PSRO-Based Robust Handover Policy Under Adversarial Perturbations

Boxiang (William) Fu

15-888 (Fall 2025) Computational Game Solving
Carnegie Mellon University

12/01/2025

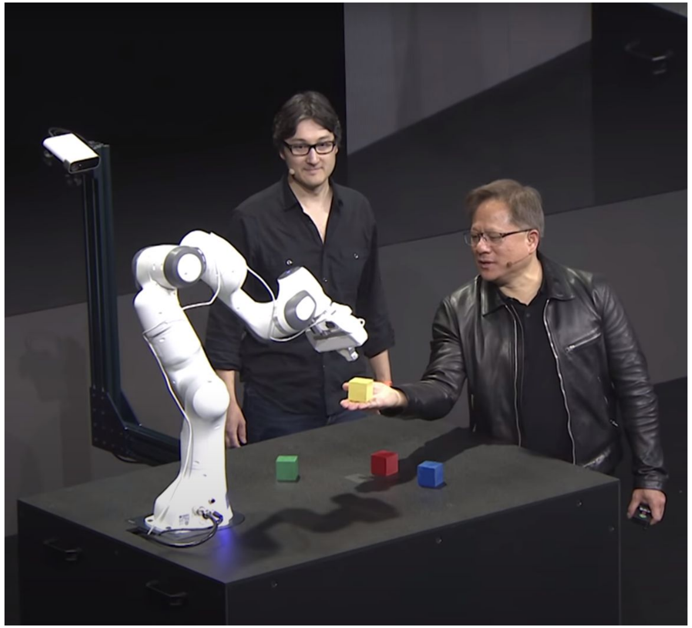
Motivation

Handover: Object transfer from human to robot

Applications: All aspects of human-robot interaction

- Collaborative assembly and manufacturing
- Surgical and rehabilitation assistance
- Collaborative warehousing tasks
- Housekeeping and home assistance

Problems: Not perturbation robust – may fail under external disturbances



Related Work

Rule based:

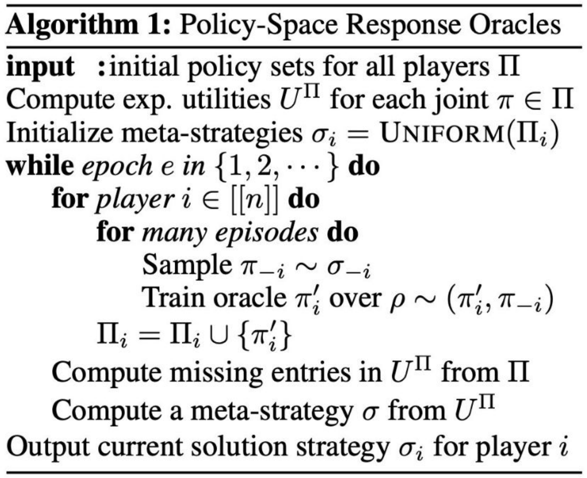
- Franka Emika Watchman: Stop the robot when force sensor detects collision or disturbance

Learning based:

- Adversarial skill learning algorithm using SAC (Jian et al., 2020)

Game theory based:

- PSRO: Computes best response to meta strategy from a restricted two player zero-sum game (Lanctot et al., 2017)
- GRAD: Temporally-coupled perturbations (Liang et al., 2024)



Algorithm 1: Policy-Space Response Oracles

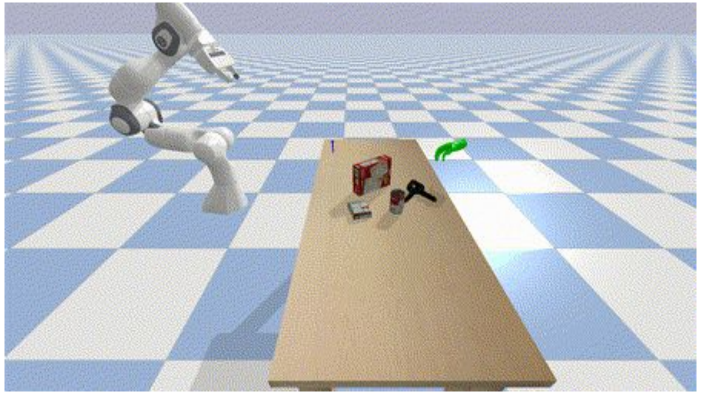
```
input :initial policy sets for all players $\Pi$
Compute exp. utilities $U^{\Pi}$ for each joint $\pi \in \Pi$
Initialize meta-strategies $\sigma_i = \textsc{Uniform}(\Pi_i)$
while epoch $e$ in $\{1, 2, \cdots\}$ do
    for player $i \in [[n]]$ do
        for many episodes do
            Sample $\pi_{-i} \sim \sigma_{-i}$
            Train oracle $\pi'_i$ over $\rho \sim (\pi'_i, \pi_{-i})$
        $\Pi_i = \Pi_i \cup \{\pi'_i\}$
    Compute missing entries in $U^{\Pi}$ from $\Pi$
    Compute a meta-strategy $\sigma$ from $U^{\Pi}$
Output current solution strategy $\sigma_i$ for player $i$
```

