# Integrating Feedback and Noisy Preferences for Adaptable Robotic Control

## Abstract

Reinforcement learning (RL) has been successful in many complex tasks, but it relies on a well-designed reward function. This reward function may be handcrafted and require significant reward engineering, or can be sparse leading to longer training times. Demonstrations (from a human or an agent) allow an agent to learn when there are no reward signals available, but collecting demonstrations requires expert control knowledge and can be expensive to collect. In contrast, sub-optimal and noisy preferences or feedback are easier to collect but can be challenging to exploit effectively. This work considers the combination of feedback to learn a high-level policy and preferences for system identification, learning an effective control policy by integrating these two modalities advice. We also introduce a novel method to handle erroneous teacher preferences, improving performance as compared to other baselines without noise filtering. Experiments in two simulated robotic environments show how integrating different types of advice and handling noisy preferences improve performance, suggesting new ways of learning in the absence of rewards with suboptimal teachers.

## 1 Introduction

Recently, Deep Reinforcement Learning (DRL) has been used in building a large number of interesting autonomous agents such as Go Silver et al. (2016), Dota Berner et al. (2019), Robotics Akkaya et al. (2019) . Despite its continuing success, it still remains challenging to define a reward function in RL, which is usually manually engineered. This reward function needs to be correct (task-aligned) for the agent to learn an optimal policy, as well as informative so that the agent learns quickly. However, these hand-crafted reward functions often overfit to specific learning algorithms or are specified by experts in a myopic fashion Booth et al. (2023). Due to these variabilities in reward functions as well as sparse reward, RL agents are known to be sample-inefficient Ibarz et al. (2021), restricting their applications to toy domains like video games. To mitigate these issues, different kinds of human advice have been used to guide RL agents. Imitation Learning Schaal (1999), Inverse RL Ho & Ermon (2016), and RL from demonstrations Rajeswaran et al. (2017) are popular learning paradigms that show demonstrations from experts are capable of improving sample efficiency. However, the difficulty in acquiring optimal demonstrations from experts makes these methods restrictive. Other types of human advice have been used in the form of discrete feedback Knox & Stone (2009), preferences Wilson et al. (2012) and action-advice Torrey & Taylor (2013), to improve RL agents' training performance. However, complex domains like robotics are characterised by high-dimensional state and action spaces, thus requiring significantly large amount of human advice like feedback or preferences. A large amount of human advice may also lead to inherent noise owing to advice giver's exhaustion or lack of attention. Hence, while it is important to effectively leverage human knowledge to ease the burden of human experts, it is critical to handle noisy advice.

In order to effectively leverage human feedback for complex robotic manipulation tasks, we propose a two-level framework using human advice at both levels. In this framework, we decompose a task into a *high-level goal selector* and a *low-level controller*. This framework has similarities to hierarchical RL Li et al. (2022), but we do not break the actual task into sub-goals. The high-level layer is designed to have an abstract state-space representation without considering low-level features of the environment. This design makes it easier for the expert to give discrete feedback as advice at the higher level because this involves fewer details and complex environment dynamics. The high-level policy is learnt using a tabular RL algorithm to identify the relevant high-level goal. For example, for a desktop wiping robotic task where the robot needs to wipe a desk, the high-level state representation can be a grid structure of the desk space with actions corresponding to moving left, right, etc. This layer does not consider all the complex dynamics regarding contact forces, distance from the desk, kinematics, joint constraints, etc. The low-level controller contains the granular state and action space representation, and we use an admittance controller to serve as the policy at this level. To effectively drive policy learning of the controller, we incorporate human/simulated agent preferences at the low-level to fine-tune the parameters of the system-identification model, which in turn helps the force controller to learn an effective policy by adapting its parameters to the current environment. Moreover, since the low-level representation is complex, we propose a method to integrate noisy preference advice from non-expert humans.

**Contributions:** To summarize, this paper

1. introduces a simple yet effective layered control framework allowing human feedback and human preference to be leveraged at different level to drive the policy learning directly or indirectly at both the levels.
2. integrates noisy human preference with noises inside a learning-based system identification models for further fine-tuning of the low-level controller.
3. effectively leverages human advice by requiring less feedback at the high level because of abstract representation and *noisy* preference at low-level because of the task complexity.
4. succeeds on complex simulated robotics manipulation tasks, outperforming existing baselines.

## 2 Related Work

**Human knowledge for shaping agent behaviour:** Leveraging prior knowledge from experts has been demonstrated as an efficient method for expediting training in the decision-making literature, spanning from behaviour cloning Rajeswaran et al. (2017), teacher-student learning framework Torrey & Taylor (2013), and inverse reinforcement learning Ng et al. (2000). However, most of these methods require access to optimal expert knowledge , which is difficult or expensive to acquire. Human feedback has been proposed in literature to exploit human knowledge easily and effectively. Knox & Stone (2009) proposed TAMER to exploit simple binary feedback signals. In TAMER, feedback is collected by letting humans watch agents and is used to learn a human model. MacGlashan et al. (2017) proposed COACH, which assumed feedback to be dependent on the agent's current policy and used it as advantage function. These methods have also been extended in Deep RL settings like Deep TAMER Warnell et al. (2018) and Deep COACH Arumugam et al. (2019). Another comparatively easier way of giving feedback is by giving preference over an action or trajectory segment as compared to another segment Wilson et al. (2012). In this method, the reward model is learnt from human-specified preferences without access to the environment reward.

**Integrating different modalities of advice:** A small number of works have looked into integrating various types of human advice for policy learning. Demonstrations and preferences from teachers have been used for reward model learning Bıyık et al. (2022); Ibarz et al. (2018) in Atari games and complex robotics tasks. Similarly, demonstrations with corrections have also been used in context of interactive imitation learning Hoque et al. (2021); Kelly et al. (2019) where corrections are used by teachers to improve the robot's policy. Mehta & Losey (2022) proposed a method to integrate demonstrations, corrections, and preferences within a single framework to learn the reward model for robotics manipulation tasks. Le et al. (2018) propose a hierarchical architecture comprising an imitation learner at high-level and an RL agent at low-level. Our work is related to the above direction, where we use different kinds of advice (feedback and preference) at two different levels to effectively solve a high dimensional robotic manipulation task. We also effectively handle noisy preferences and this has not been considered in the previously mentioned work.

**Parameter estimation of system identification:** System identification is a procedure where we establish an understanding of the environmental dynamics model based on observed data. Thus, applying system identification could contribute to robotic control methods that require a dynamics model. One special case is to estimating unknown parameters of the system dynamics. Some data-driven methods for system identification are established for reinforcement learning. In the SimOpt framework Chebotar et al. (2019), to bridge the gap between the simulator and the real world, Relative Entropy Policy Search Peters et al. (2010) based parameter search is used. