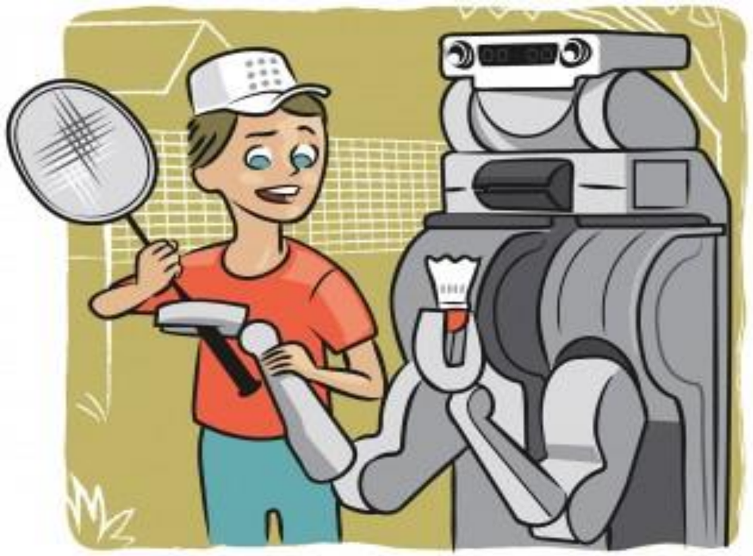
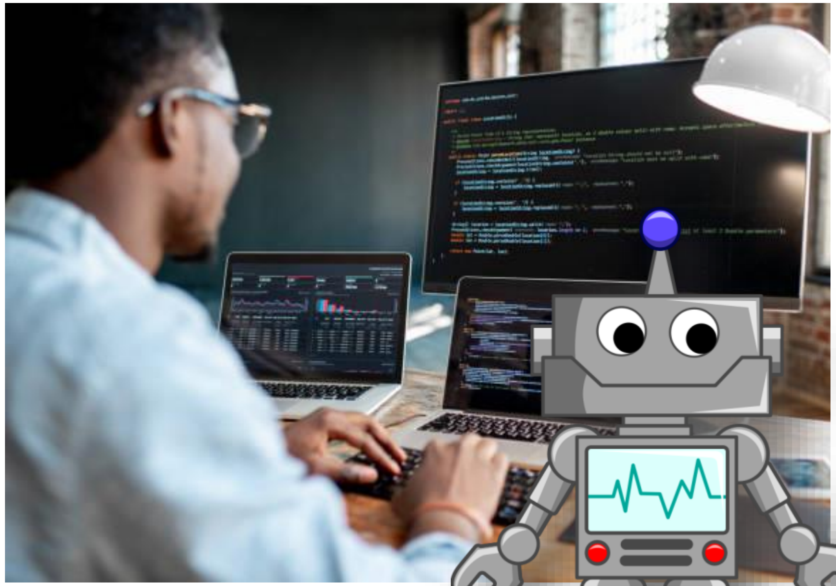
Behavioral Cloning and Interactive Imitation Learning



Instructor: Daniel Brown

[Some slides adapted from Sergey Levine (CS 285) and Alina Vereshchaka (CSE4/510)]



Brief Machine Learning Refresher

There are roughly 3 main branches of machine learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

ML

Data mining

This class

$D = \{ (x, y) \ldots \}$

$x \rightarrow \boxed{\text{model}} \rightarrow y$

$f(x) = y$

Supervised Learning

- **Setting/Assumptions**: In supervised learning, the model is trained on labeled data, where the input data is paired with the correct output (i.e., the "ground truth").
- **Goal**: To learn a mapping from inputs to outputs so that the model can predict the output for new, unseen inputs.
- **Common Use Cases**:
  - Classification (e.g., spam email detection, image recognition). Discrete # of labels
  - Regression (e.g., predicting house prices, stock market trends). Continuous output
- **Example models**:
  - Linear regression, decision trees, support vector machines, and neural networks.

$\vec{x} \in \mathbb{R}^2 \quad y = w_0 + w_1 x_1 + w_2 x_2$

$C = \#$ classes

Classification

$y = $ cat $= 1$

$x = $ 

$y = $ dog $= 2$

$x = $ 

$C = 2$

model $\rightarrow$ logits $\begin{bmatrix} l_1 \\ l_2 \end{bmatrix}$

$\hat{y}_1 = Pr(y = \text{cat}) = \dfrac{\exp(l_1)}{\exp(l_1) + \exp(l_2)}$

minimize this

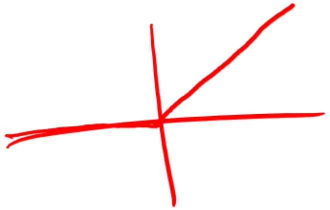
True label

$$\text{Cross-Entropy Loss} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

$= -y_1 \log(\hat{y}_1) + \cancel{y_2 \log(\hat{y}_2)}$ 0

$= -1 \cdot \log(\hat{y}_1)$

PyTorch Example



```
import torch.nn as nn
import torch.optim as optim

class ClassificationNetwork(nn.Module):
    def __init__(self, input_dim, num_classes):
        super(ClassificationNetwork, self).__init__()
        self.fc1= nn.Linear(input_dim, num_classes) = 3

    def forward(self, x):
        return self.fc(x)   self.fc2(self.relu(self.fc1(x)))

model = ClassificationNetwork(input_dim, num_classes)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

for epoch in range(num_epochs):
    for inputs, labels in dataloader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

3 #'s describing house

West Coast, Middle, East Coast

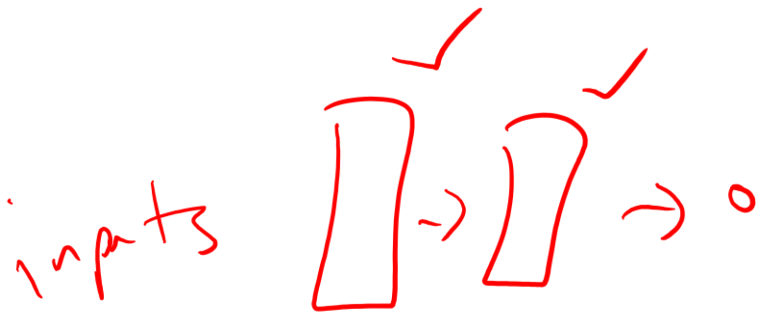
input output

$$w_i \leftarrow w_i - \frac{\partial L}{\partial w_i} \cdot \alpha$$

self.fc2 = nn.Linear(x, y)

self.relu = nn.ReLU()

Regression



pred true

$$\text{MSE Loss} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

PyTorch Example

```
import torch.nn as nn
import torch.optim as optim

class ClassificationNetwork(nn.Module):
    def __init__(self, input_dim, num_classes):
        super(ClassificationNetwork, self).__init__()
        self.fc = nn.Linear(input_dim, num_classes)

    def forward(self, x):
        return self.fc(x)

model = ClassificationNetwork(input_dim, num_classes)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

for epoch in range(num_epochs):
    for inputs, labels in dataloader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

Reg

1

MSE

ReLu

$$\text{output}_1 = (x^T w_1 + b_2)$$

$$\hat{y} = \text{output}_1^T w_2 + b_2$$

Unsupervised Learning

- **Setting/Assumptions**: In unsupervised learning, the model is trained on data without labeled outputs. It seeks to find patterns, structures, or relationships in the data. No “ground truth” labels.
- **Goal**: To explore the data and identify meaningful clusters, associations, or representations.
- **Common Use Cases**:
  - Clustering (e.g., customer segmentation).
  - Dimensionality reduction (e.g., PCA for visualization).
  - Anomaly detection (e.g., fraud detection).
- **Example models**:
  - K-means clustering, hierarchical clustering, and autoencoders.

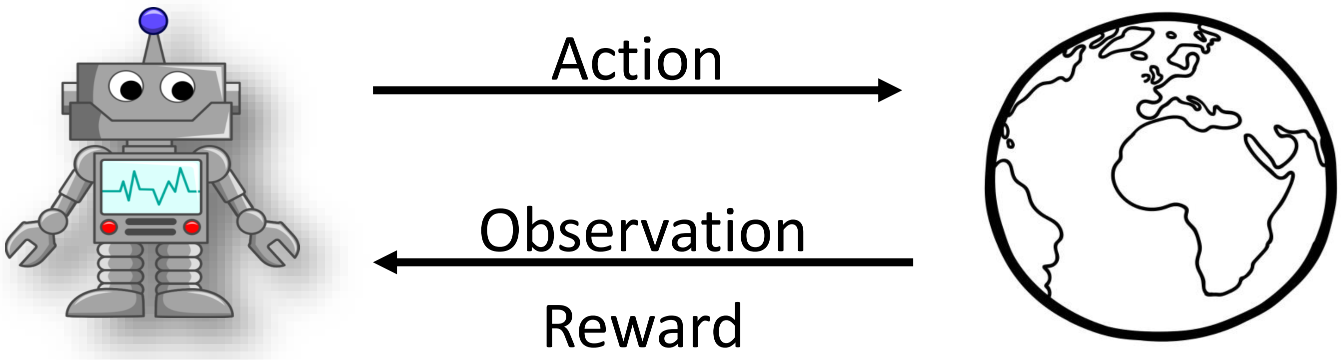
Reinforcement Learning

- **Setting/Assumptions**: Reinforcement learning (RL) involves training an agent to make decisions by interacting with an environment. The agent learns through trial and error (receiving rewards and penalties), optimizing its behavior to maximize cumulative rewards.
- **Goal**: To learn a policy that maps states of the environment to actions that achieve the highest reward.
- **Common Use Cases**:
  - Game-playing AI (e.g., AlphaGo, chess-playing bots).
  - Robotics (e.g., autonomous navigation).
  - Dynamic resource allocation (e.g., in networking or traffic management).

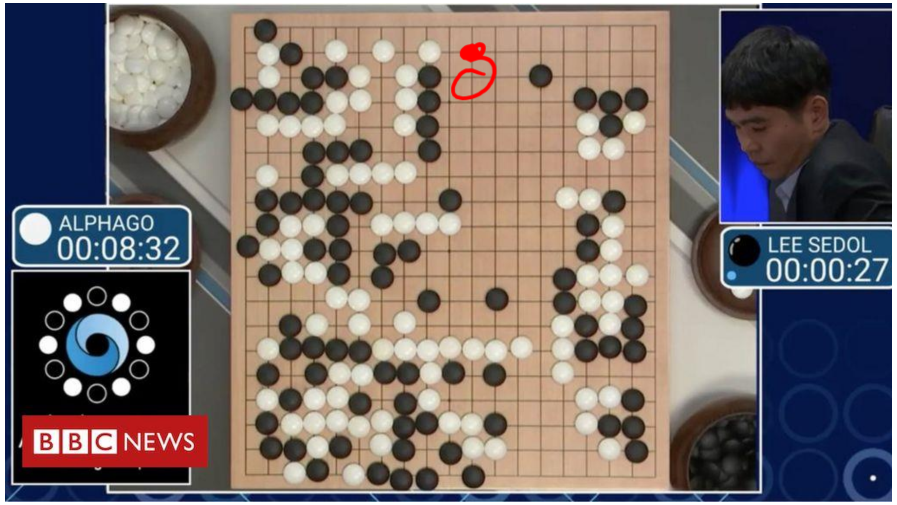
LLMs

- **Examples**:
  - Q-learning, Deep Q-Networks (DQN), and Proximal Policy Optimization (PPO).