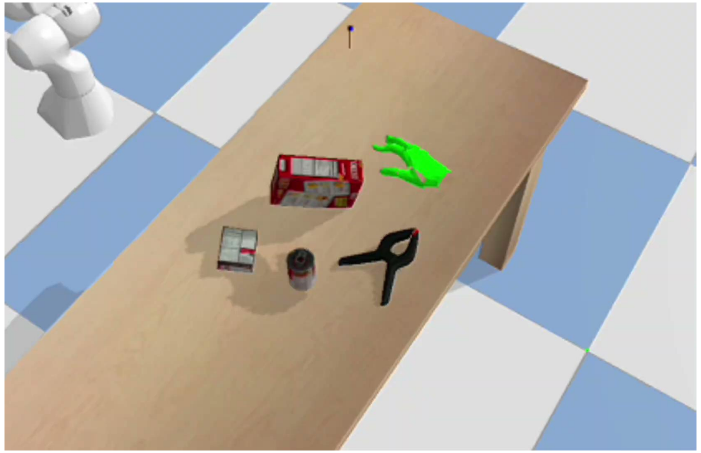
Methodology

- Follows the methodology from GRAD (Liang et al., 2024)
- Two-player zero-sum game of manipulator agent vs. adversarial noise
- Manipulator agent attempts to maximize success rate of object handover from human
- Adversarial noise attempts to minimize success rate through perturbations
- Iterate using meta strategies from manipulator and adversary
- Compute best-responses and add to action set
- Recompute Nash equilibrium meta strategy in new restricted action set

Simulation

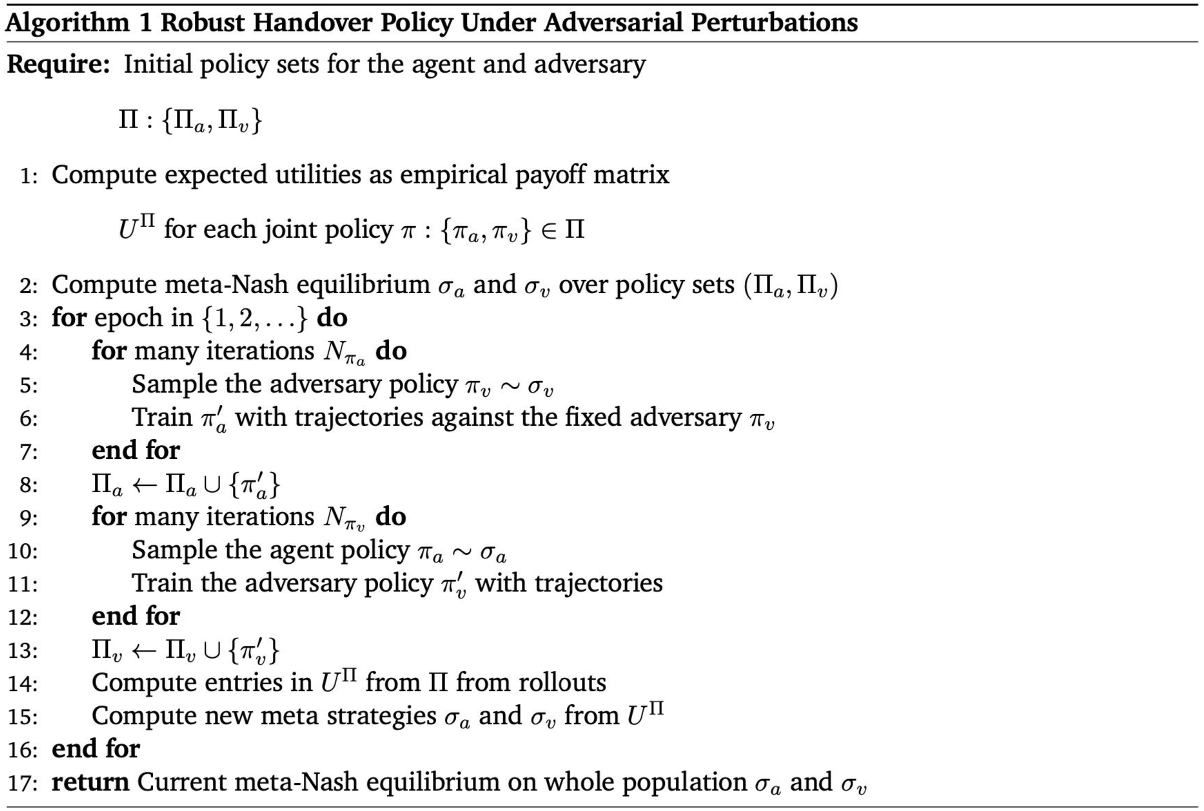
- Handover-Sim from NVlabs using Isaac Gym (https://handover-sim.github.io/)
- Simplified environment to train only on cereal box object
- Manipulator modeled as a Franka Emika Panda with 7 DoF arm and 2 DoF gripper
- Perturbations modelled as Gaussian force (N) and torque (Nm) inputs to the hand & cereal box object over a time period (s)





