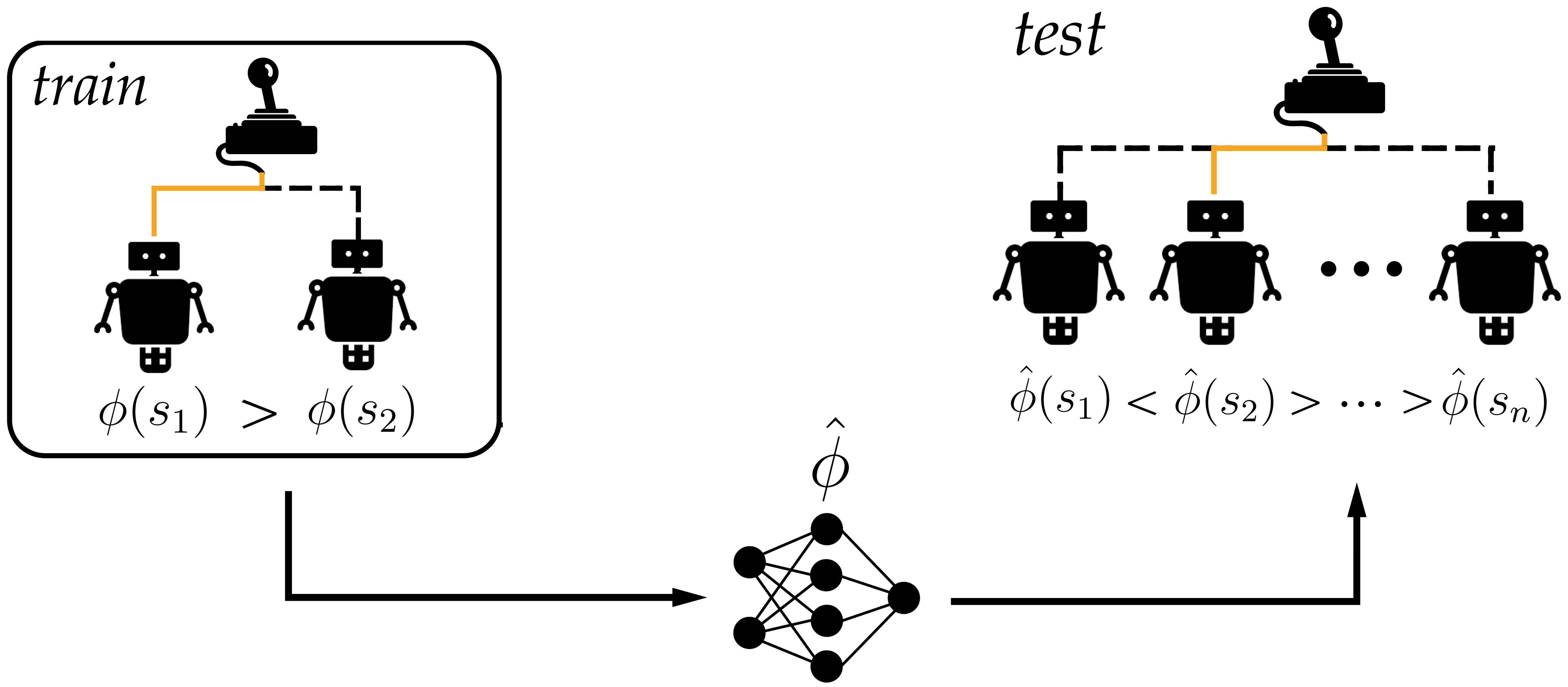
$t = 1$
s_1
s_2
s_3


$t = 2$
s_1
s_2
s_4


$p(R_i) = \frac{e^{\phi(s_i)}}{\sum_j e^{\phi(s_j)}}$
$\phi(s)$
s_1
s_2

Scaled Autonomy

Step 1: *Let user freely choose which of a few robots to teleoperate.*



Step 3: *Take the argmax over the learned function to automatically choose a robot for the user.*

Step 2: *Train a network to mimic user choices by maximizing the likelihood of the demonstrated choices under the Luce model.*

Questions?