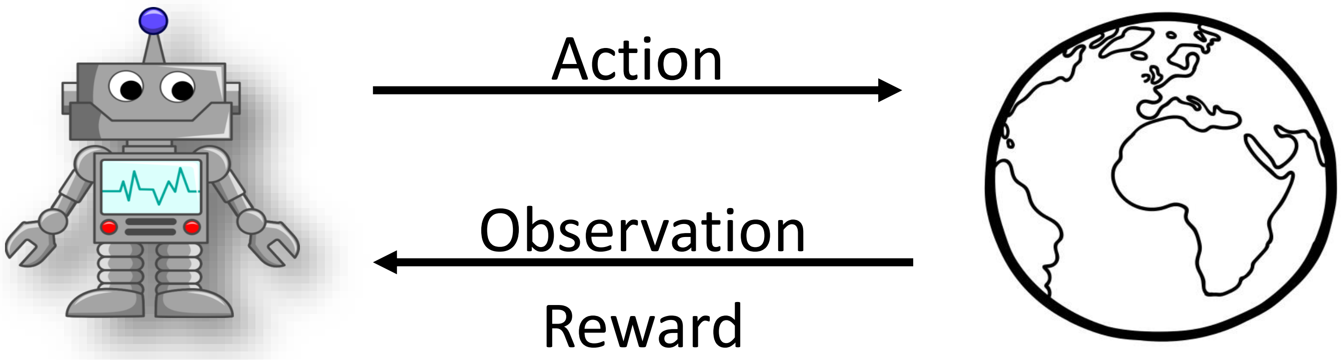
Reinforcement Learning

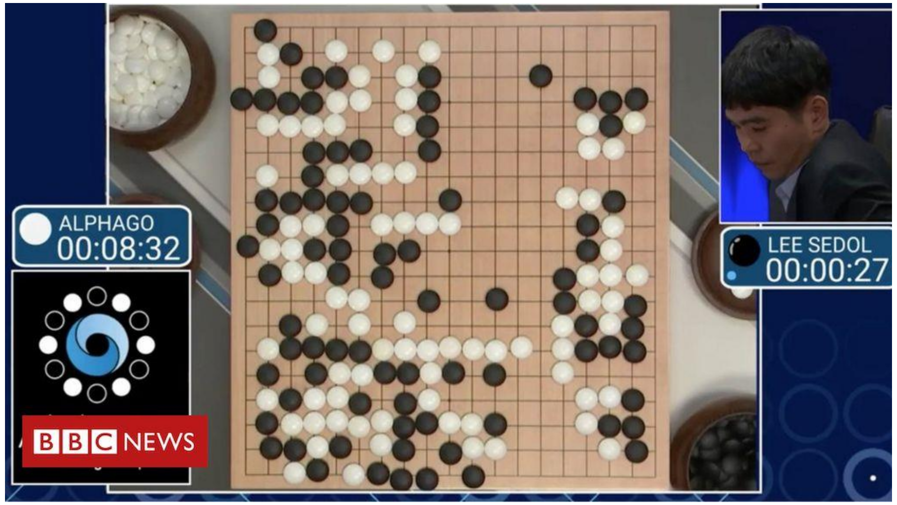


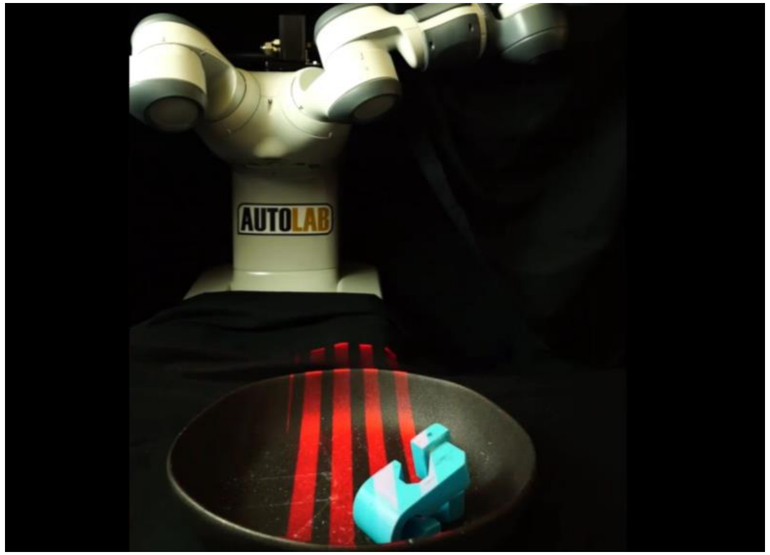
+1 , 0 , -1



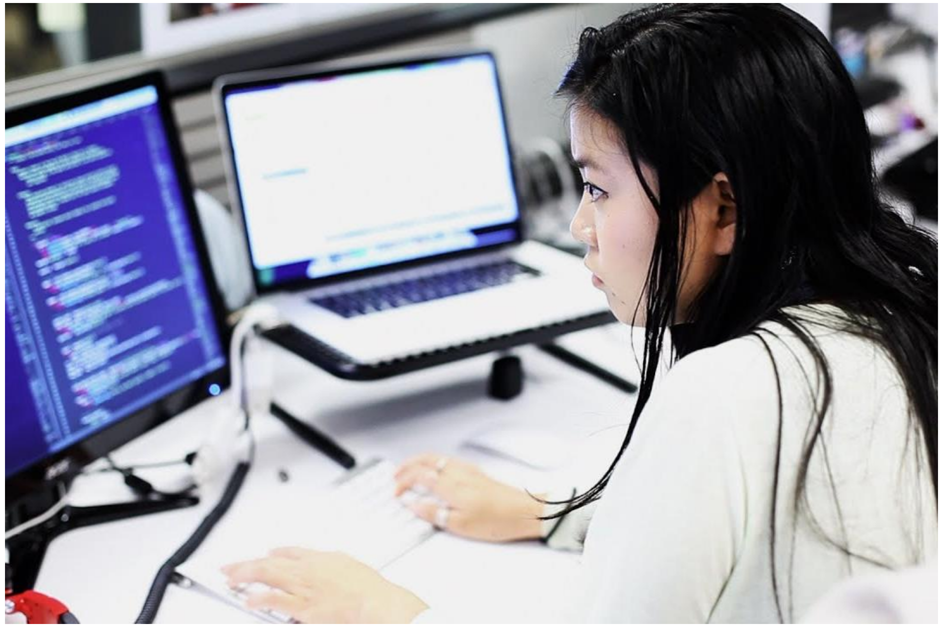
Reinforcement Learning

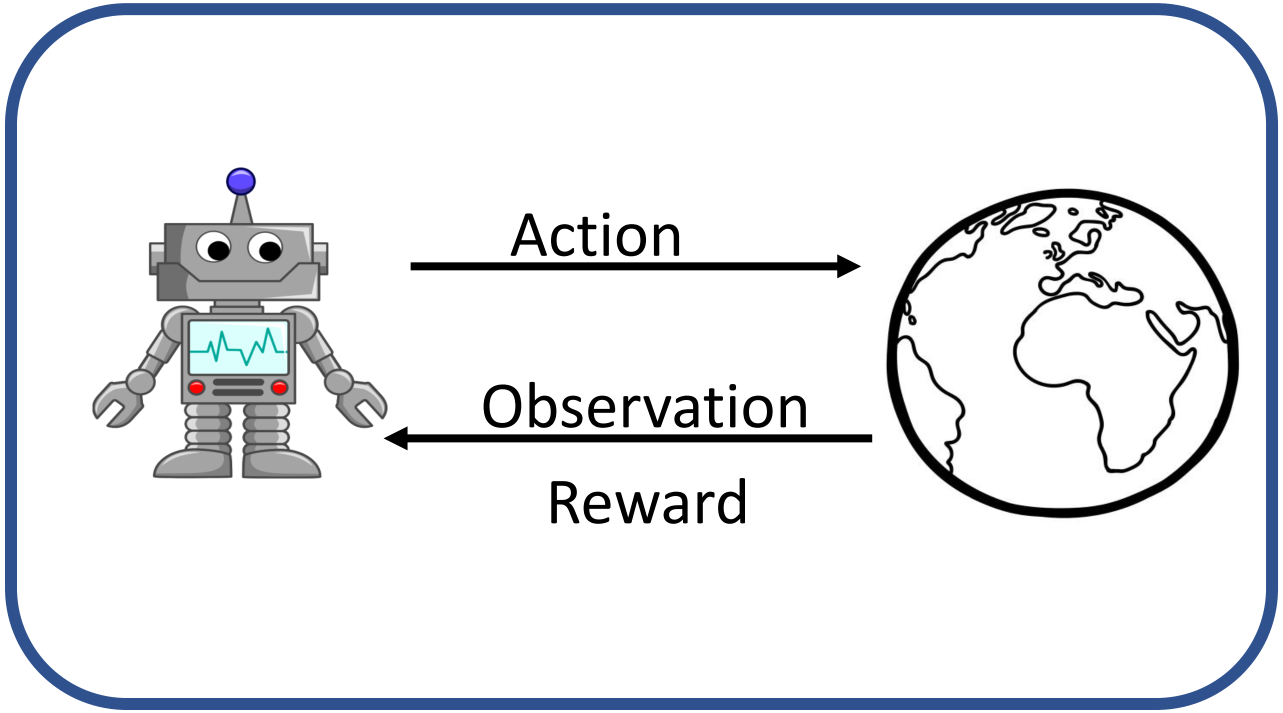




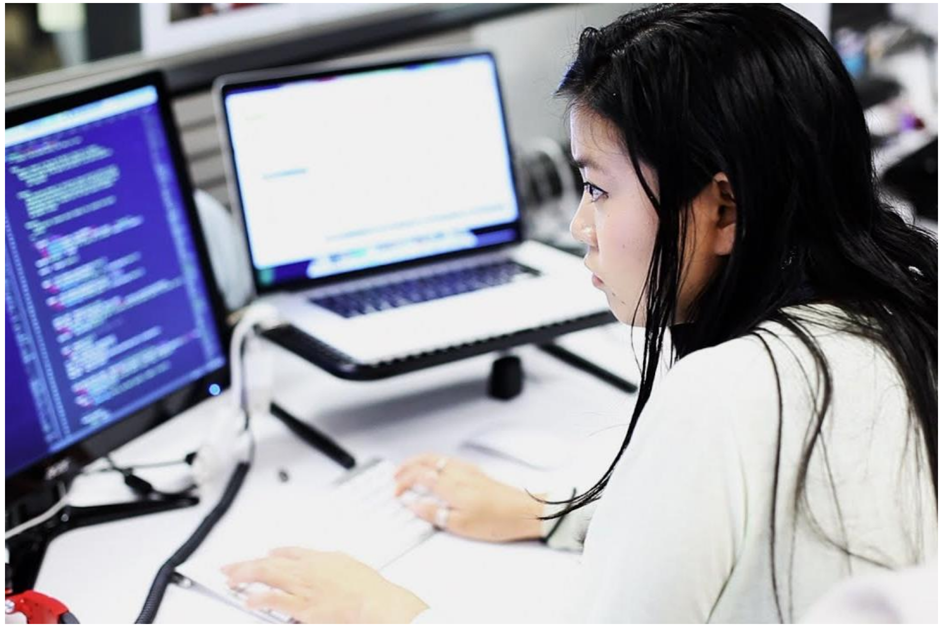


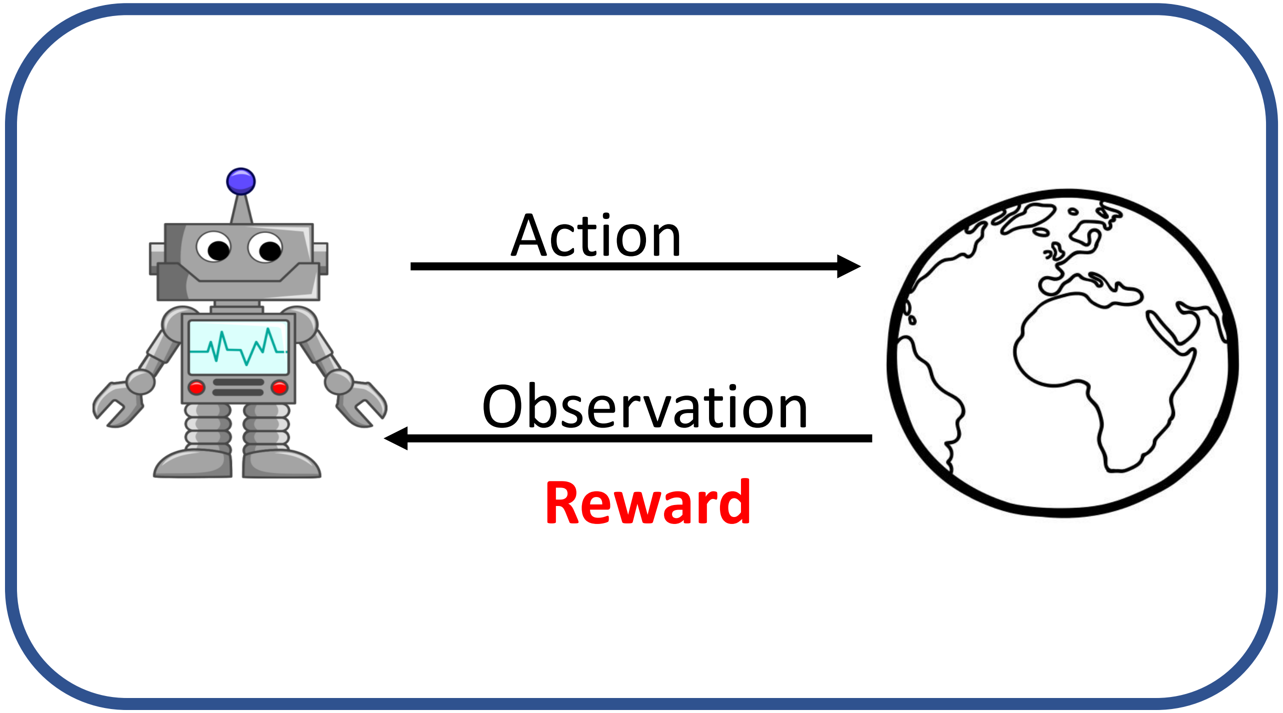
Reward engineering is hard!



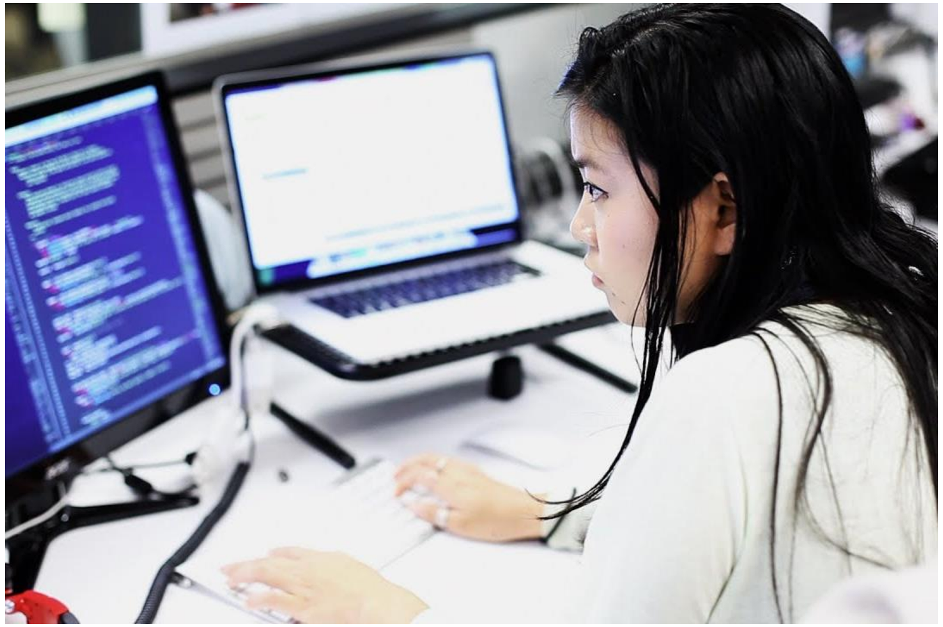


Reward engineering is hard!

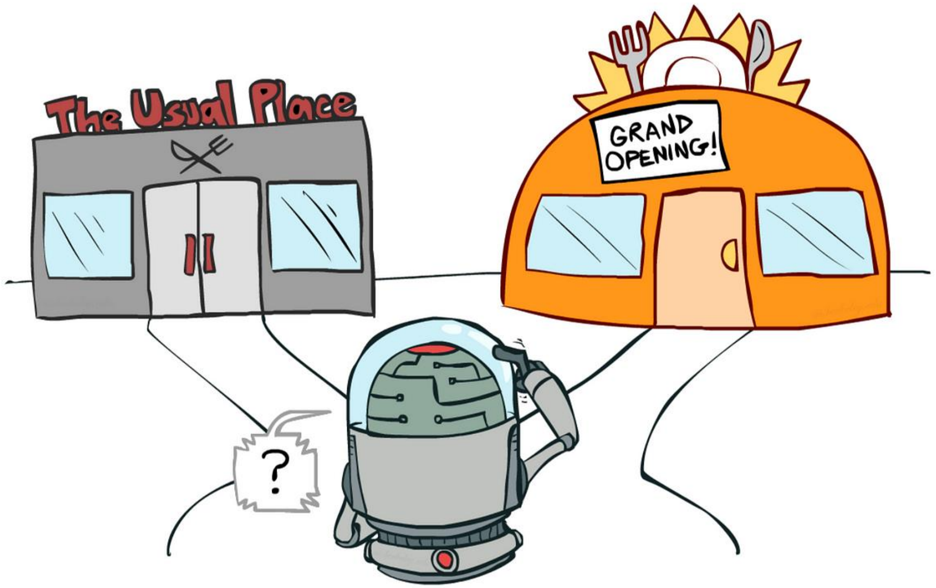




Reward engineering is hard!



Reinforcement learning is hard…even with a reward function!

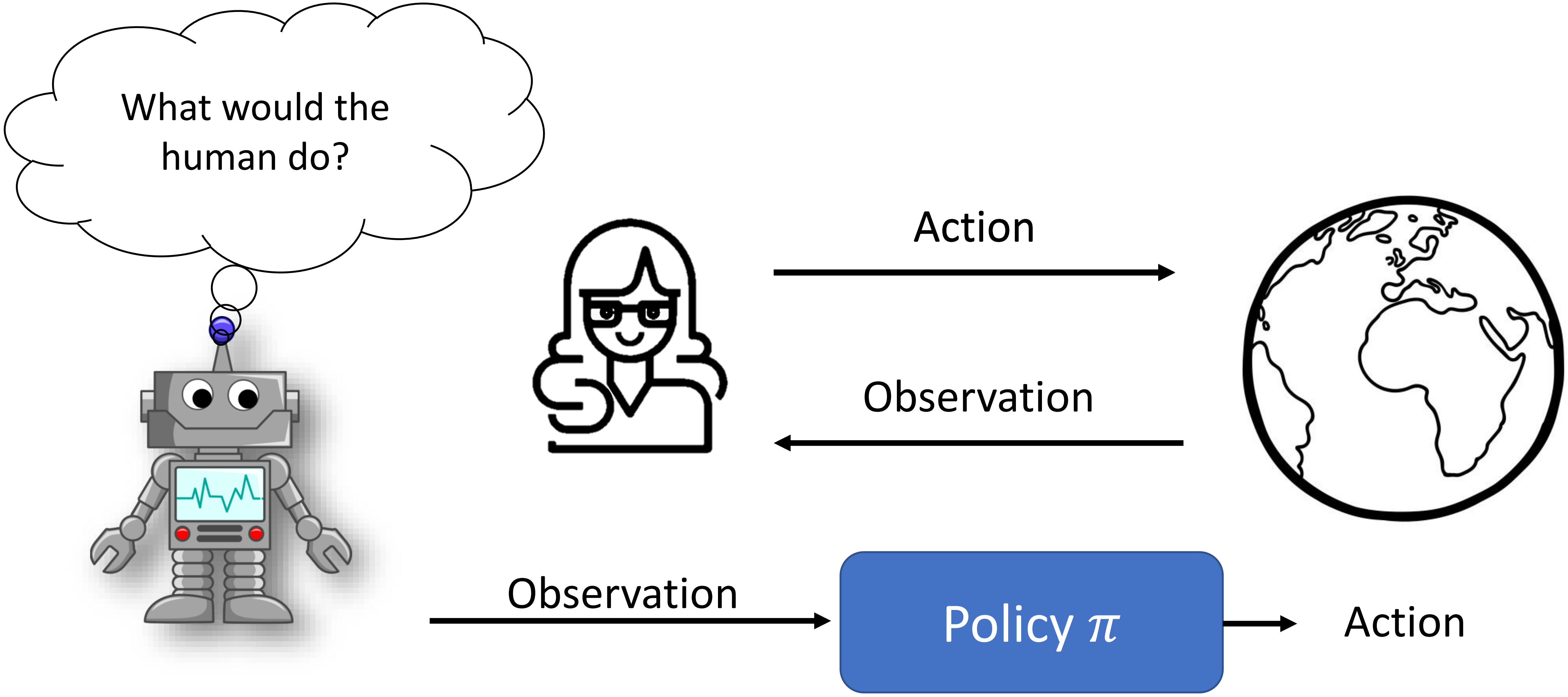


Imitation Learning (Learning from Demonstrations):

Learn a policy from examples of good behavior.

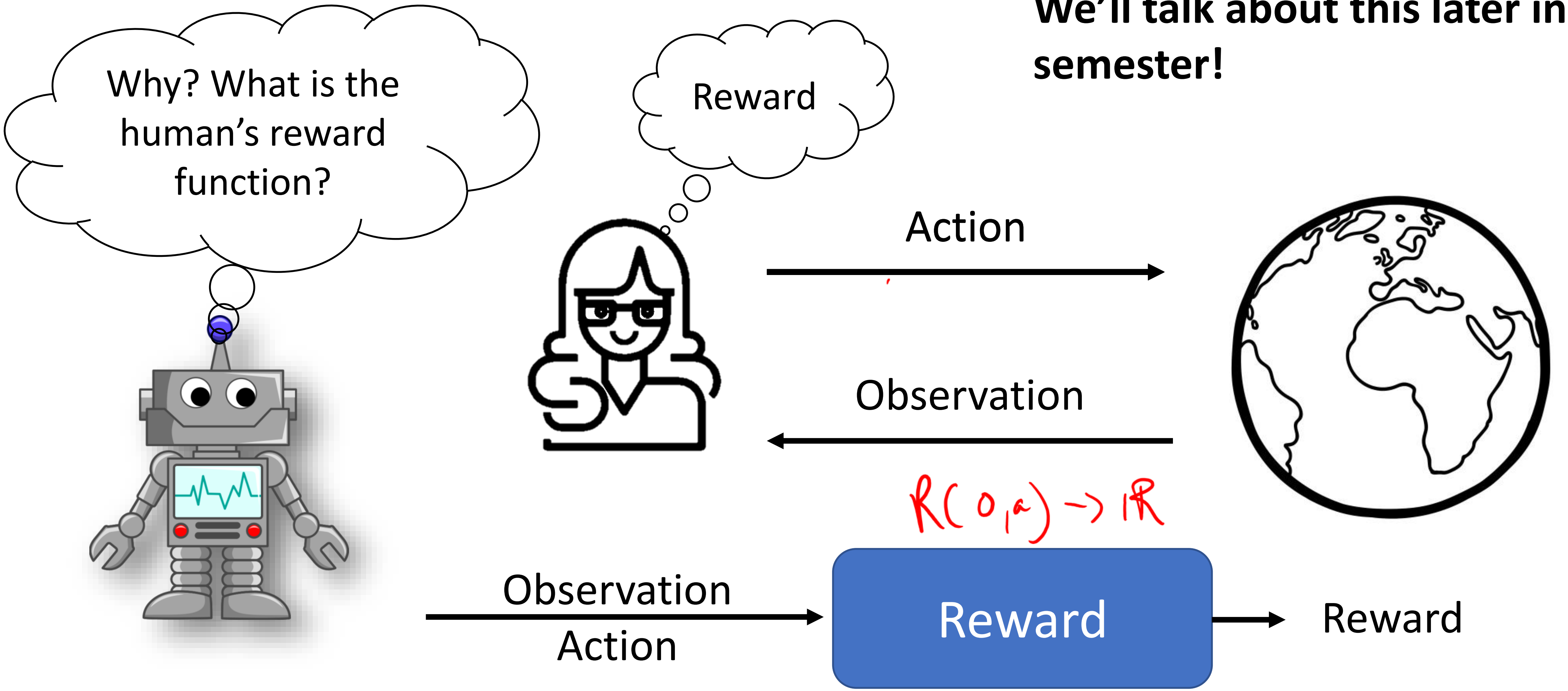
- Often showing is easier than telling.
- Alleviates problem of exploration.

Behavioral Cloning

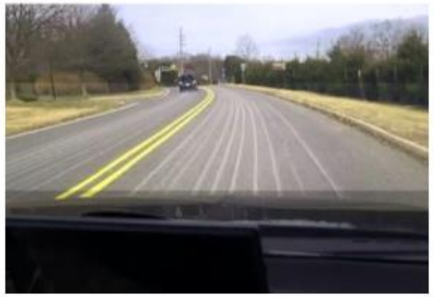


Inverse Reinforcement Learning

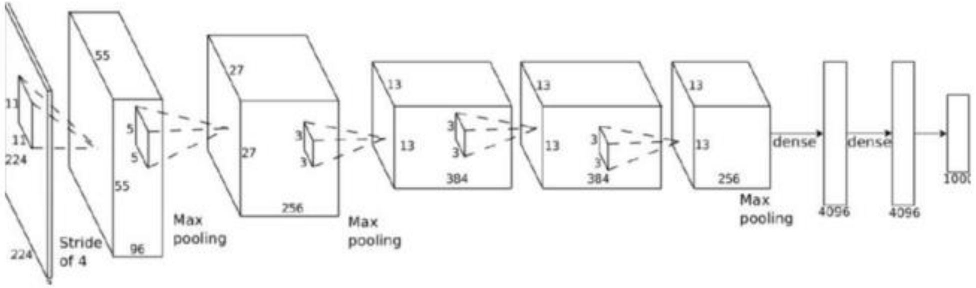
We'll talk about this later in the semester!