Implementation

Credit: Pseudo-code template from (Liang et al., 2024)



Algorithm 1 **Robust Handover Policy Under Adversarial Perturbations**

```
Require: Initial policy sets for the agent and adversary
        $\Pi : \{\Pi_a, \Pi_v\}$
 1: Compute expected utilities as empirical payoff matrix
        $U^{\Pi}$ for each joint policy $\pi : \{\pi_a, \pi_v\} \in \Pi$
 2: Compute meta-Nash equilibrium $\sigma_a$ and $\sigma_v$ over policy sets $(\Pi_a, \Pi_v)$
 3: for epoch in $\{1, 2, \dots\}$ do
 4:     for many iterations $N_{\pi_a}$ do
 5:         Sample the adversary policy $\pi_v \sim \sigma_v$
 6:         Train $\pi'_a$ with trajectories against the fixed adversary $\pi_v$
 7:     end for
 8:     $\Pi_a \leftarrow \Pi_a \cup \{\pi'_a\}$
 9:     for many iterations $N_{\pi_v}$ do
10:         Sample the agent policy $\pi_a \sim \sigma_a$
11:         Train the adversary policy $\pi'_v$ with trajectories
12:     end for
13:     $\Pi_v \leftarrow \Pi_v \cup \{\pi'_v\}$
14:     Compute entries in $U^{\Pi}$ from $\Pi$ from rollouts
15:     Compute new meta strategies $\sigma_a$ and $\sigma_v$ from $U^{\Pi}$
16: end for
17: return Current meta-Nash equilibrium on whole population $\sigma_a$ and $\sigma_v$
```

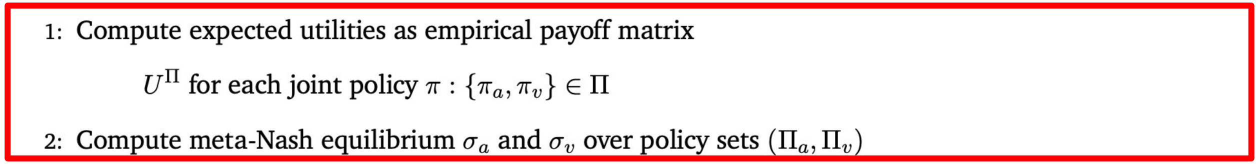
Implementation – Initialization

Require: Initial policy sets for the agent and adversary

$$\Pi : \{\Pi_a, \Pi_v\}$$

- **Agent:** Trained handover policy at 40,000 epochs against no disturbance adversary (single agent RL problem)
- **Adversary:** No disturbance

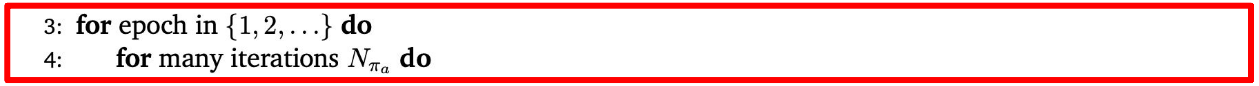
Implementation – Initialization



```
1: Compute expected utilities as empirical payoff matrix
       $U^{\Pi}$ for each joint policy $\pi : \{\pi_a, \pi_v\} \in \Pi$
2: Compute meta-Nash equilibrium $\sigma_a$ and $\sigma_v$ over policy sets $(\Pi_a, \Pi_v)$
```

- Snapshot policy has empirical handover success rate of 42%
- Both agent and adversary only have a single available action
- Nash equilibrium is for both to play that action with probability 1

Implementation – Looping

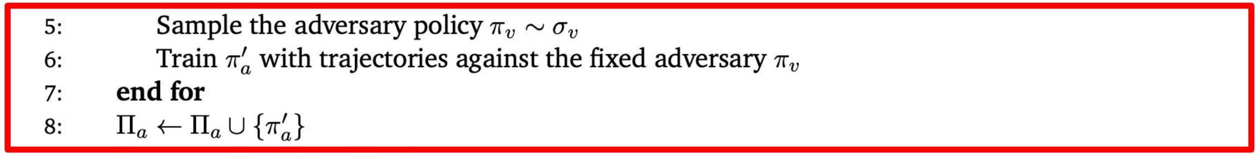


```
3: for epoch in $\{1, 2, \ldots\}$ do
4:     for many iterations $N_{\pi_a}$ do
```

- Capped maximum number of epochs to 9 mainly due to compute limits
- Agent handover policy trained over 40,000 epochs (allows comparison with initial policy)

Implementation – Agent

- Deep Deterministic Policy Gradient (DDPG) RL (Lillicrap et al. 2016)
- Augmented by Goal-Auxiliary Actor-Critic (GA-DDPG) (Wang et al. 2021)
- Trained both online by interacting with the environment and offline using stored replay buffer

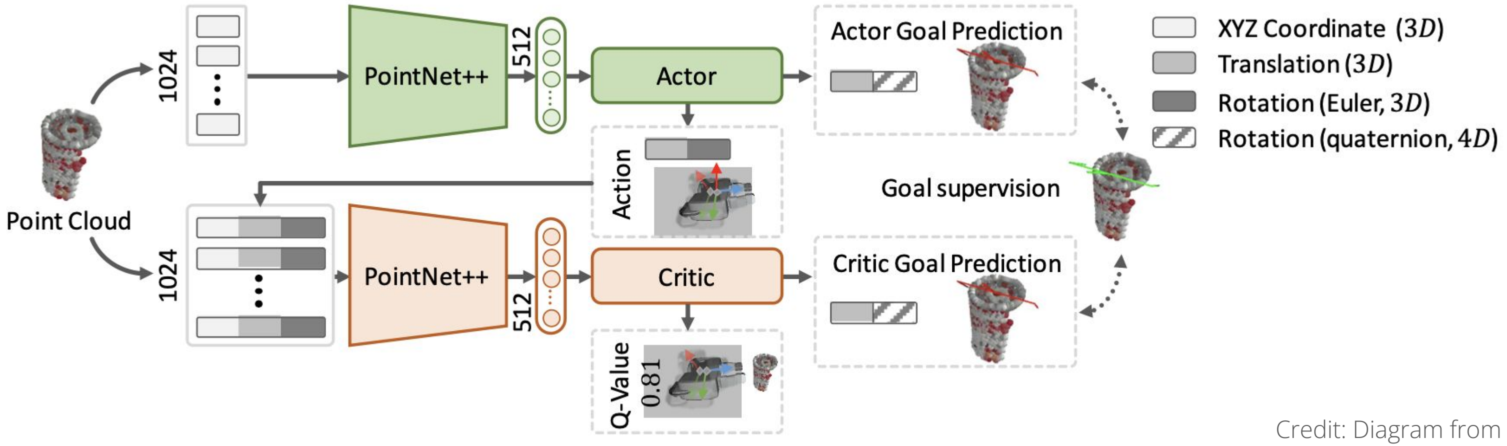


```
5:         Sample the adversary policy $\pi_v \sim \sigma_v$
6:         Train $\pi'_a$ with trajectories against the fixed adversary $\pi_v$
7:     end for
8:     $\Pi_a \leftarrow \Pi_a \cup \{\pi'_a\}$
```

- Perturbations added based on adversary's disturbance policy
- Simplified agent policy task to only hand over cereal box object (due to compute limits)

Implementation Details – GA-DDPG

1. Represent state as point cloud and action as end-effector pose
2. Generate expert labels using classical planner. Obtain action and final grasp goal for each timestep
3. Train actor & critic networks with DDPG
4. Add goal-auxiliary heads to both actor and critic

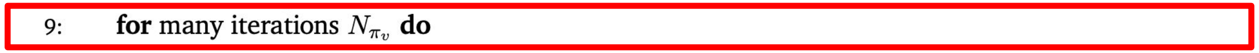


Credit: Diagram from (Wang et al., 2021)

Carnegie Mellon University
Robotics Institute

Implementation – Looping

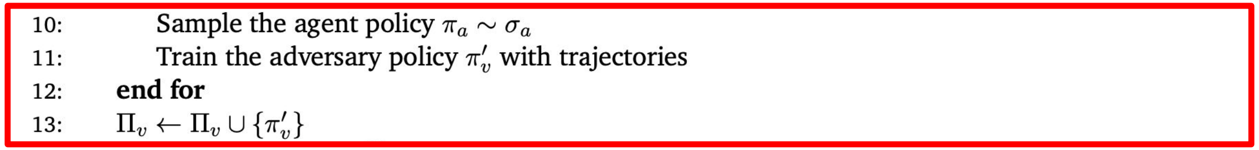
- Adversary policy trained over 15 iterations



```
9:    for many iterations $N_{\pi_v}$ do
```