Imitation Learning via Behavioral Cloning



$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

$\mathbf{o}_t$
$\mathbf{a}_t$

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

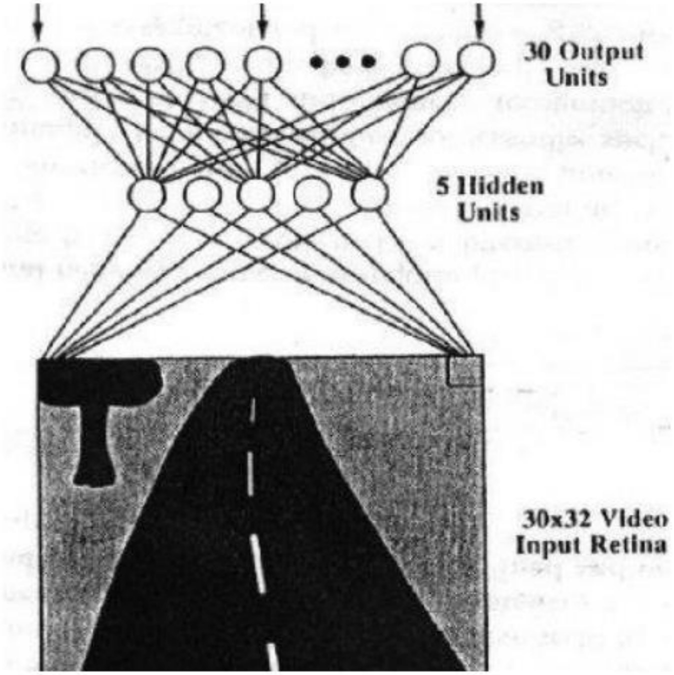
Live demo

python test_gym.py

python mountain_car_bc.py --num_demos 1

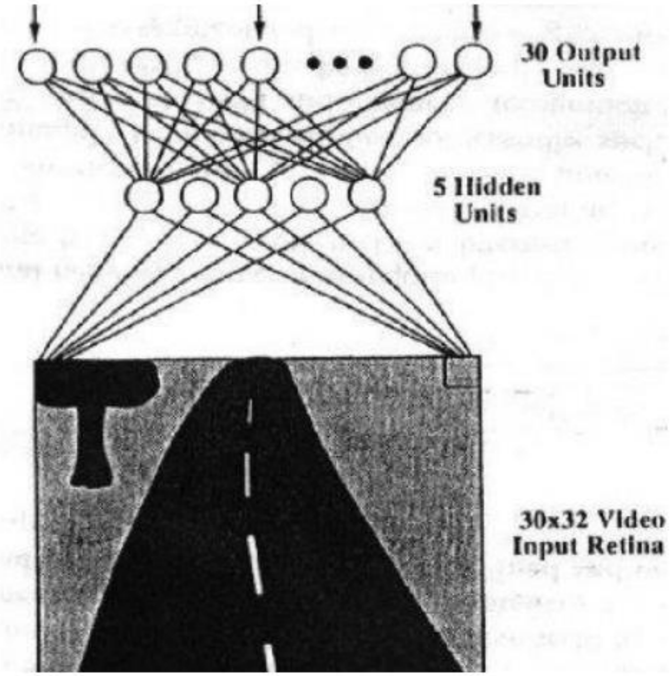
ALVINN: One of the first imitation learning systems

ALVINN: **A**utonomous **L**and **V**ehicle **I**n a **N**eural **N**etwork
1989

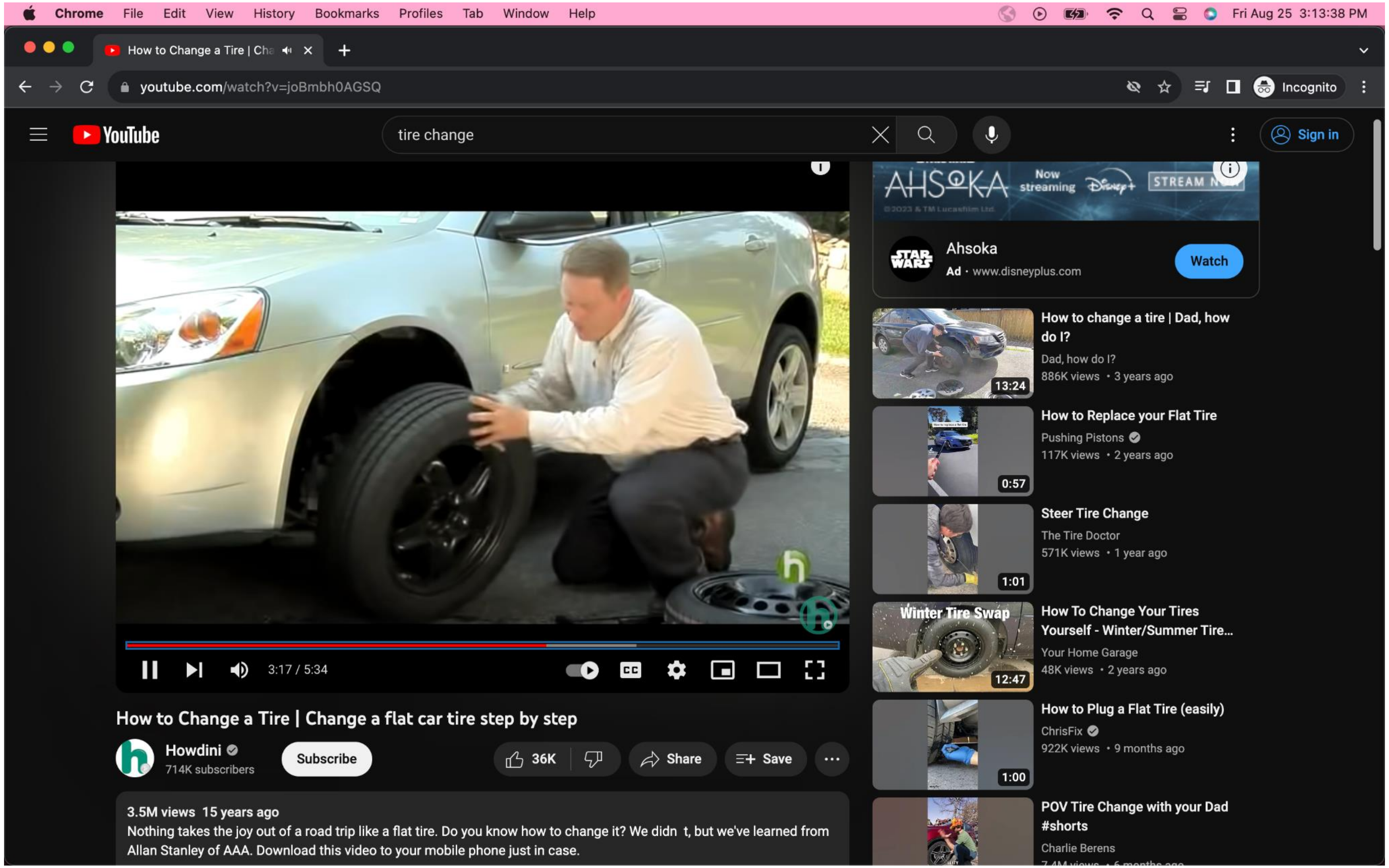


ALVINN: One of the first imitation learning systems

ALVINN: **A**utonomous **L**and **V**ehicle **I**n a **N**eural **N**etwork
1989



What if you don’t have actions?



Behavioral Cloning from Observation (Torabi et al. 2018)

demo = $(s_1, s_2 \ldots\ldots s_{N-1}, s_N)$

$\hat{a}_1 = g(s_1, s_2) \quad \ldots\ldots \quad \hat{a}_{N-1}$

Explore → $\underline{s, a, s', a'}, s''$

Dynamics model

$f(s,a) \rightarrow s'$

Train action predictor

Inverse dynamics model

$g(s,s') \rightarrow a$

$s \xrightarrow{a} s'$

BC(O) Explore, learn g, relabel, BC

What could go wrong?

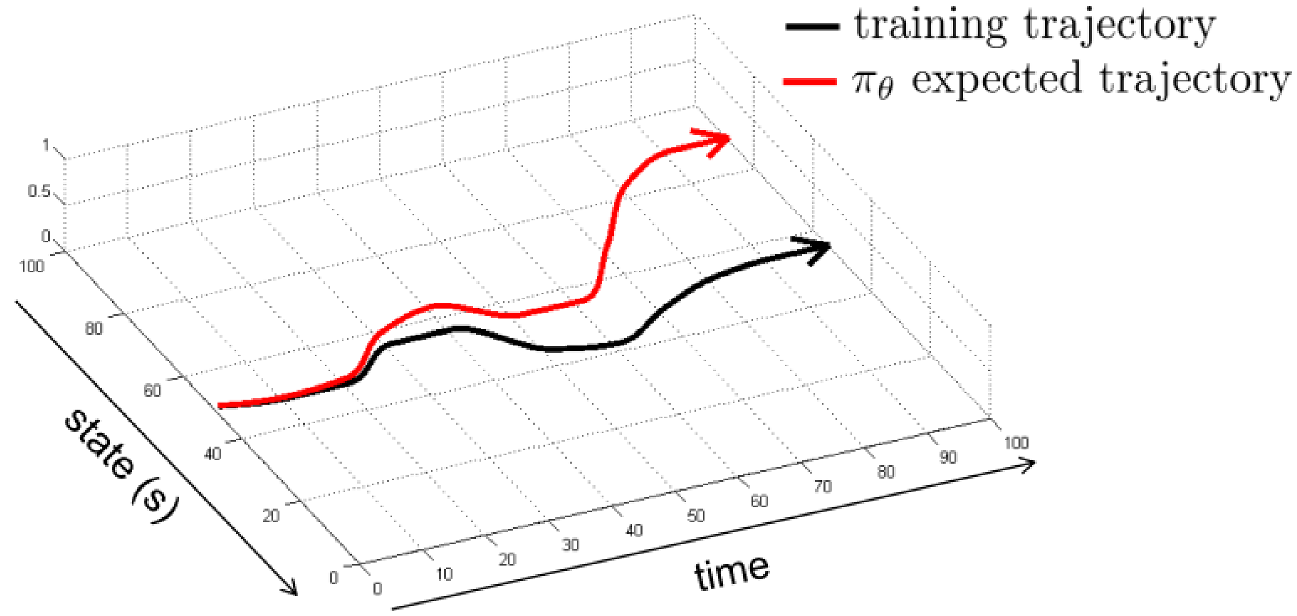
$\mathbf{o}_t$

$\mathbf{a}_t$

training data

supervised learning

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

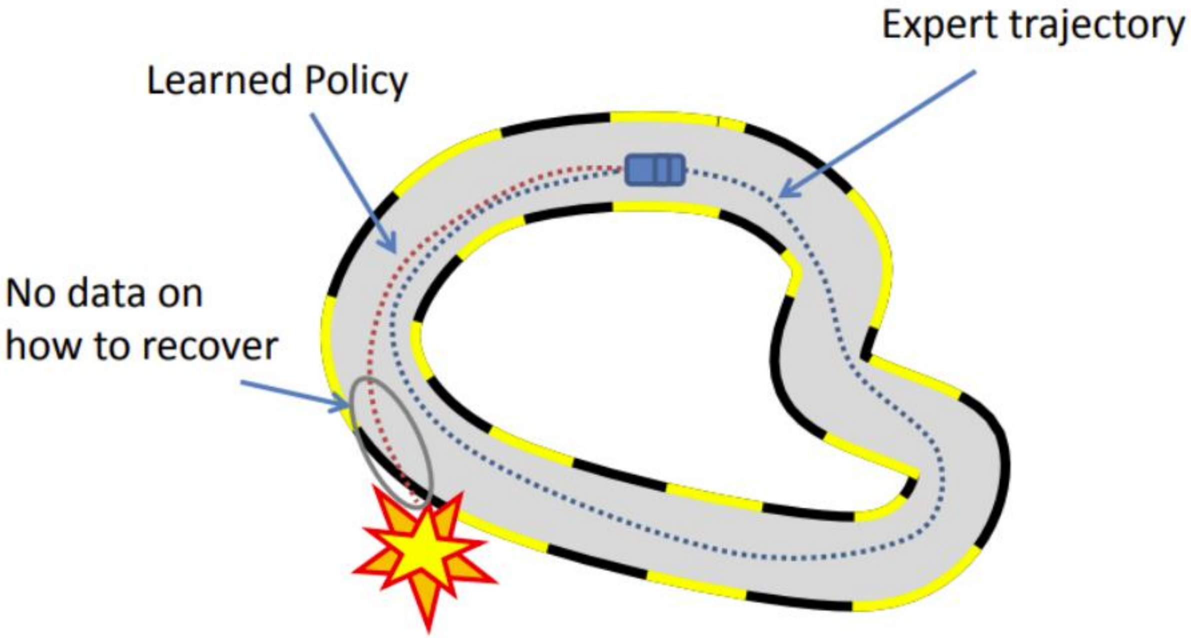


Distribution Shift

$\pi(s) \rightarrow a$

$$p_{\pi^*}(o_t) \neq p_{\pi_\theta}(o_t)$$

π^* expert

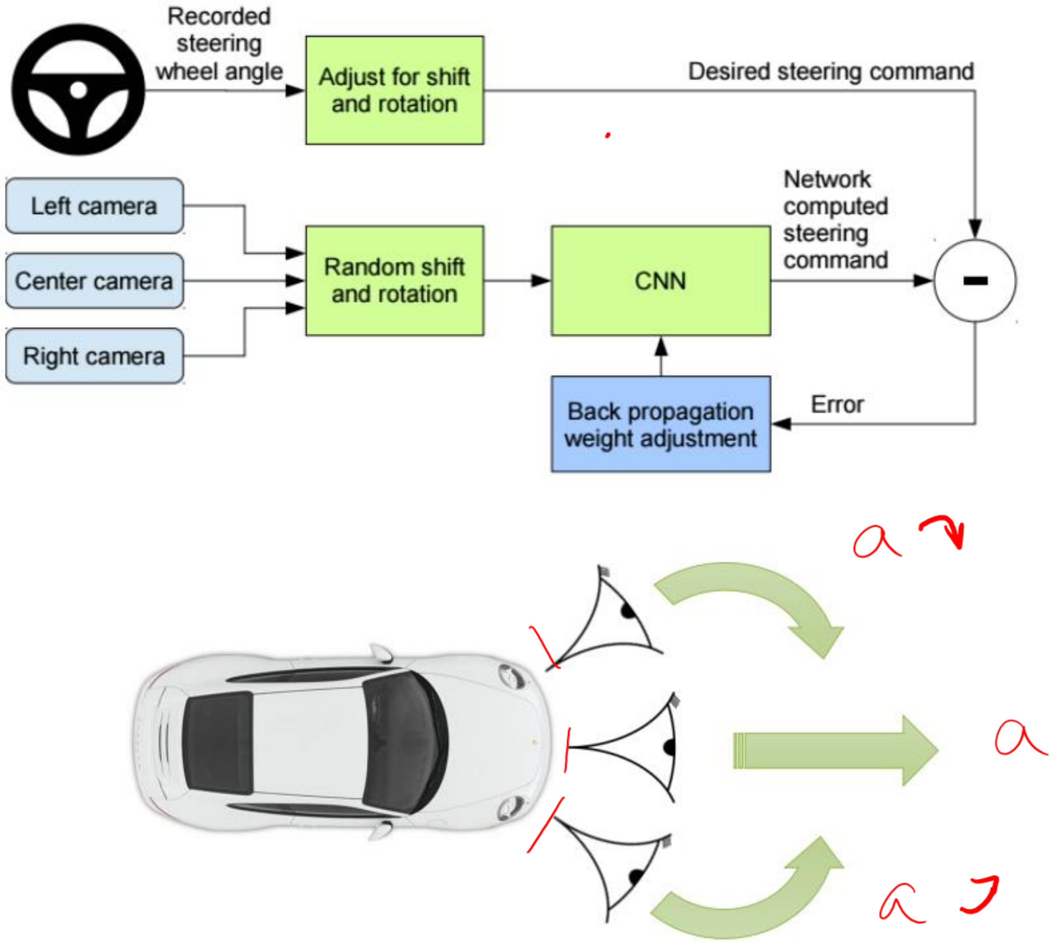


	Supervised Learning	Supervised Learning + Control
Train	$(x, y) \sim D$	$s \sim P(\cdot\,\vert s, \pi^*(s))$
Test	$(x, y) \sim D$	$s \sim P(\cdot\,\vert s, \pi(s))$

But it still can work in practice…

Bojarski et al. ‘16, NVIDIA

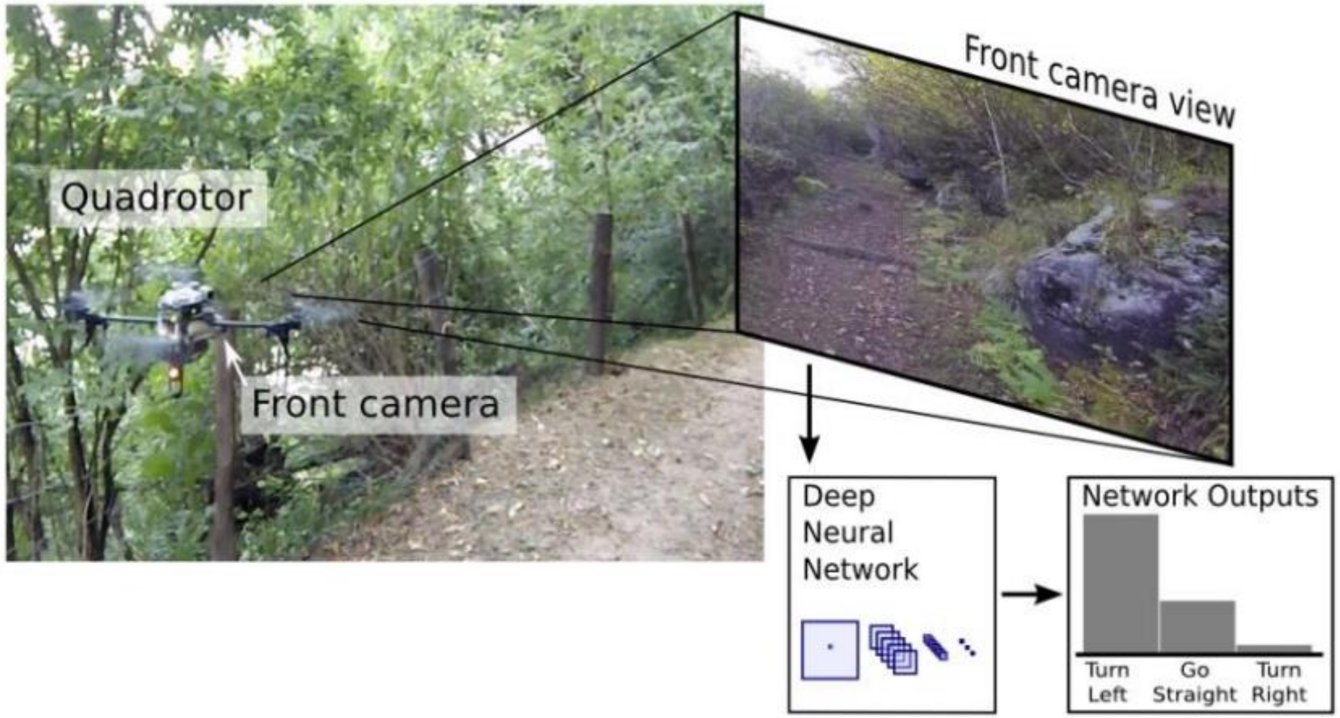
How?



Bojarski et al. '16, NVIDIA

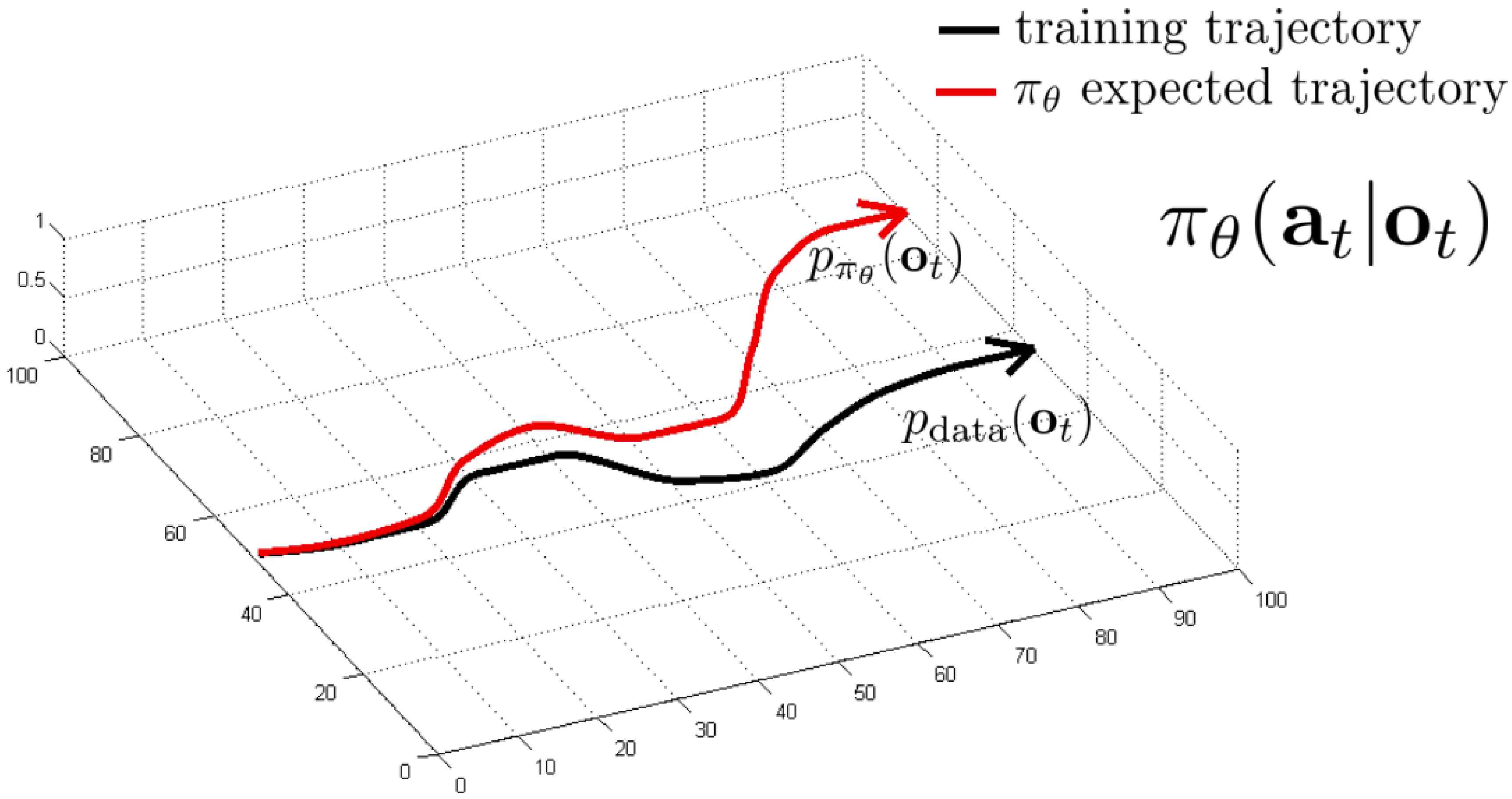
A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots

Alessandro Giusti[1], Jérôme Guzzi[1], Dan C. Cireşan[1], Fang-Lin He[1], Juan P. Rodríguez[1]
Flavio Fontana[2], Matthias Faessler[2], Christian Forster[2]
Jürgen Schmidhuber[1], Gianni Di Caro[1], Davide Scaramuzza[2], Luca M. Gambardella[1]



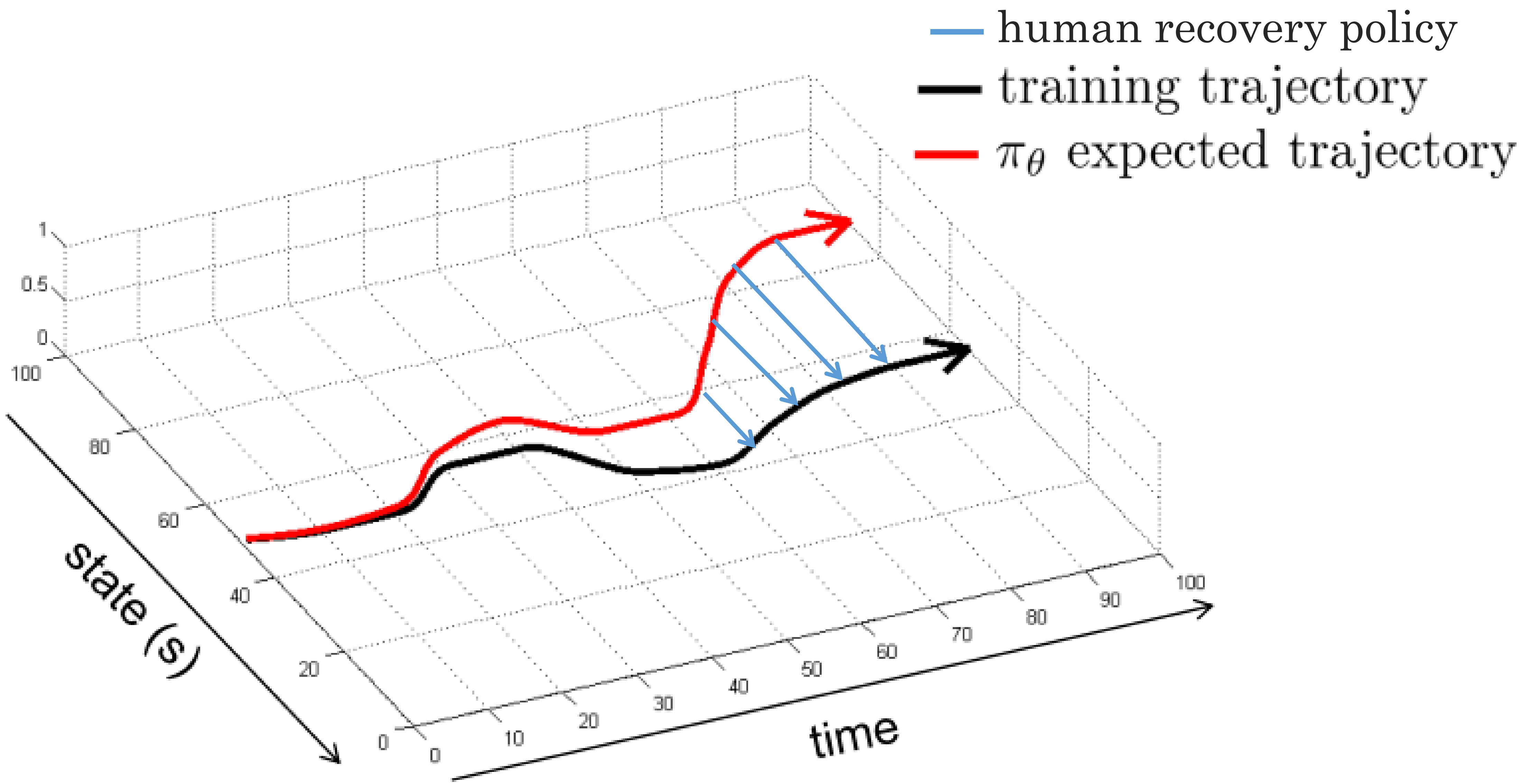

Deep Neural Network
Control Signal

Can we make it work more often?



$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?


human recovery policy
training trajectory
π_θ expected trajectory
1
0.5
0
100
80
60
40
20
0
state (s)
0
10
20
30
40
50
60
70
80
90
100
time

Dataset Aggregation

DAgger

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

idea: instead of being clever about $p_{\pi_\theta}(\mathbf{o}_t)$, be clever about $p_{\text{data}}(\mathbf{o}_t)$!

DAgger: Dataset Aggregation

(o, a)

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

but need labels $\mathbf{a}_t$!

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t^*$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Ross et al. '11

DAgger has very nice theoretical guarantees.

Why might it be **hard** to implement in practice?

DAgger: Dataset Aggregation

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

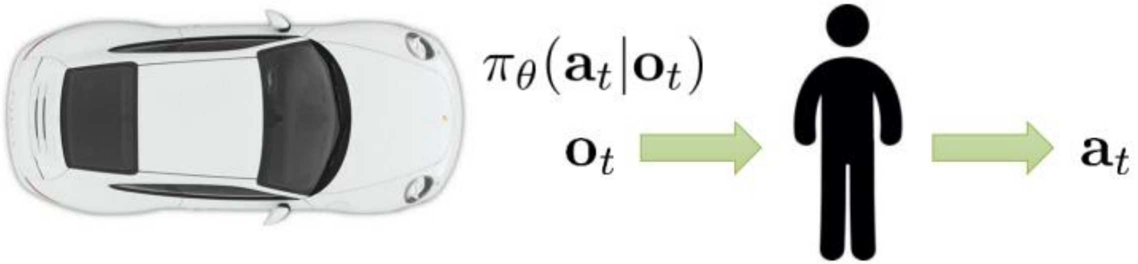
how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

but need labels $\mathbf{a}_t$!

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Ross et al. ‘11

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$



DAgger has very nice theoretical guarantees.

Why might it be **easy** to implement in practice?

DAgger: Dataset **Aggregation**

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

but need labels $\mathbf{a}_t$!

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask ~~human~~ oracle to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Ross et al. '11

Learn from an Algorithmic Supervisor!

Seita et al. 2020. “Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor”

But we don’t always have access to an algorithmic supervisor…

Can we make DAgger more practical when dealing with real human labeling?

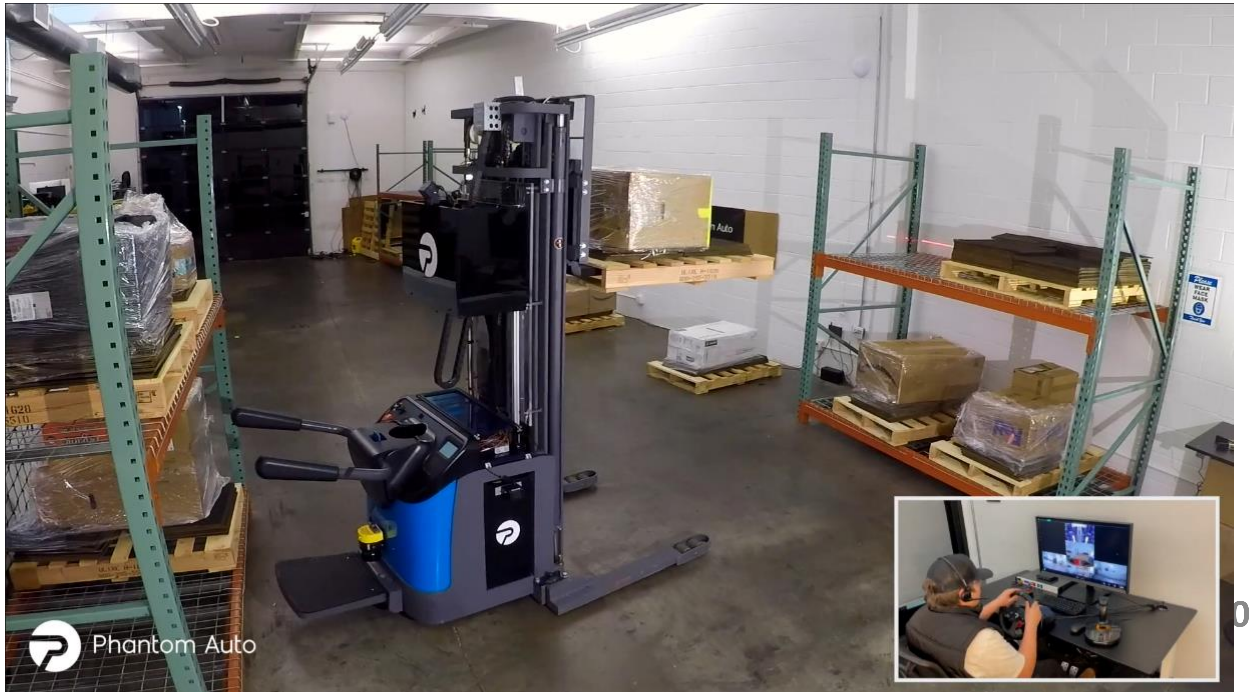
PLUS ONE ROBOTICS

WAYMO

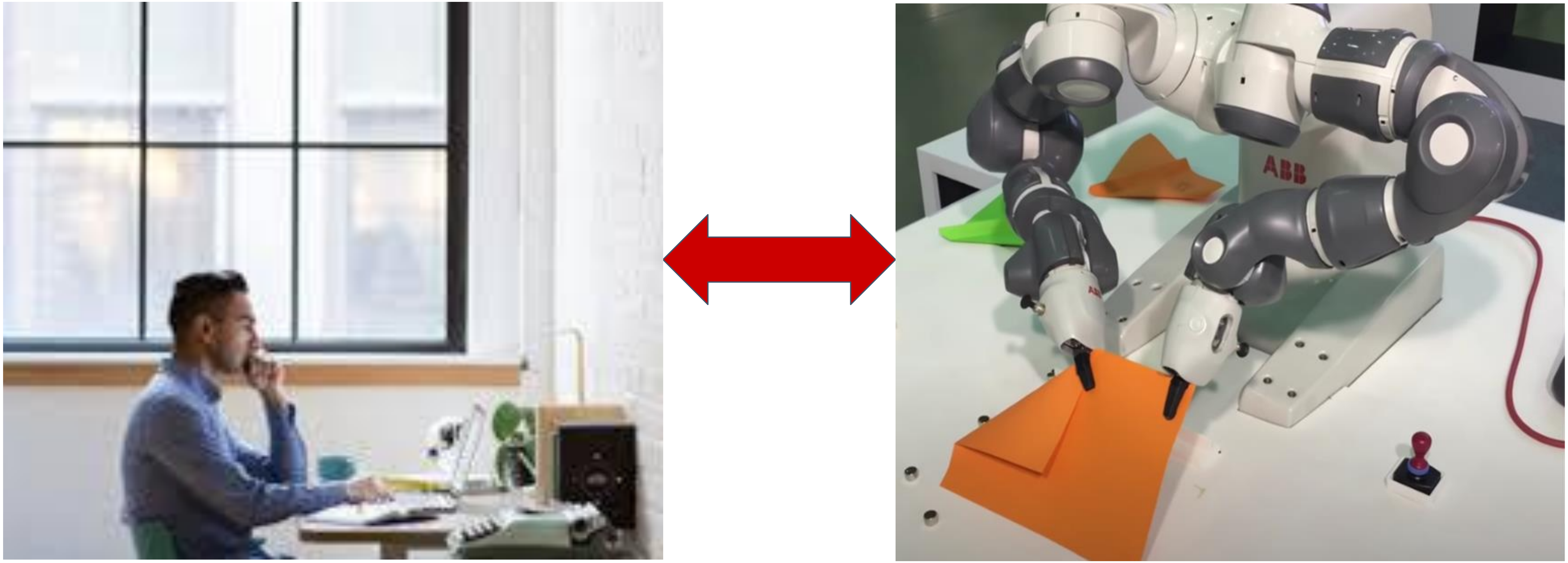
zoox

nimble

Phantom Auto



Interactive IL



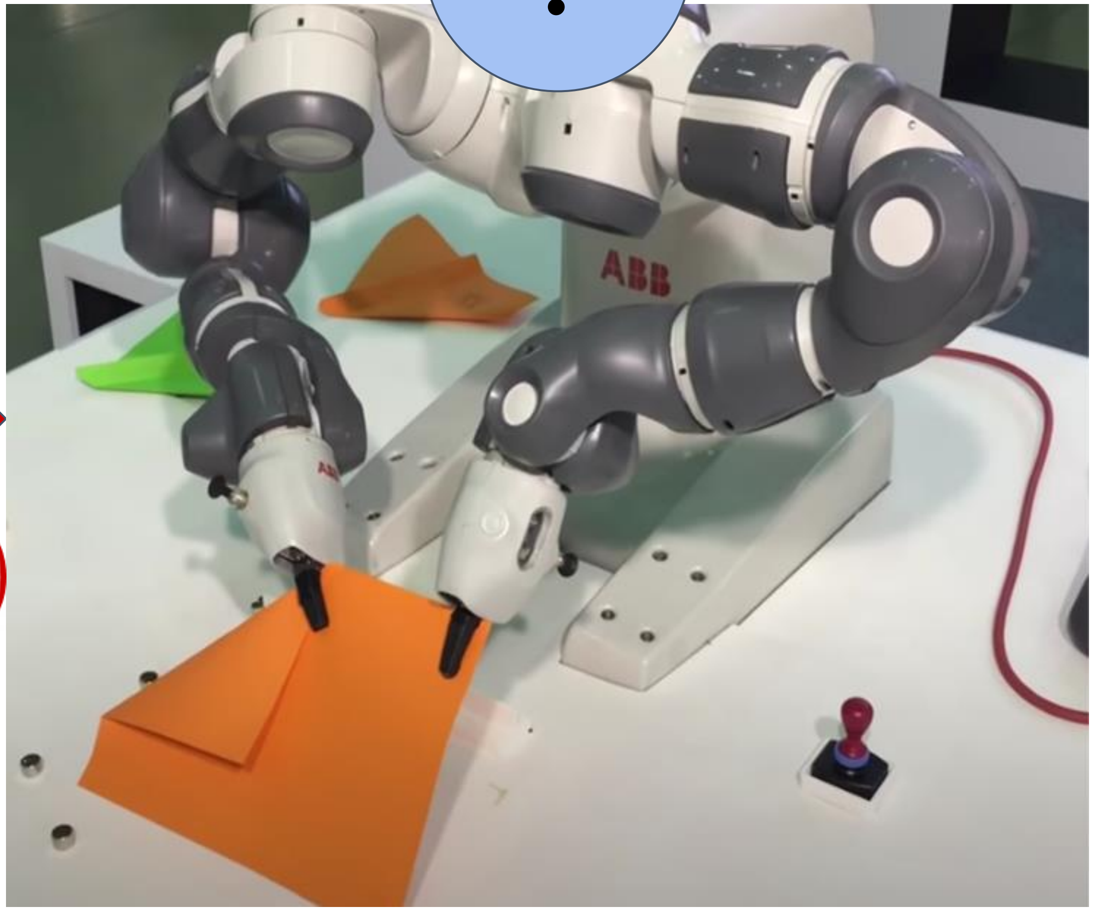
Interactive IL

?

$\pi_H(s)$

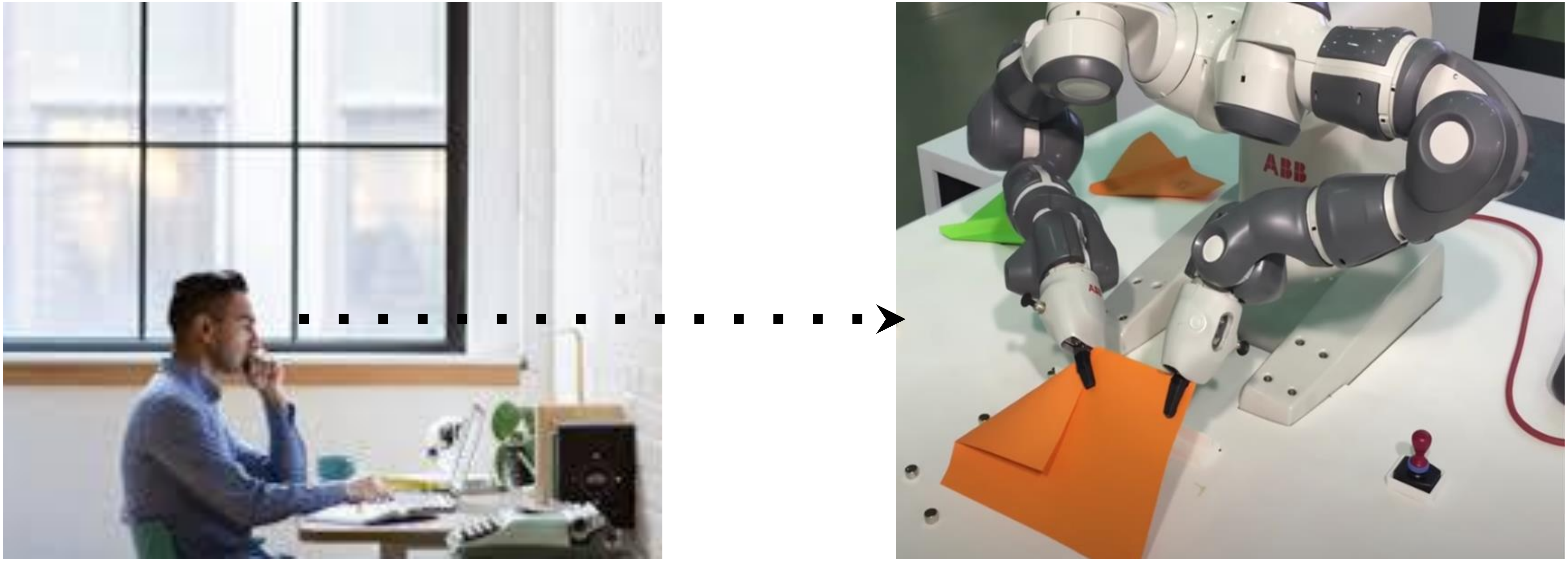
$\pi_{\text{meta}}(s)$

???