Implementation – Adversary

- Black box Bayesian optimization over valid perturbations to find parameters that minimize expected success rate of agent meta strategy
- 6 infinitely divisible “credits” that can be used on perturbation standard deviation for Force (Newtons), Torque (Newton-meters), or Duration (seconds)
- For example, [1.0, 1.0, 1.0] is a standard deviation of 1 Newton force & 1 Newton-meter torque applied to the cereal box + hand object for 1 second

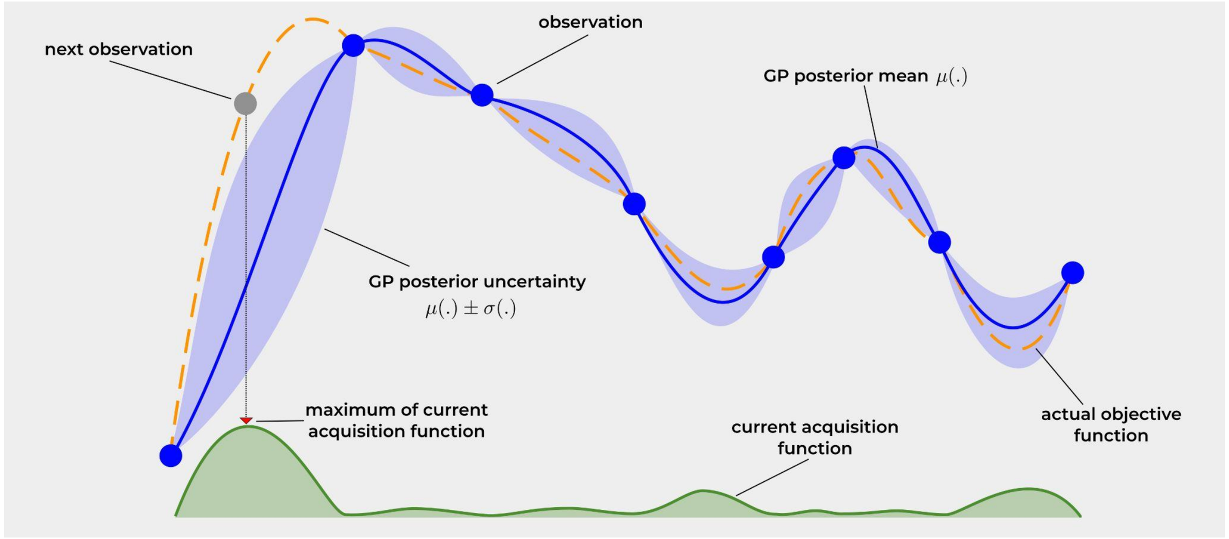


```
10:         Sample the agent policy $\pi_a \sim \sigma_a$
11:         Train the adversary policy $\pi'_v$ with trajectories
12:     end for
13:     $\Pi_v \leftarrow \Pi_v \cup \{\pi'_v\}$
```

- Cap each axis at maximum 2.0 credits that can be allotted

Implementation Details – Bayesian Optimization

1. Build a surrogate model that predicts performance of hyperparameters
2. Use an acquisition function to choose the next hyperparameter point to evaluate
3. Iteratively update the surrogate with each new observation
4. Run until we obtain good hyperparameter estimates

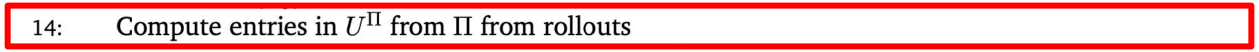


observation
next observation
GP posterior mean $\mu(.)$
GP posterior uncertainty
$\mu(.) \pm \sigma(.)$
maximum of current acquisition function
current acquisition function
actual objective function

Credit: Diagram from (Al-Hafez, 2021)

Implementation – Payoff Matrix

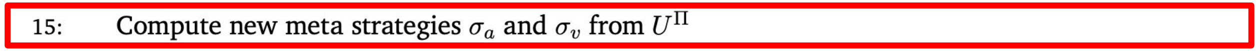
- Perform 100 rollouts for each agent-adversary pair and calculate empirical handover success rate
- Populate payoff matrix using empirical rates



```
14:    Compute entries in $U^{\Pi}$ from $\Pi$ from rollouts
```

Implementation – Meta Strategy

- Compute two-player zero-sum game Nash equilibrium using linear programming



```
15:    Compute new meta strategies $\sigma_a$ and $\sigma_v$ from $U^{\text{II}}$
```

Implementation Details – Nash from LP

LP for row player

$$\begin{aligned} \text{minimize} \quad & t \\ \text{subject to} \quad & t \geq \boldsymbol{x}^\top \mathbf{A}[:, a_2] \text{ for all } a_2 \in \mathcal{A}_2, \\ & \sum_{a_1 \in \mathcal{A}_1} \boldsymbol{x}[a_1] = 1, \\ & \boldsymbol{x} \geq 0. \end{aligned}$$

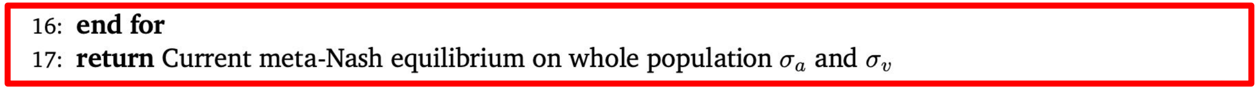
LP for column player

$$\begin{aligned} \text{maximize} \quad & t \\ \text{subject to} \quad & t \leq \boldsymbol{y}^\top \mathbf{A}[a_1, :] \text{ for all } a_1 \in \mathcal{A}_1, \\ & \sum_{a_2 \in \mathcal{A}_2} \boldsymbol{y}[a_2] = 1, \\ & \boldsymbol{y} \geq 0. \end{aligned}$$

Credit: Diagram from (Anagnostides & Sandholm, 2025)

Implementation – Return

- A probability vector for both agent and adversary on their meta-strategies
- A vector for the agent containing the Neural Network for each trained policy
- A vector for the adversary containing the perturbations used for each action



```
end for
return Current meta-Nash equilibrium on whole population $\sigma_a$ and $\sigma_v$
```

Training

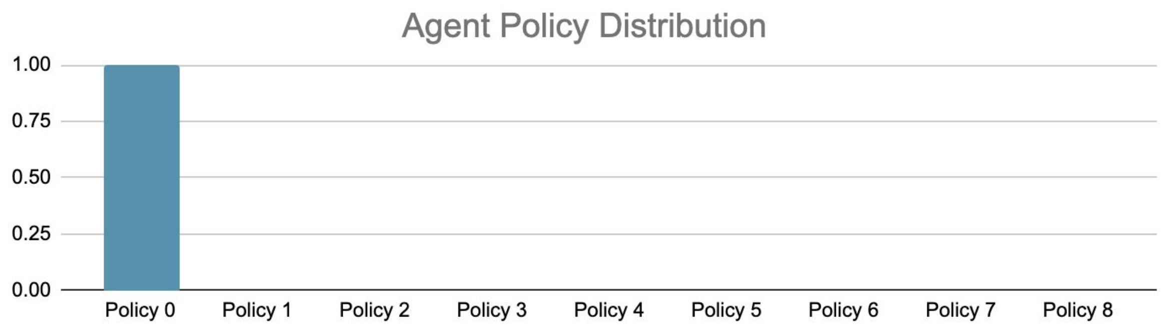
- ~1.5 days on a NVIDIA L4 GPU and 128 GB RAM on Google Cloud
- 9 x 40,000 = 360,000 agent policy updates
- 9 x 15 = 135 adversary parameter updates
- (285+135) x 100 = 42,000 agent vs. adversary evaluations

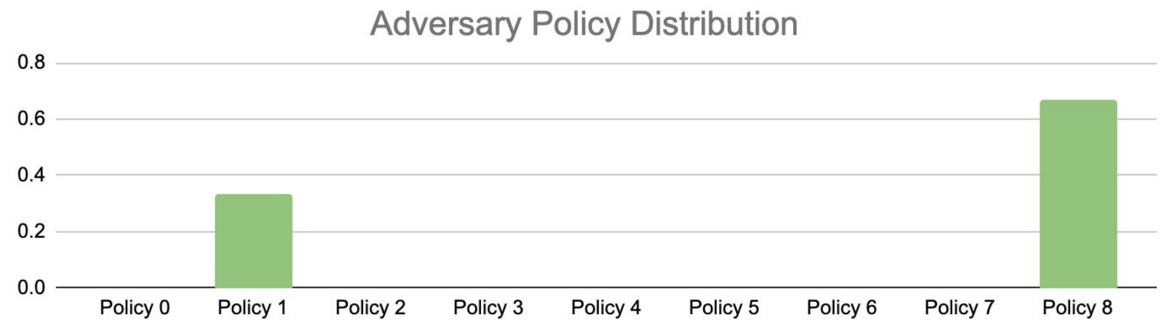
Results

However…

Training was not very successful…

Results – Meta Strategy





Observations:

- Final meta strategy chooses initial agent policy with probability 1
- Agent was not learning against adversary in PSRO iterations

Adversary Disturbances:

- Policy 1:
  - Prob: 0.3333
  - Dist: [0.62, 1.72, 0.97]
- Policy 8:
  - Prob: 0.6667
  - Dist: [1.06, 1.74, 0.66]