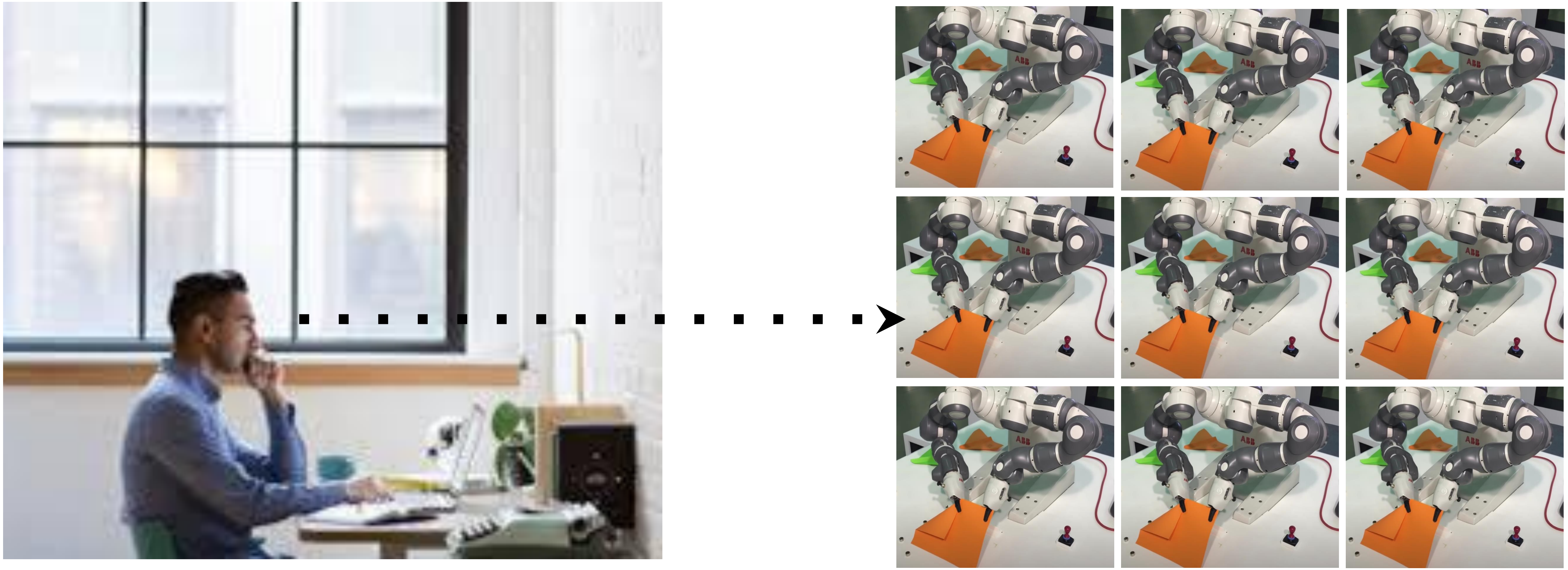
$\pi_R(s)$

Human-Gated Interactive IL



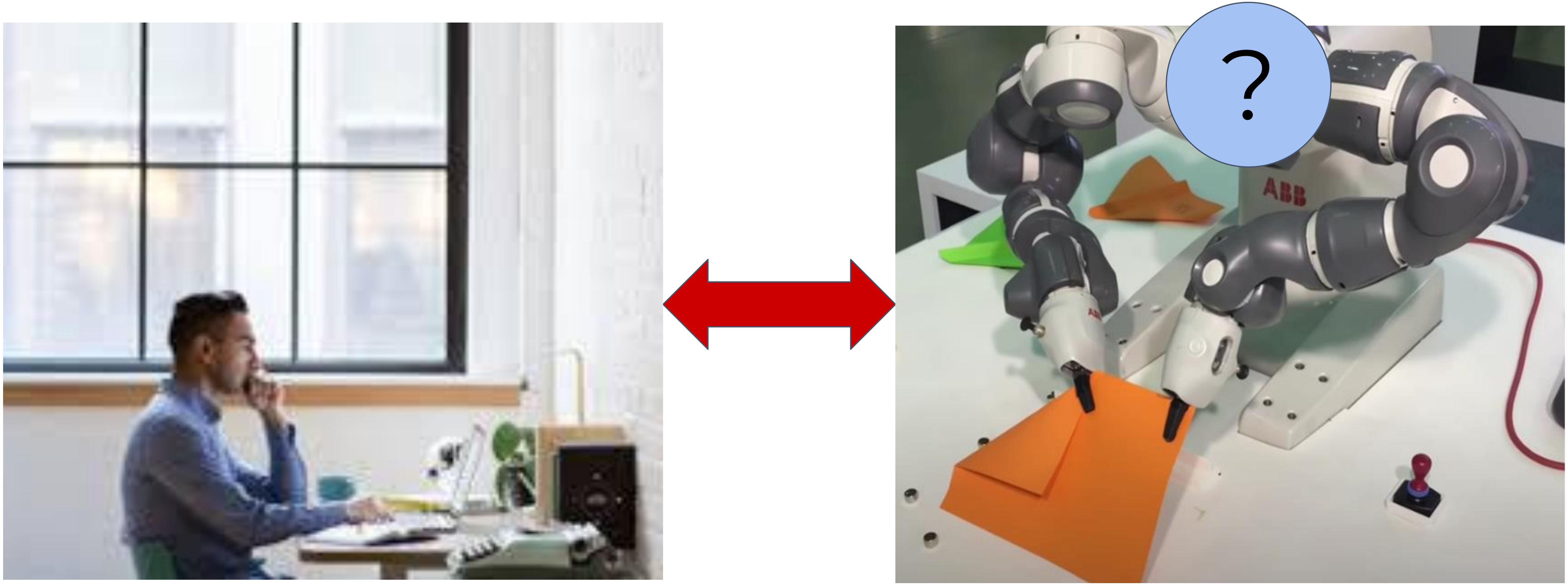
[3] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. HG-DAgger: Interactive Imitation Learning with Human Experts. ICRA 2019.

Human-Gated Interactive IL



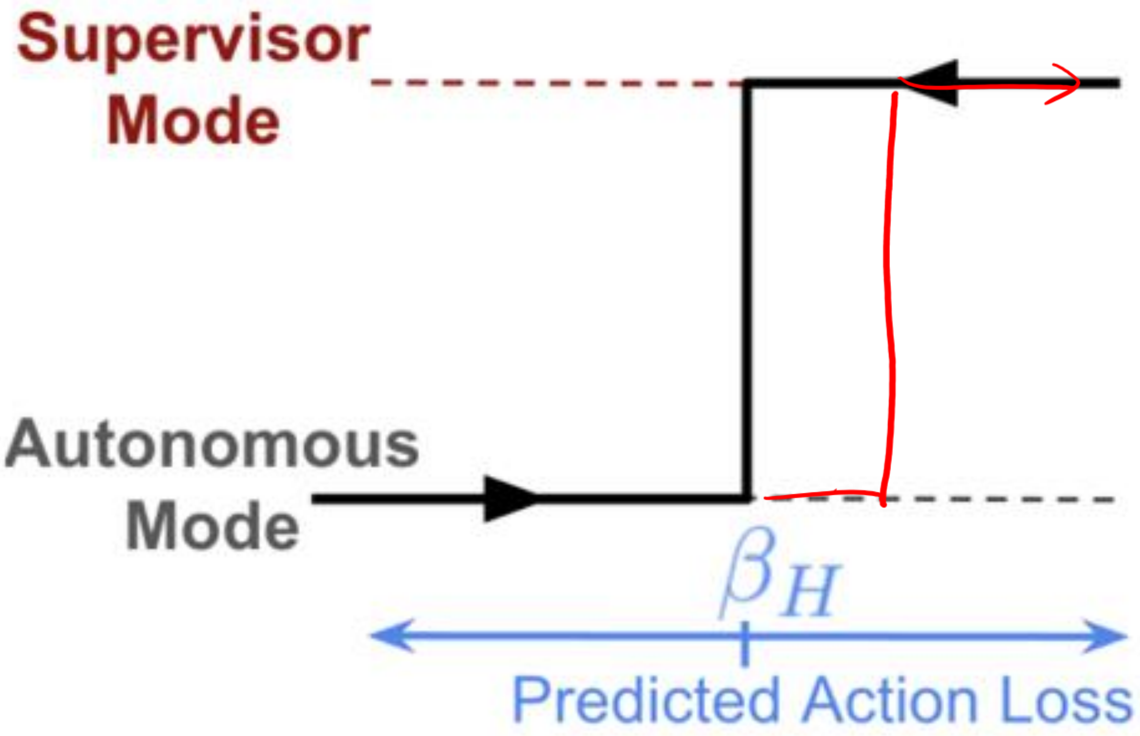
[3] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. HG-DAgger: Interactive Imitation Learning with Human Experts. ICRA 2019.

Robot-Gated Interactive IL



[4] J. Zhang, K. Cho. Query-Efficient Imitation Learning for End-to-End Autonomous Driving. AAAI 2017.
[5] K. Menda, K. Driggs-Campbell, M. Kochenderfer. EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning. IROS 2019.

SafeDAgger



Predicted action loss = predicted difference between human and robot action.

Trained using held-out set of data from human.

J. Zhang, K. Cho. Query-Efficient Imitation Learning for End-to-End Autonomous Driving. AAAI 2017.

When should a robot ask for help?

Novel (and risky)

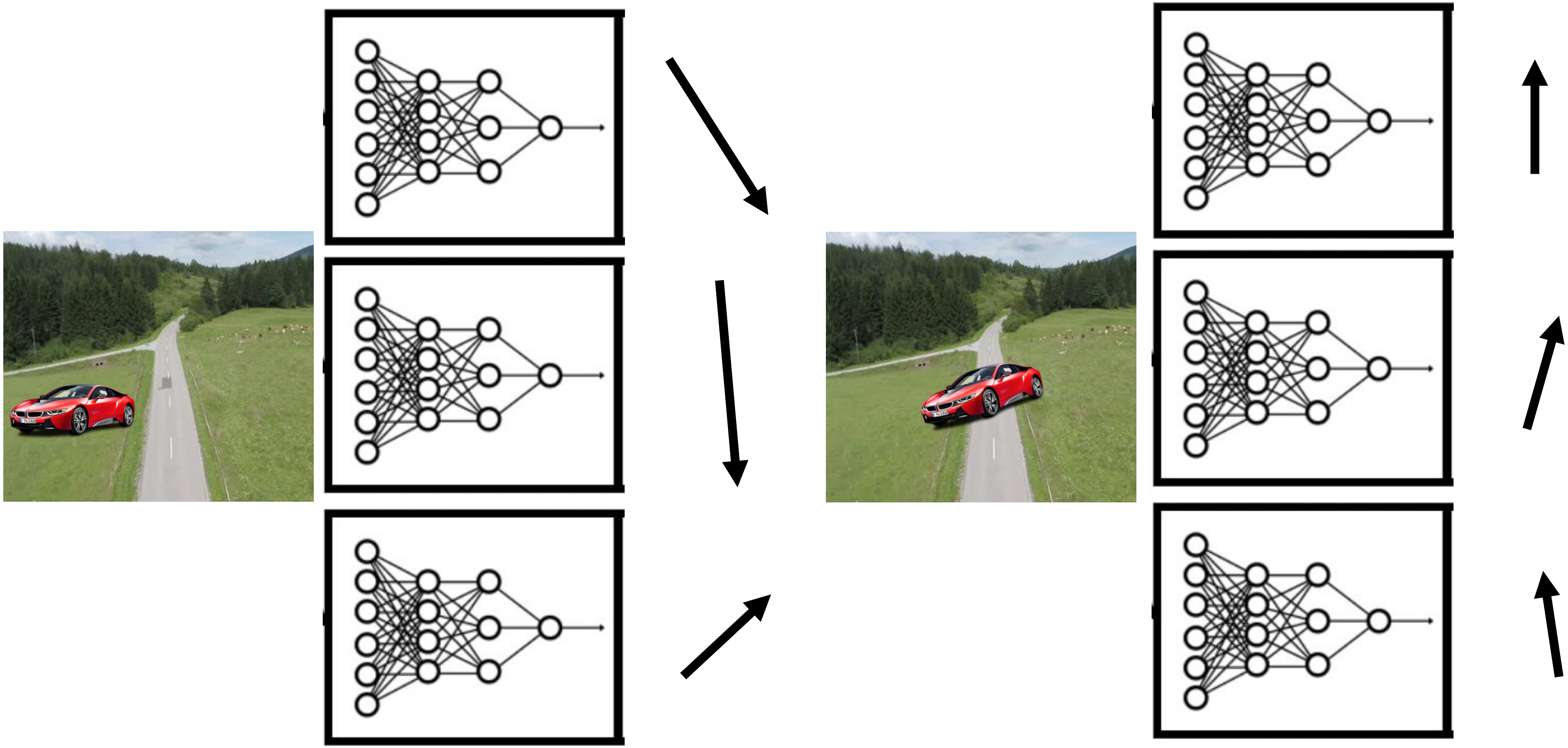
When should a robot ask for help?

Novel (and risky)

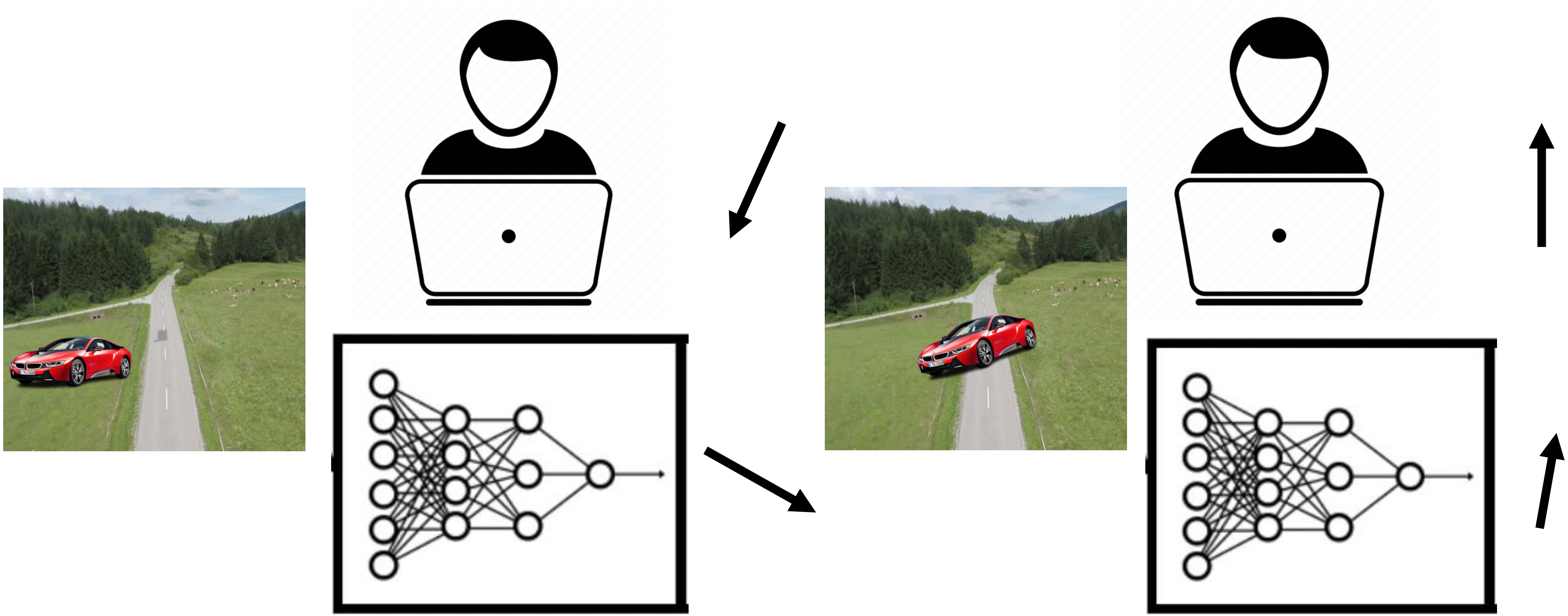
Risky (but not novel)

Novelty Estimation

Ensemble



Novelty Estimation: Supervisor Mode



Risk Estimation

goal classifier

$$Q_{\mathcal{G}}^{\pi_r}(s_t, a_t) = \mathbb{E}_{\pi_r}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \mathbb{1}_{\mathcal{G}}(s'_t) | s_t, a_t\right]$$

Risk Estimation

$$Q_{\mathcal{G}}^{\pi_r}(s_t, a_t) = \mathbb{E}_{\pi_r}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \mathbb{1}_{\mathcal{G}}(s'_t) | s_t, a_t\right]$$

$$\text{Risk}^{\pi_r}(s, a) = 1 - \hat{Q}_{\phi,\mathcal{G}}^{\pi_r}(s, a)$$

Risk Estimation

$$Q_{\mathcal{G}}^{\pi_r}(s_t, a_t) = \mathbb{E}_{\pi_r}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \mathbb{1}_{\mathcal{G}}(s'_t) | s_t, a_t\right]$$

$$\mathrm{Risk}^{\pi_r}(s, a) = 1 - \hat{Q}_{\phi,\mathcal{G}}^{\pi_r}(s, a)$$

$$J_{\mathcal{G}}^{Q}(s_t, a_t, s_{t+1}; \phi) =$$
$$\frac{1}{2}\left(\hat{Q}_{\phi,\mathcal{G}}^{\pi_r}(s_t, a_t) - (\mathbb{1}_{\mathcal{G}}(s_t) + \ (1 - \mathbb{1}_{\mathcal{G}}(s_t))\gamma \hat{Q}_{\phi,\mathcal{G}}^{\pi_r}(s_{t+1}, \pi_r(s_{t+1})))\right)^2$$

Putting it all together…

AUTONOMOUS MODE

$$\text{Novelty}(s_t) > \delta_h$$

OR

$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) > \beta_h$$

$\longrightarrow$ Switch to SUPERVISOR MODE

Putting it all together...

AUTONOMOUS MODE

$$\text{Novelty}(s_t) > \delta_h$$
OR
$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) > \beta_h$$

$\longrightarrow$ Switch to SUPERVISOR MODE

SUPERVISOR MODE

$$||\pi_r(s_t) - \pi_h(s_t)||_2^2 < \delta_r$$
AND
$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) < \beta_r$$

$\longrightarrow$ Switch to AUTONOMOUS MODE

Putting it all together…

AUTONOMOUS MODE

$$\text{Novelty}(s_t) > \delta_h$$

OR

$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) > \beta_h$$

→ Switch to SUPERVISOR MODE

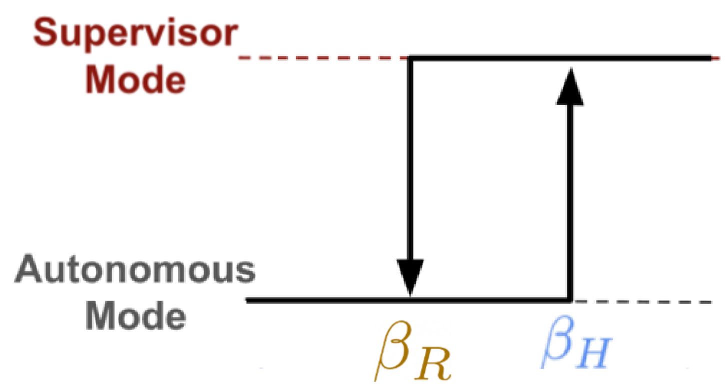
SUPERVISOR MODE

$$||\pi_r(s_t) - \pi_h(s_t)||_2^2 < \delta_r$$

AND

$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) < \beta_r$$

→ Switch to AUTONOMOUS MODE



How do we deal with all the hyperparameters?

Putting it all together...

AUTONOMOUS MODE

$$\text{Novelty}(s_t) > \boxed{\delta_h}$$

OR

$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) > \boxed{\beta_h}$$

Set to 1-α quantiles of empirical data

→ Switch to SUPERVISOR MODE

SUPERVISOR MODE

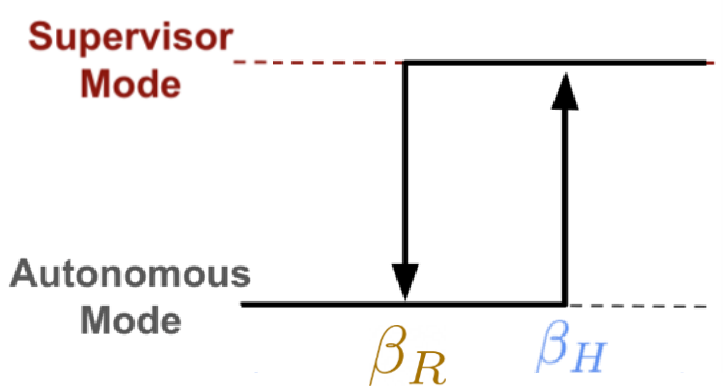
$$||\pi_r(s_t) - \pi_h(s_t)||_2^2 < \delta_r$$

AND

$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) < \beta_r$$

→ Switch to AUTONOMOUS MODE

$$\alpha = \frac{\text{desired \# interventions}}{\text{\# robot actions}}$$



Putting it all together...

AUTONOMOUS MODE

$$\text{Novelty}(s_t) > \delta_h$$
OR
$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) > \beta_h$$

Set to 1-α quantiles of empirical data

Switch to SUPERVISOR MODE

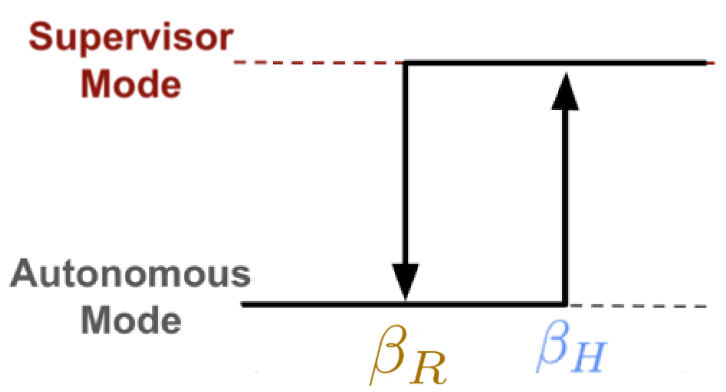
SUPERVISOR MODE

$$||\pi_r(s_t) - \pi_h(s_t)||_2^2 < \delta_r$$
AND
$$\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) < \beta_r$$

Switch to AUTONOMOUS MODE

Set to medians of empirical data

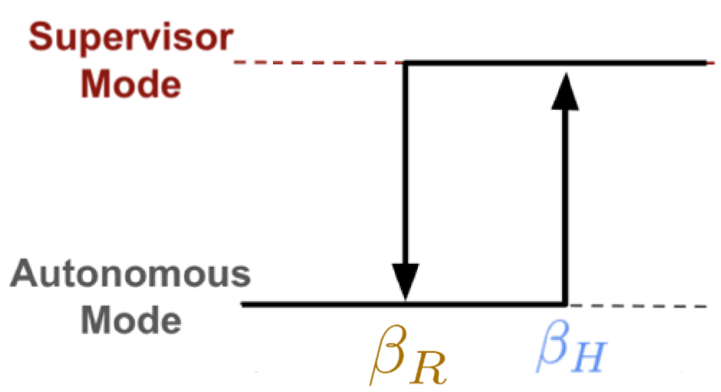
$$\alpha = \frac{\text{desired \# interventions}}{\text{\# robot actions}}$$



Putting it all together...

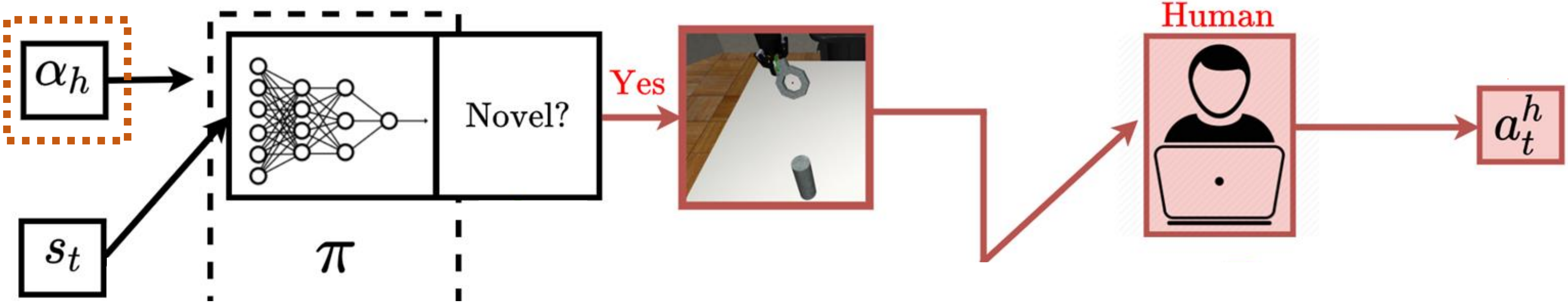
AUTONOMOUS MODE	$\text{Novelty}(s_t) > \delta_h$ OR $\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) > \beta_h$	$\longrightarrow$	Switch to SUPERVISOR MODE
SUPERVISOR MODE	$\lVert\pi_r(s_t) - \pi_h(s_t)\rVert_2^2 < \delta_r$ AND $\text{Risk}^{\pi_r}(s_t, \pi_r(s_t)) < \beta_r$	$\longrightarrow$	Switch to AUTONOMOUS MODE

$$\alpha = \frac{\text{desired \# interventions}}{\text{\# robot actions}}$$



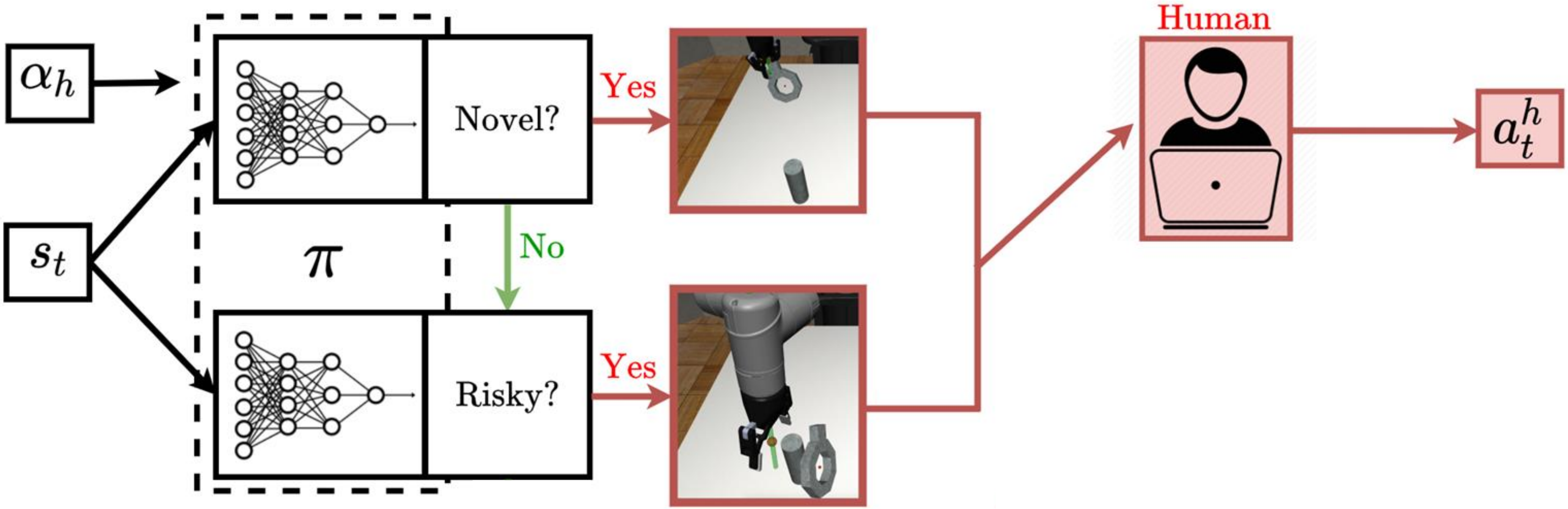
ThriftyDAgger

Target percent of time human wants to give interventions.



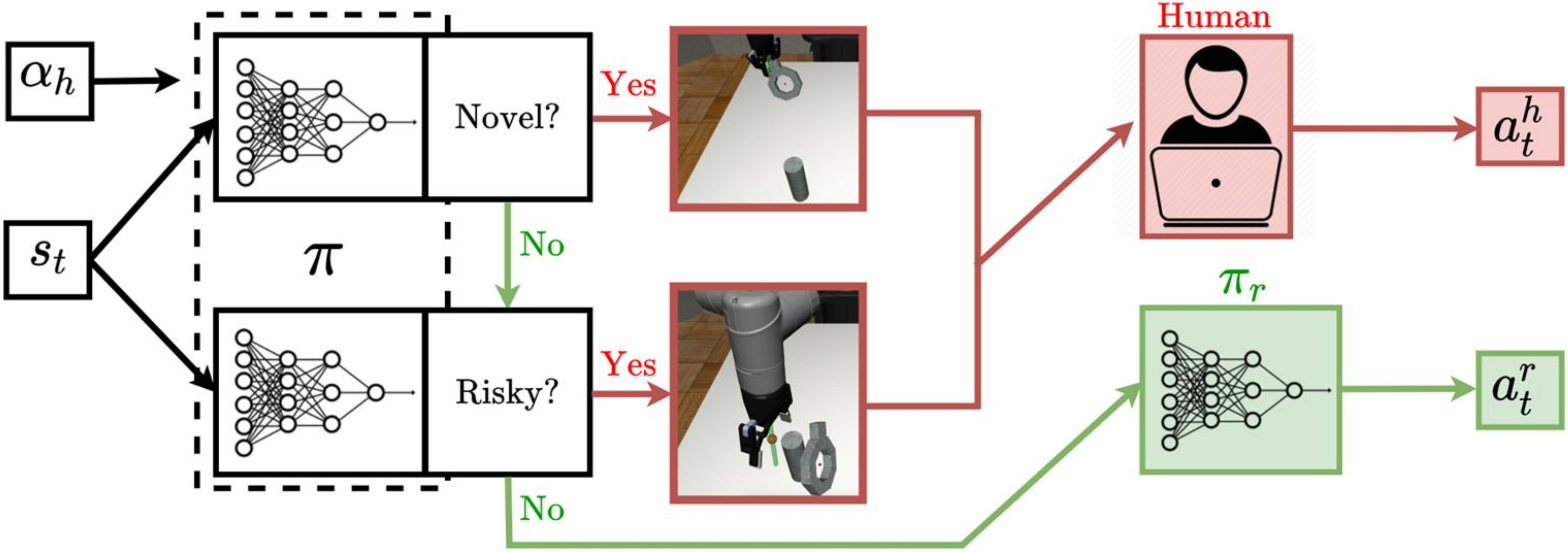
Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