Results – Game Payoff

Game Payoff Matrix

$$\begin{bmatrix} 0.6 & 0.5 & 0.6 & 0.6 & 0.6 & 0.6 & 0.7 & 0.8 & 0.5 \\ 0.1 & 0.4 & 0.2 & 0.2 & 0.5 & 0.1 & 0.1 & 0.3 & 0.2 \\ 0.4 & 0.2 & 0.4 & 0.3 & 0.3 & 0.5 & 0.5 & 0.2 & 0.5 \\ 0.1 & 0.2 & 0.1 & 0.5 & 0.1 & 0.1 & 0.1 & 0.3 & 0.4 \\ 0.1 & 0.3 & 0.2 & 0.1 & 0.3 & 0.3 & 0.3 & 0.2 & 0.1 \\ 0.4 & 0.3 & 0.4 & 0.4 & 0.2 & 0.3 & 0.4 & 0.2 & 0.5 \\ 0.5 & 0.3 & 0.4 & 0.6 & 0.7 & 0.6 & 0.7 & 0.7 & 0.6 \\ 0.1 & 0.6 & 0.3 & 0.1 & 0.4 & 0.2 & 0.3 & 0.2 & 0.2 \\ 0.4 & 0.5 & 0.4 & 0.4 & 0.4 & 0.4 & 0.5 & 0.3 & 0.3 \end{bmatrix}$$

Game Value: 0.50

Results – Success Rate

- **Task:** Obtain cereal box object from human hand without touching the hand and/or dropping the object

- **Adversary Strategy:** Random sampling from its 2 actions in meta strategy
- **Agent Success Rate:** 34%

- **Adversary Strategy:** No perturbation
- **Agent Success Rate:** 42%

Meta strategy from PSRO is only benefitting the adversary!

It is not giving us a better agent policy!

Observations

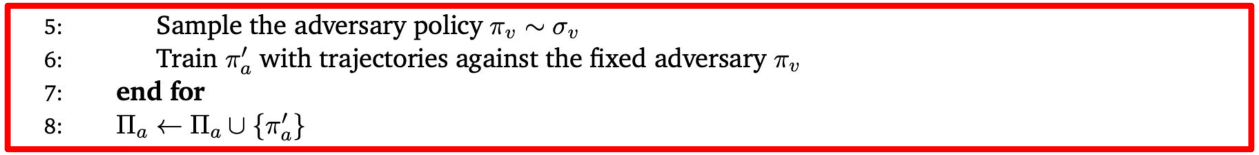
- Aside from the initial policy, the learned agent policies have zero probability of being chosen
- Zero probability policies often have less than 20% handover success (even under no disturbances)
- 40,000 epochs to train GA-DDPG may not be enough to even learn what the task is about (let alone train against adversary)
- Expert policy from Wang et al. (2021) was trained on 150,000 epochs
- By chance, initial handover policy is a very strong policy, could be very hard to surpass under random reinitialization
- Trying to learn to "reinvent the wheel" every time the engine is shut off

Checkpoint Pre-Training

- Instead of restarting the agent policy for each PSRO iteration from random/uniform policy, start from a baseline policy checkpoint (trained on zero perturbations) that has already learned the handover task
- Train on top of this policy to additionally learn the adversarial perturbation

Implementation – Agent

- Deep Deterministic Policy Gradient (DDPG) RL (Lillicrap et al. 2016)
- Augmented by Goal-Auxiliary Actor-Critic (GA-DDPG) (Wang et al. 2021)



```
5:        Sample the adversary policy $\pi_v \sim \sigma_v$
6:        Train $\pi'_a$ with trajectories against the fixed adversary $\pi_v$
7:    end for
8:    $\Pi_a \leftarrow \Pi_a \cup \{\pi'_a\}$
```

- (New) Initialize agent policy with baseline checkpoint against a no disturbance adversary
- (New) Train additional 20,000 epochs against adversary disturbance