ThriftyDAgger

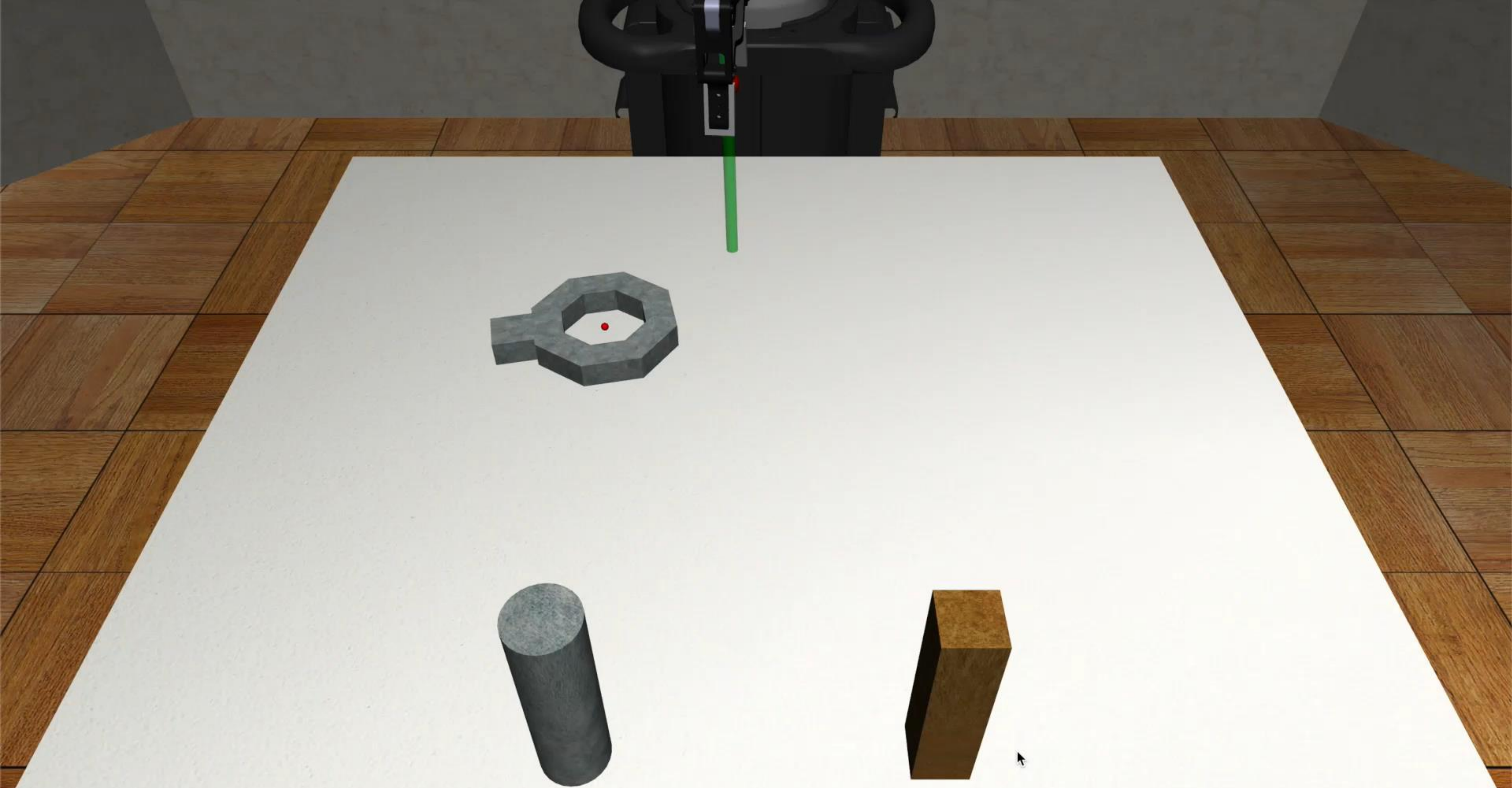


Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

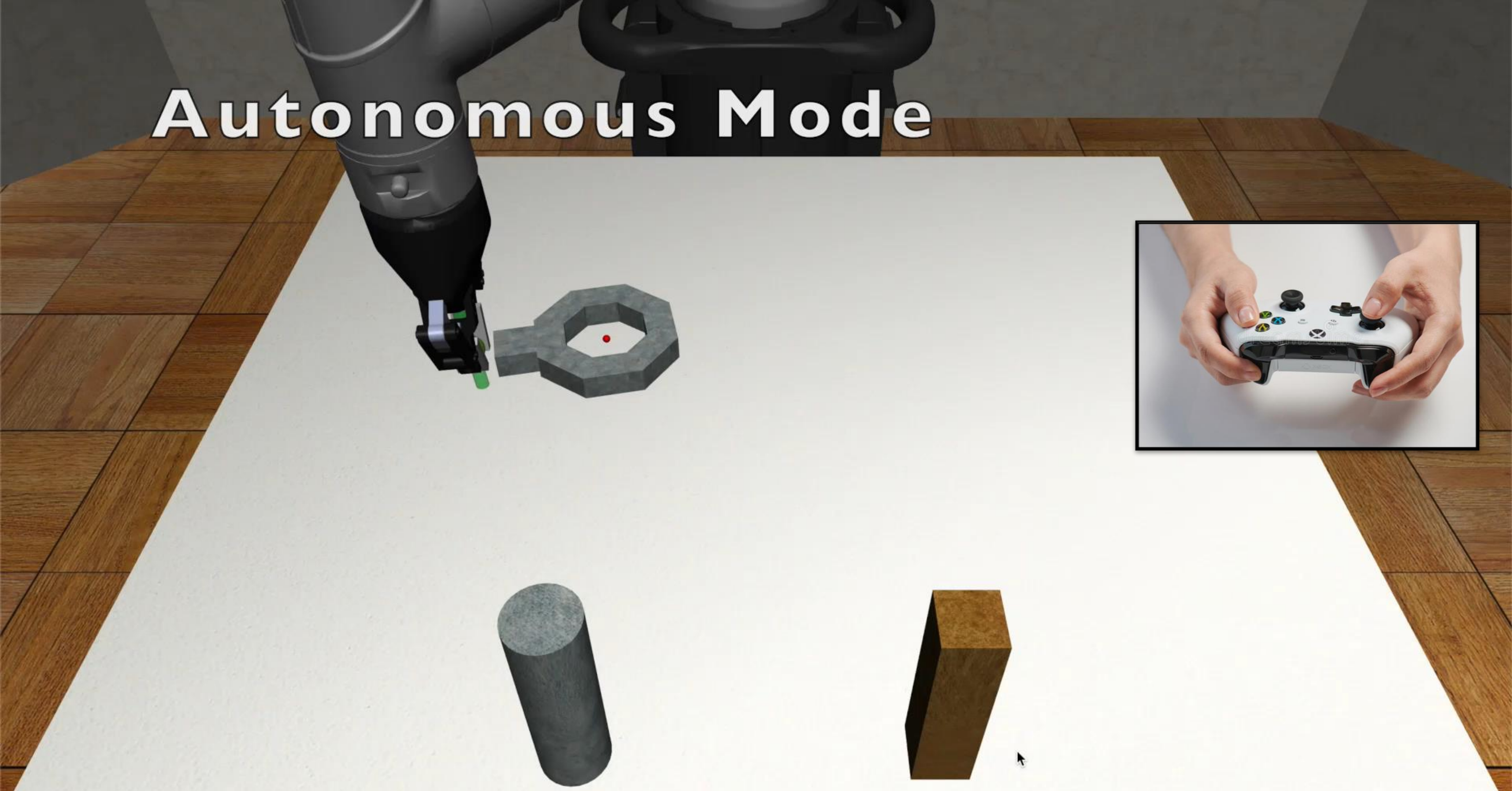
ThriftyDAgger



Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

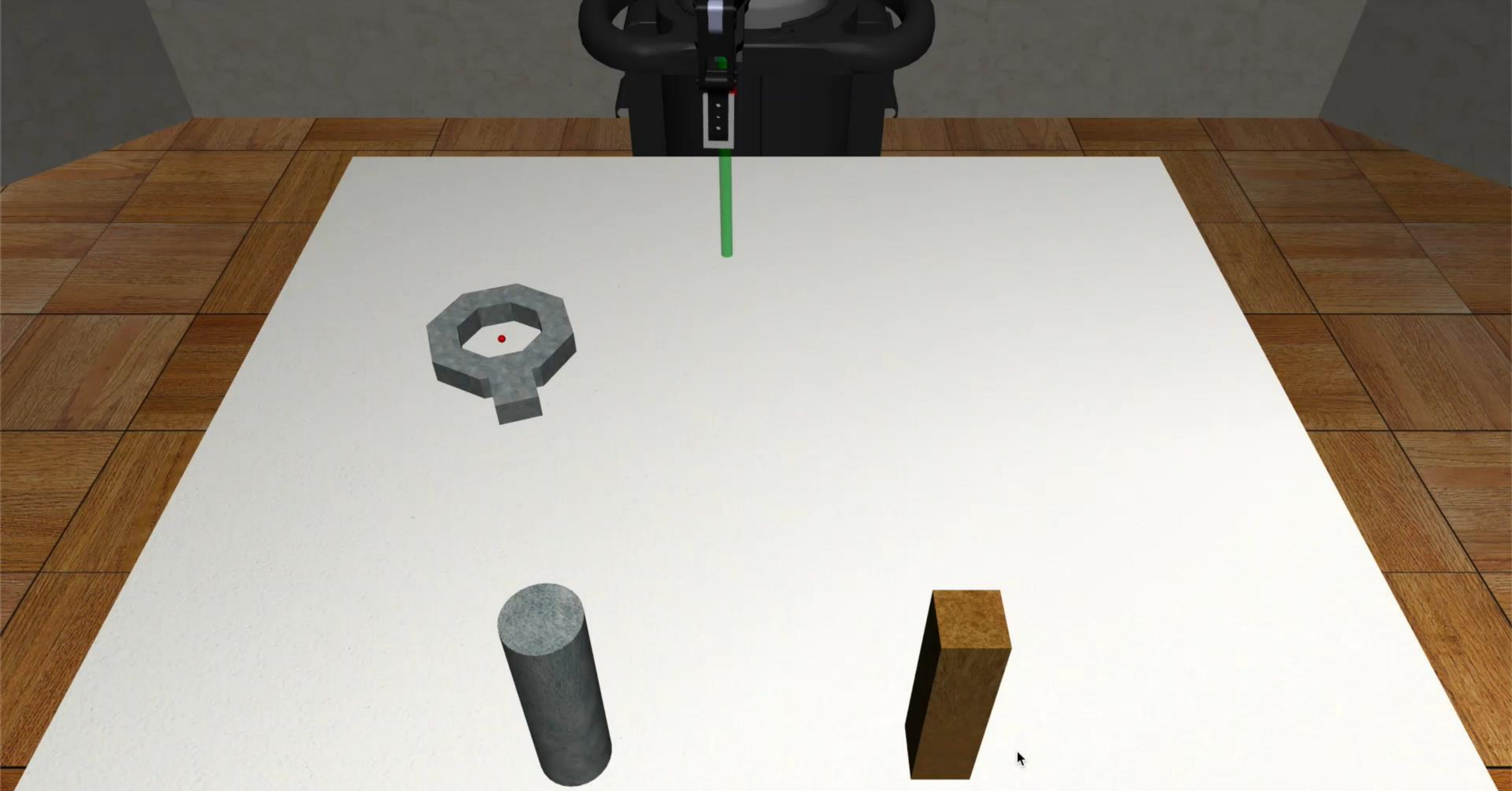


Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

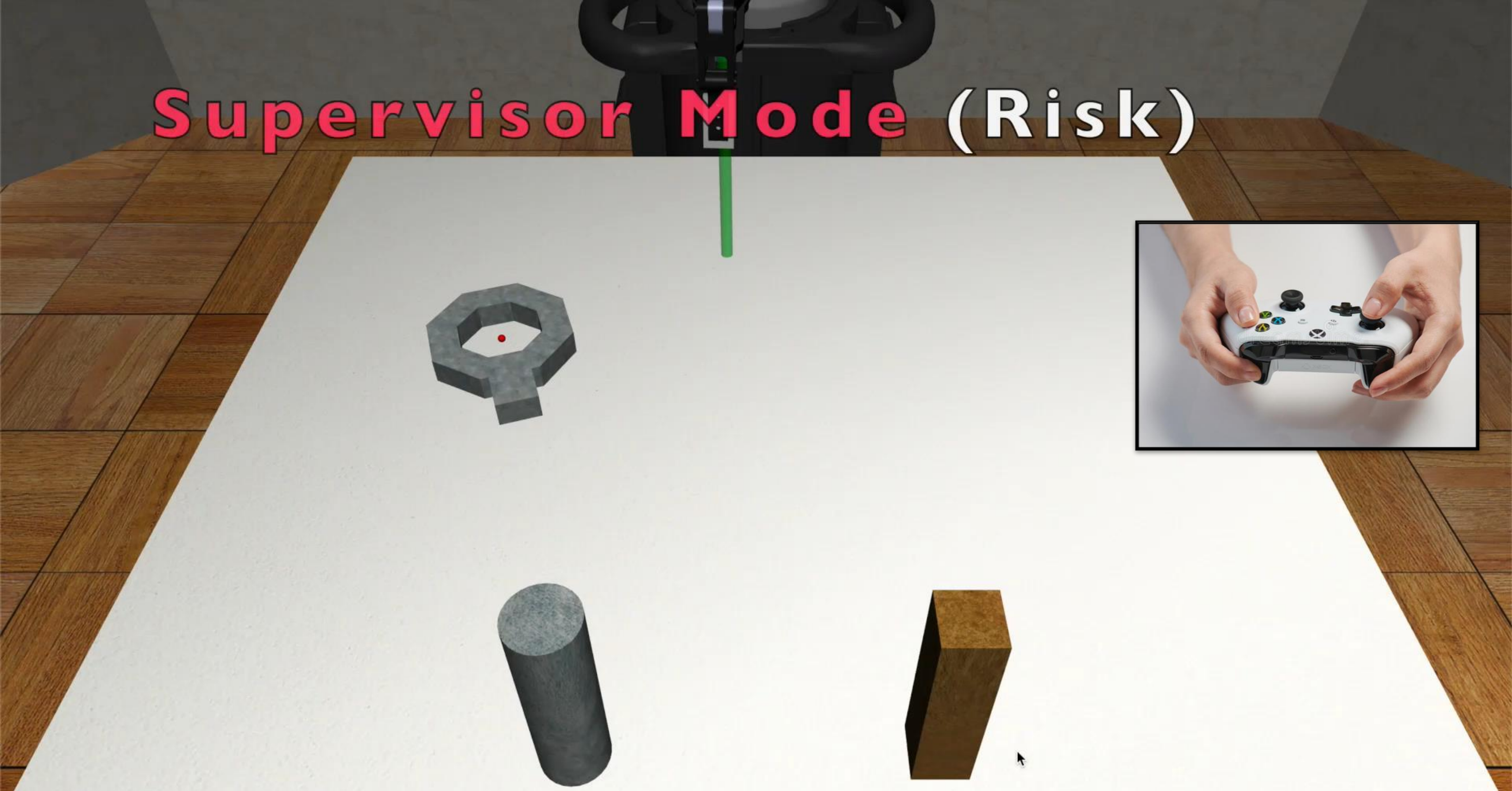


Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

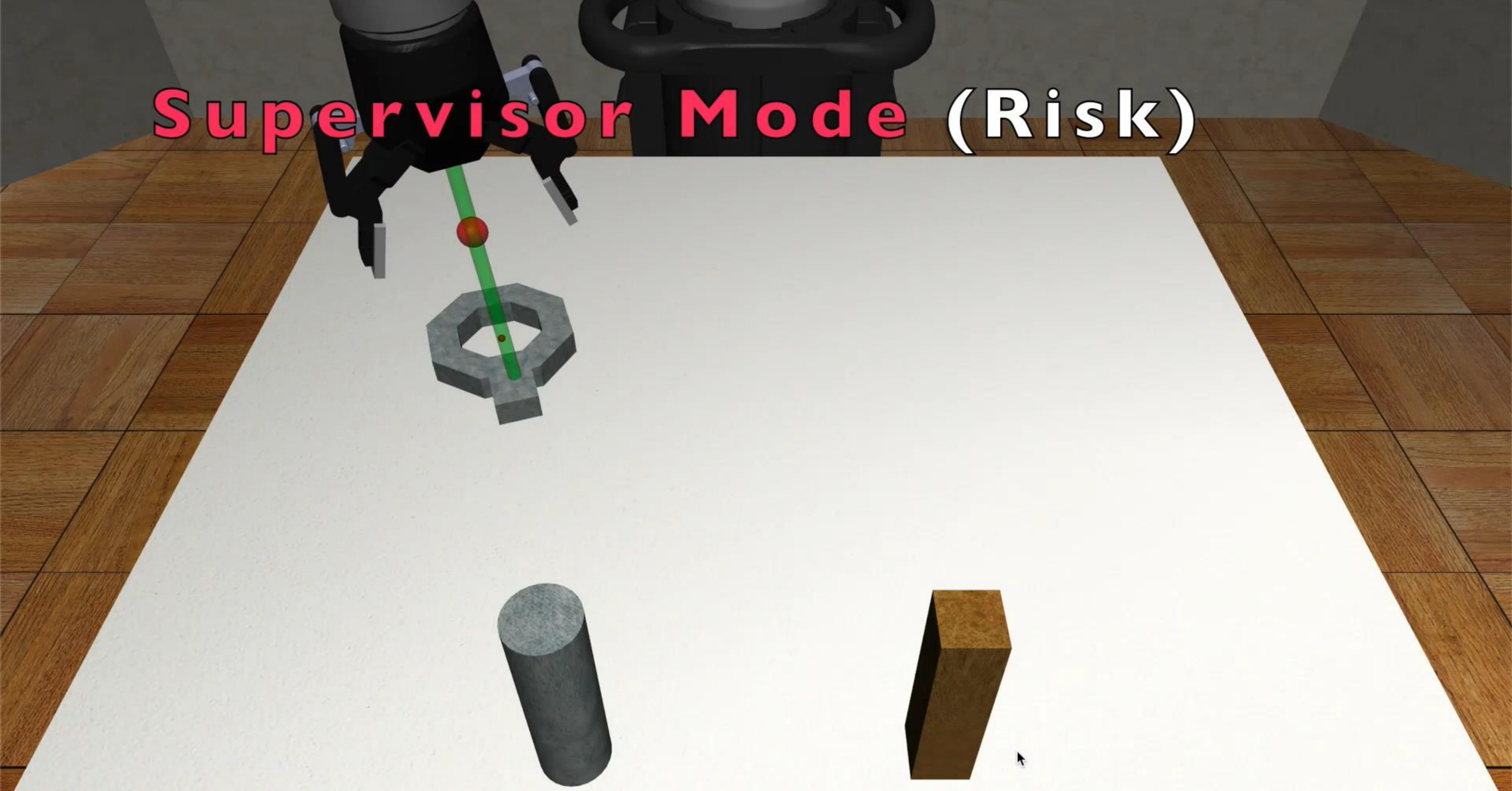
Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.



Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

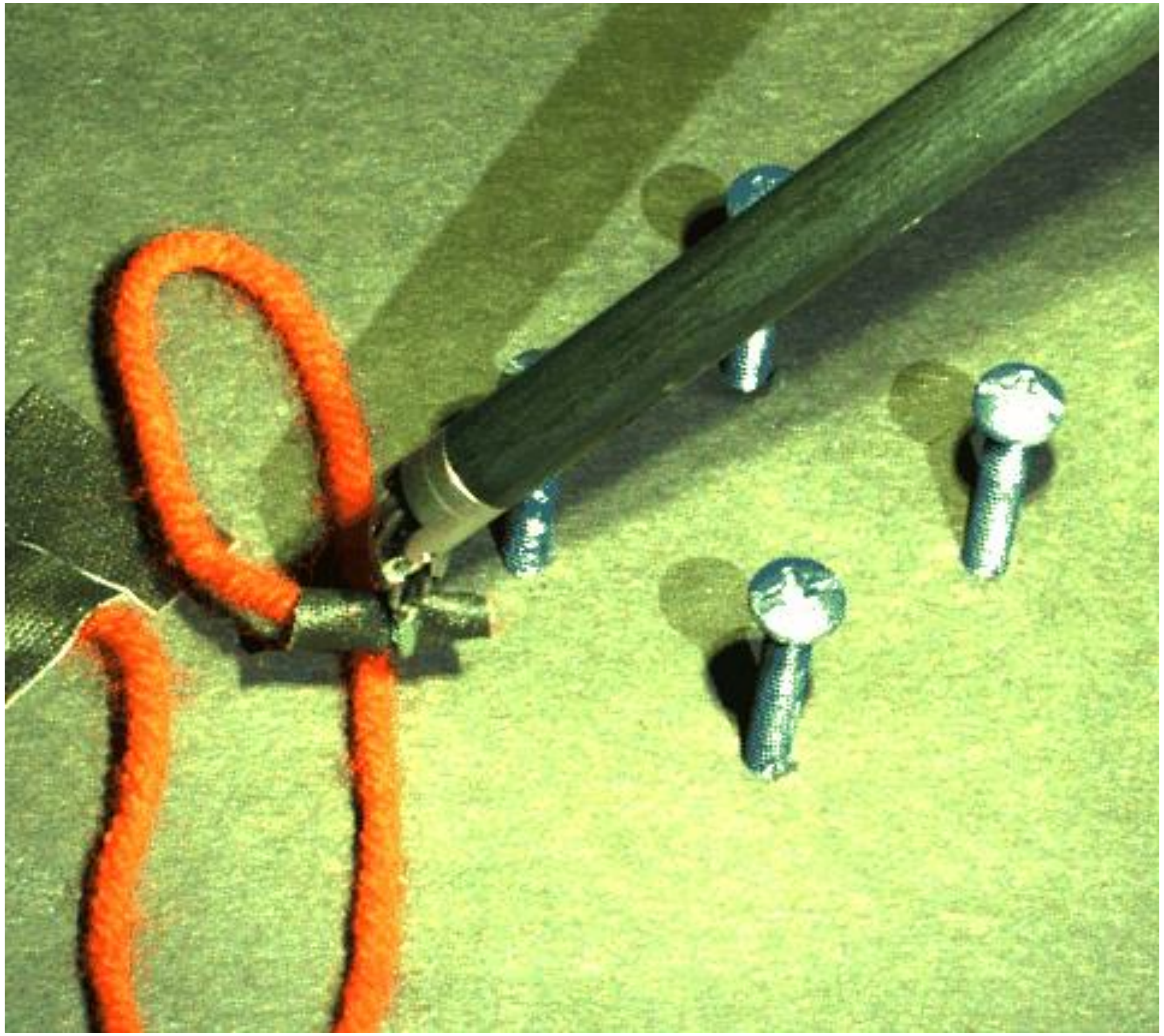


Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

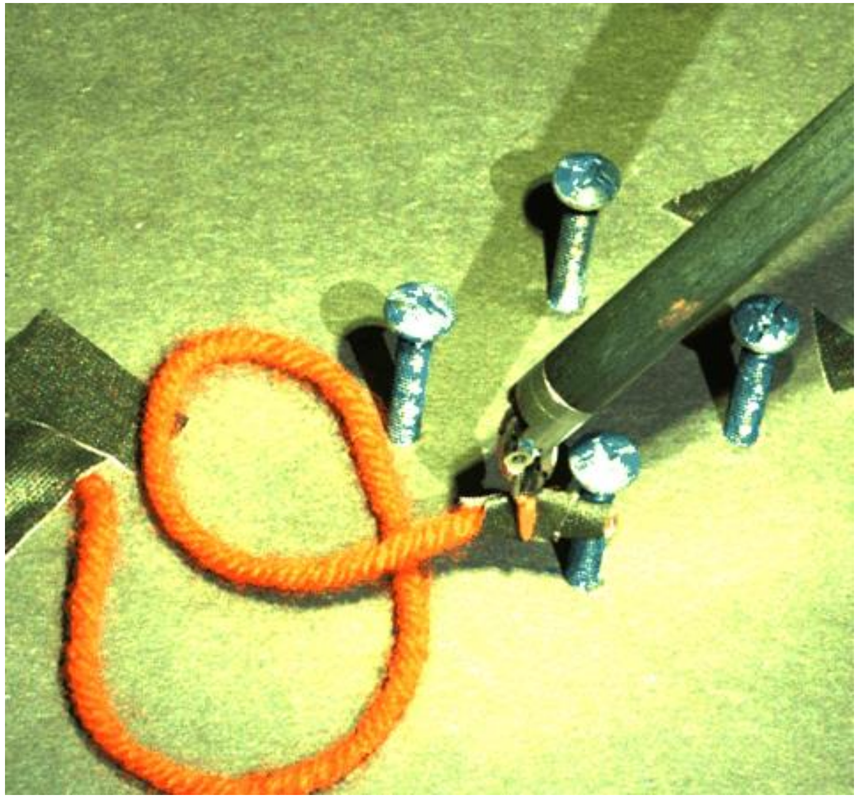


Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

Human Demonstration

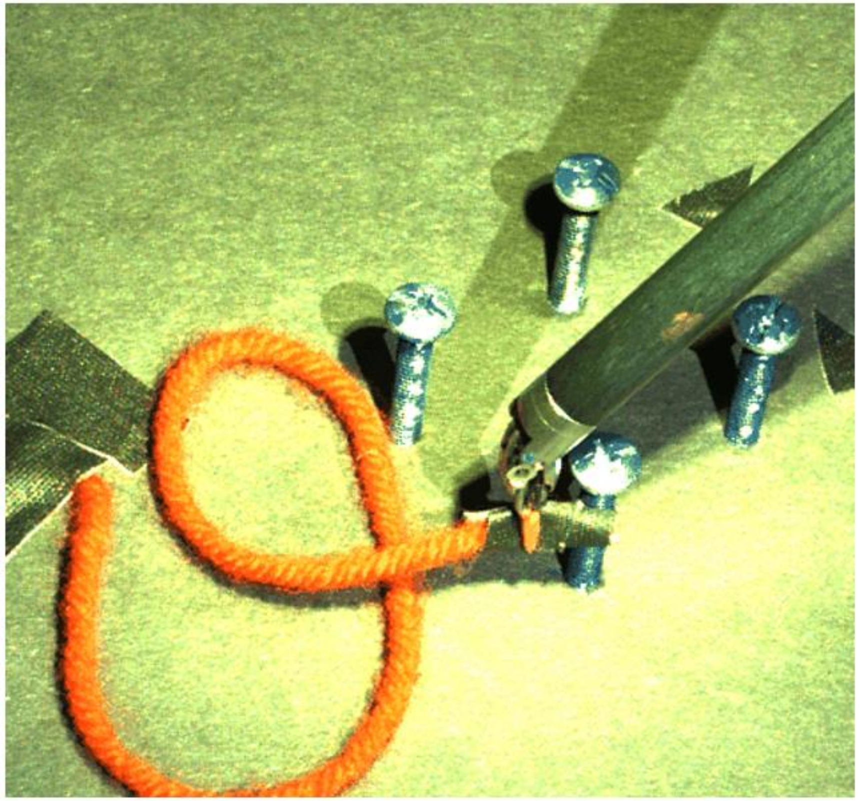


Behavior Cloning

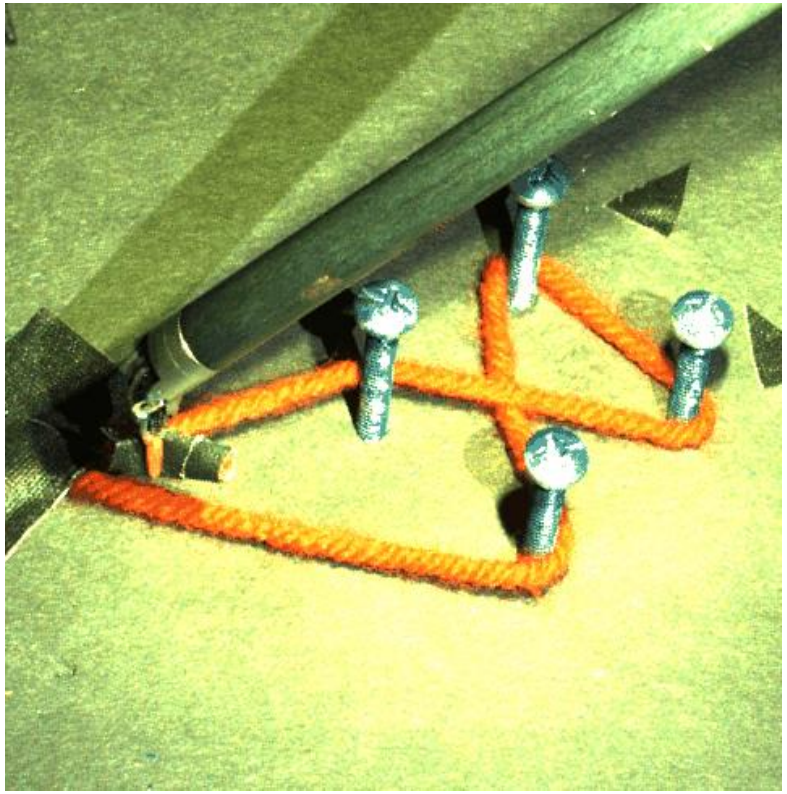


Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

Behavior Cloning

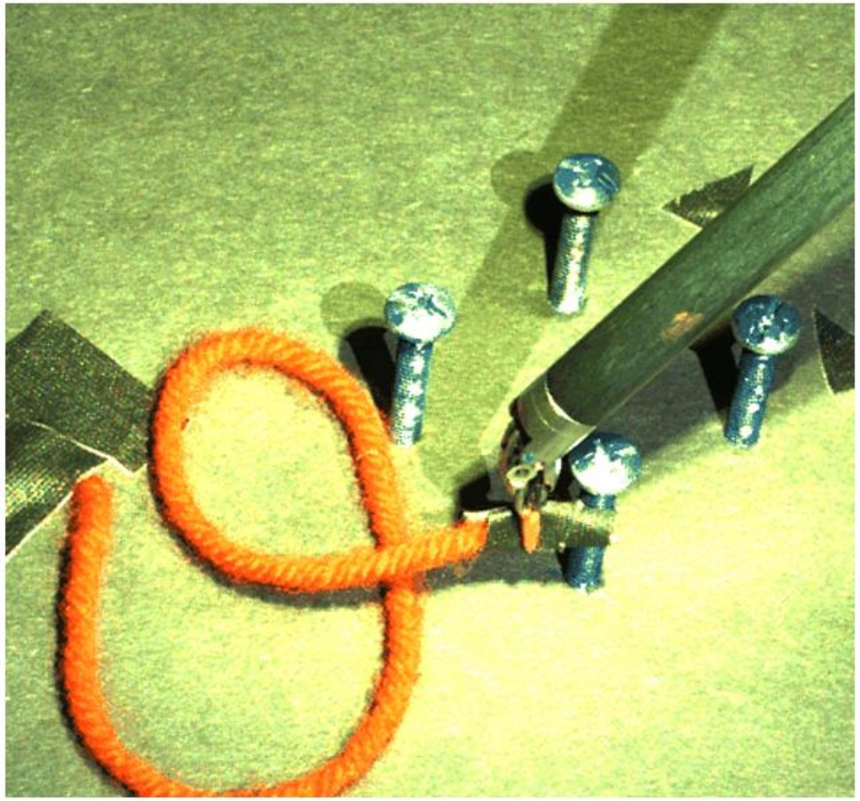


ThriftyDAgger (autonomous)