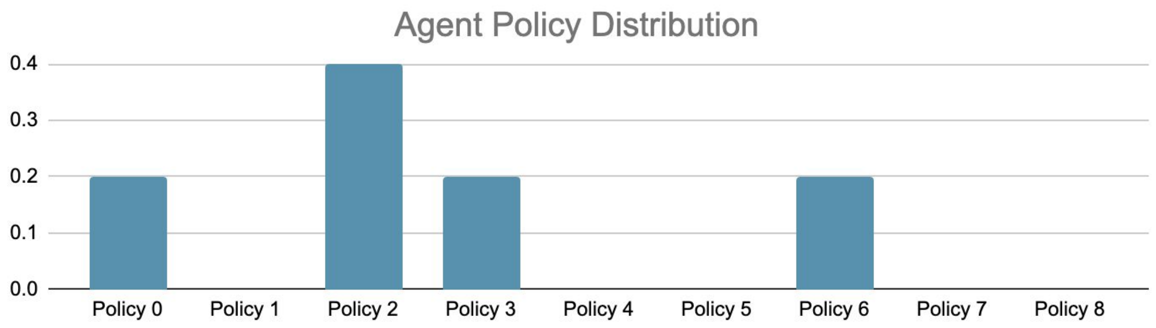
Training

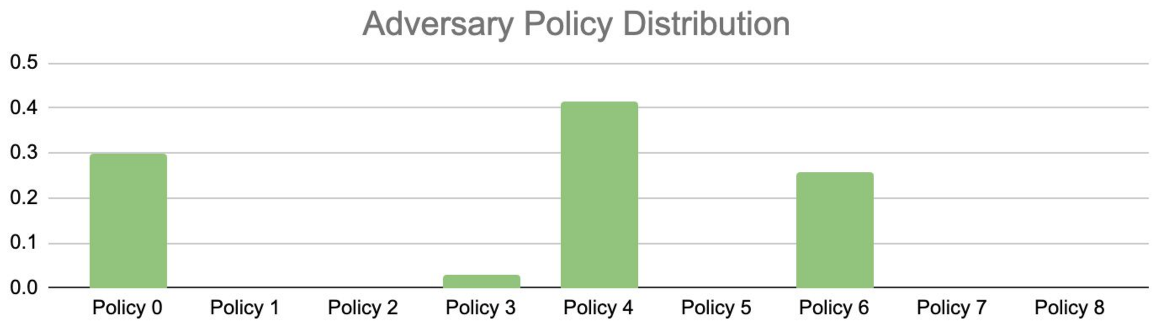
- ~1.5 days on a NVIDIA RTX 3050 GPU and 16 GB RAM laptop
- 9 x 20,000 = 180,000 agent policy updates
- 9 x 15 = 135 adversary parameter updates
- (285+135) x 100 = 42,000 agent vs. adversary evaluations

Results

Meta strategy is much more diverse

Results – Meta Strategy





Adversary Disturbances:

- Policy 0:
  - Prob: 0.3
  - Dist: [0.00, 0.00, 0.00]
- Policy 3:
  - Prob: 0.0286
  - Dist: [0.43, 0.71, 0.99]
- Policy 4:
  - Prob: 0.4143
  - Dist: [0.37, 1.86, 1.89]
- Policy 6:
  - Prob: 0.2571
  - Dist: [0.11, 1.66, 0.73]

Results – Game Payoff

Game Payoff Matrix

$$\begin{bmatrix}
0.6 & 0.7 & 0.8 & 1.0 & 0.4 & 0.8 & 0.8 & 0.5 & 0.7 \\
0.4 & 0.3 & 0.5 & 0.5 & 0.5 & 0.5 & 0.3 & 0.6 & 0.8 \\
0.5 & 0.4 & 0.5 & 0.4 & 0.7 & 1.0 & 0.5 & 0.6 & 0.8 \\
0.7 & 0.8 & 0.6 & 0.3 & 0.5 & 0.7 & 0.6 & 0.5 & 0.5 \\
0.0 & 0.1 & 0.0 & 0.2 & 0.0 & 0.4 & 0.2 & 0.1 & 0.2 \\
0.4 & 0.6 & 0.6 & 0.5 & 0.4 & 0.4 & 0.7 & 0.5 & 0.4 \\
0.6 & 0.7 & 0.9 & 0.8 & 0.6 & 0.8 & 0.5 & 0.7 & 0.6 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.0 \\
0.1 & 0.4 & 0.3 & 0.4 & 0.1 & 0.4 & 0.3 & 0.4 & 0.2
\end{bmatrix}$$

Game Value: 0.58

Results – Success Rate

- **Task:** Obtain cereal box object from human hand without touching the hand and/or dropping the object

- **Adversary Strategy:** Random sampling from its 4 actions in meta strategy
- **Baseline Agent Policy Success Rate:** 29%
- **Meta-Strategy Agent Policy Success Rate:** 58%

- **Adversary Strategy:** No perturbation
- **Baseline Agent Policy Success Rate:** 42%
- **Meta-Strategy Agent Policy Success Rate:** 61%

Future Work

- Real world deployment of meta strategy and report on performance metrics
- Train on more objects & scenes other than the cereal box object
- Model the perturbations as temporally coupled (as is in Liang et al., 2024)
- Model the perturbations to be more human-like (e.g. pushing, yanking, shaking, drooping)
- Use imitation learning / behavior cloning to aggregate set of meta strategy policies into one policy
- Dynamic number of iterations and stopping when no novel best response exists for both players (aligns closer to theory)

Conclusion

- PSRO-based framework for robust human–robot handover under adversarial disturbances
- Initializing from scratch did not work very well
- Initializing from baseline checkpoints improves performance
- Gains over baseline:
    - No disturbance: 42% → 61% success rate
    - Adversarial: 29% → 58% success rate
- Learned disturbance policies reveal which perturbations are most harmful to handover stability

Thank You!

https://github.com/williamfbx/CMU-15888/tree/main/final_project