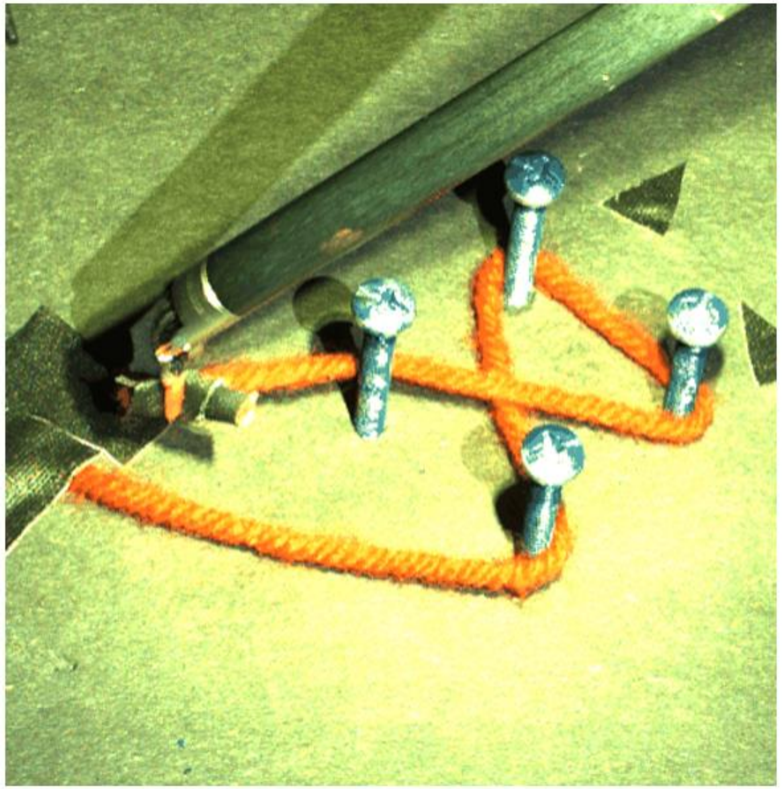


Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

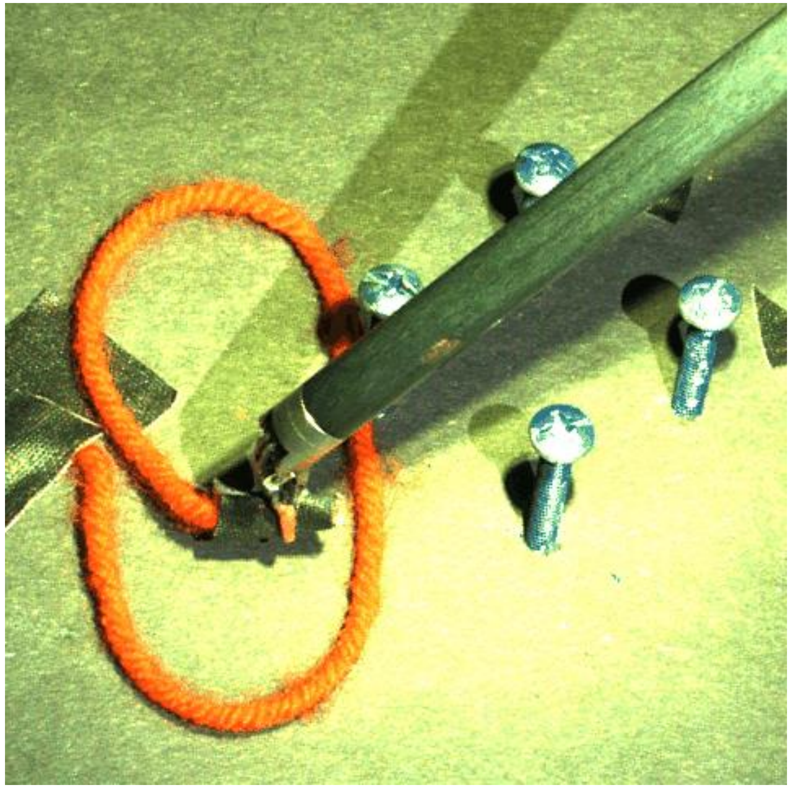
Behavior Cloning



ThriftyDAgger (autonomous)



ThriftyDAgger (+human)

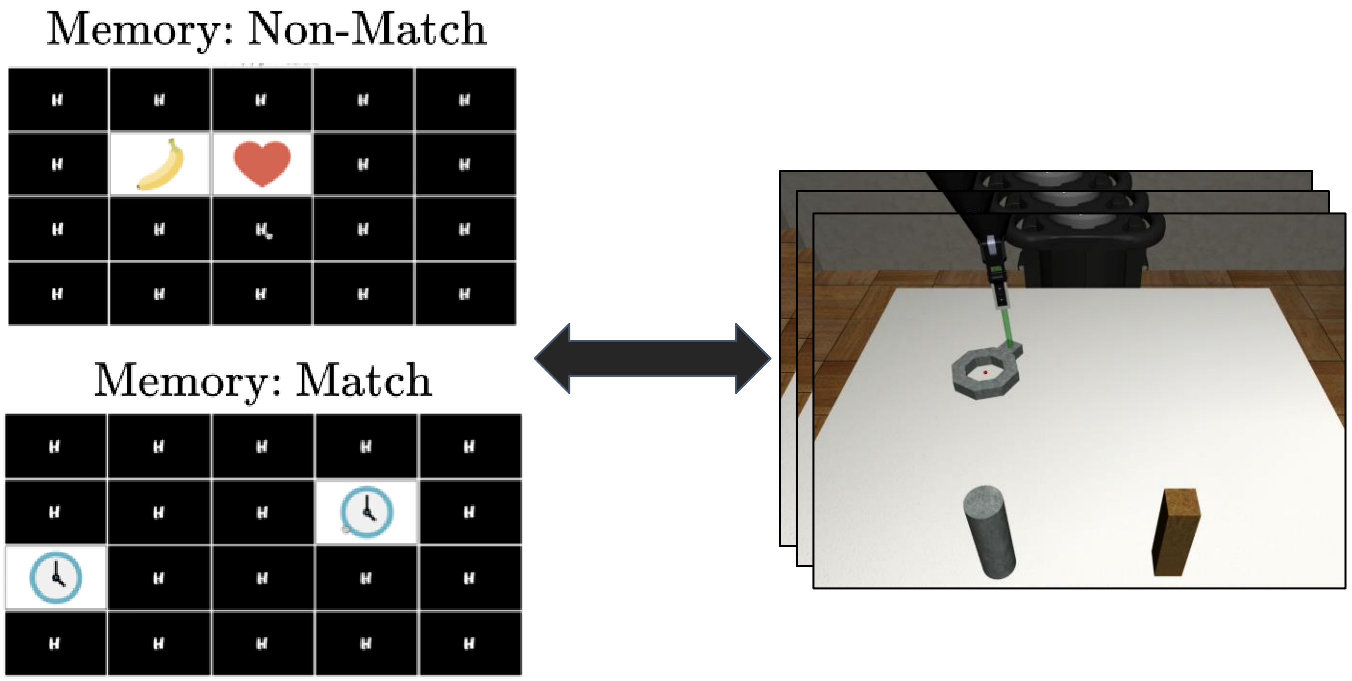


Hoque et al. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning." CoRL 2021.

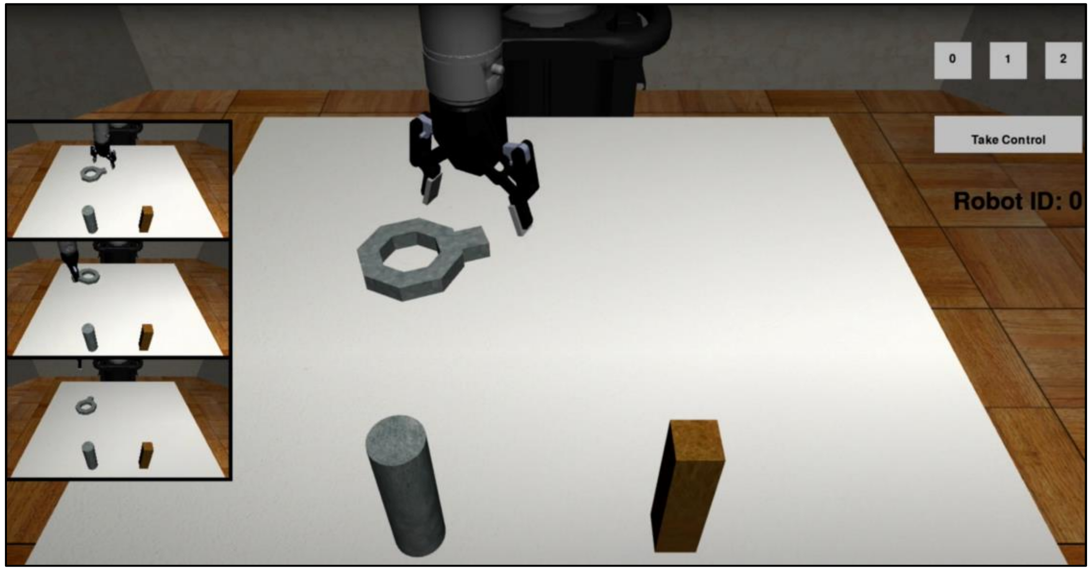
User Study

N=10 subjects each control 3 robots in simulation.

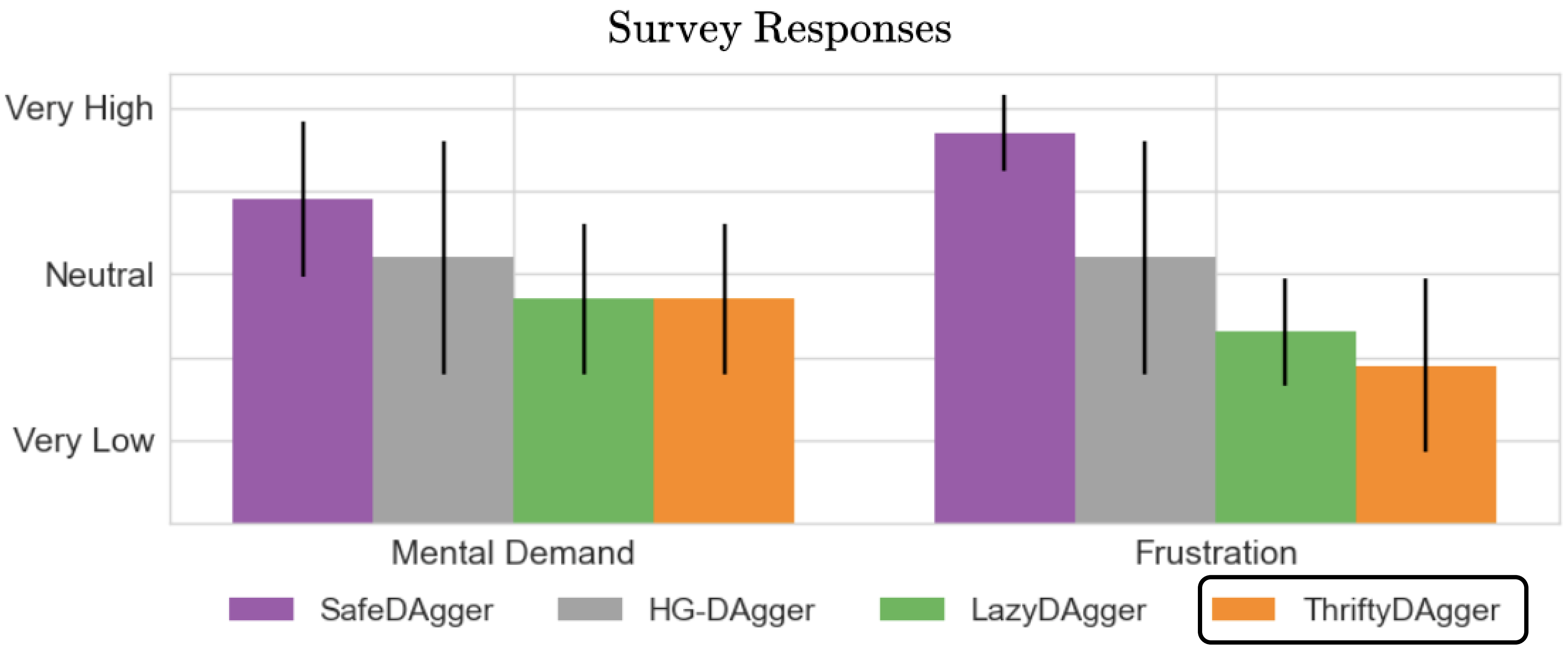
Robot-Gated



Human-Gated



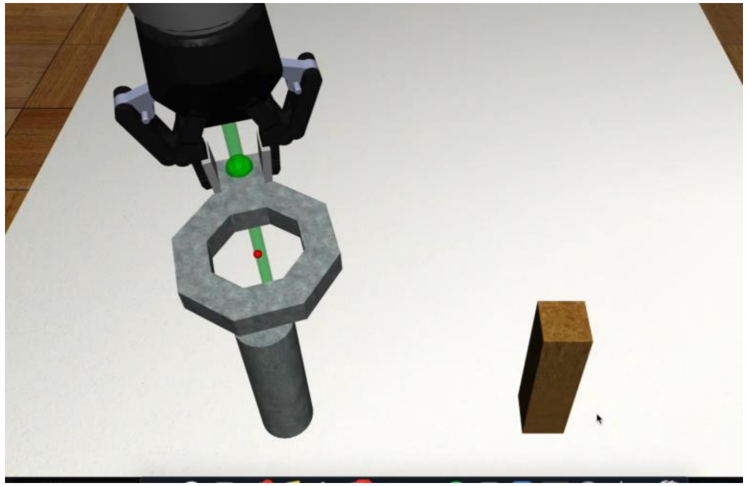
ThriftyDAgger Qualitative Results



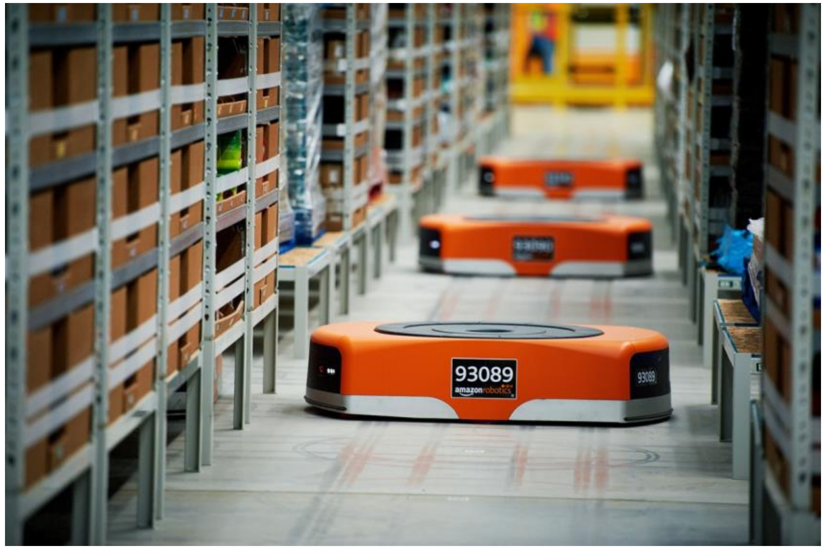
User Study Quantitative Results

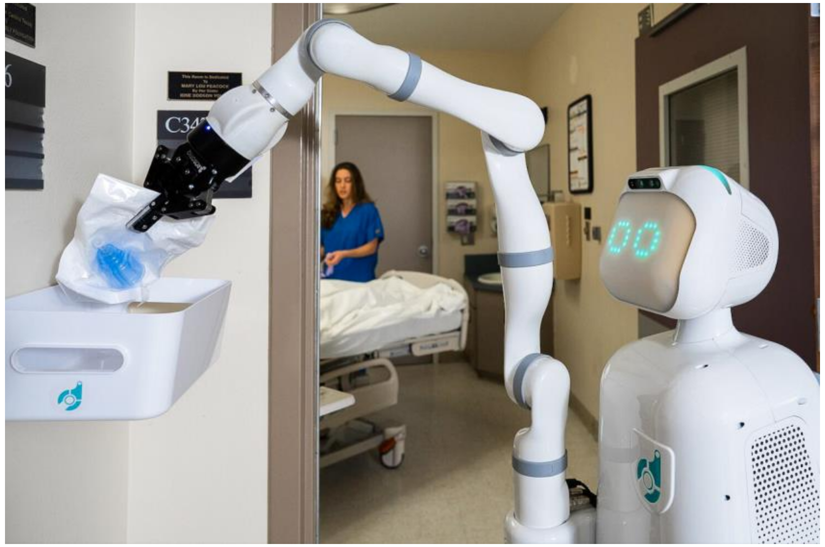
ThriftyDAgger had

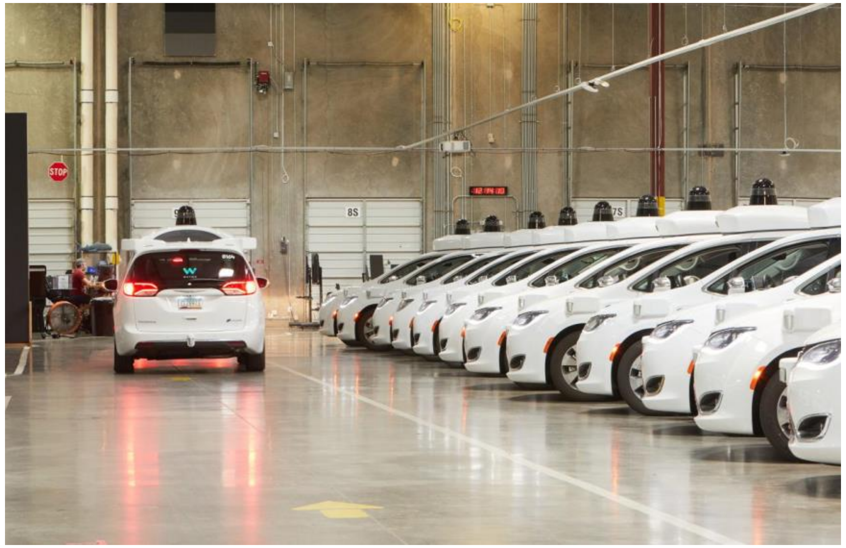
- 21% fewer human interventions
- 57% more concentration pairs found
- 80% more throughput



Scalable and safe robot fleets are possible when robots ask for help in ways that minimize human supervisor burden.







Next Time: Survey of Recent BC methods

- Choose your own adventure reading assignment
  - Implicit Behavior Cloning
  - Action Chunking Transformer
  - Diffusion Policy
- Submit a paragraph before class summarizing at a high-level:
  - What’s the problem the authors want to solve?
  - Why is important?
  - What is their proposed solution?
  - What evidence do they give that their solution is good?
  - What is one question you had about the paper?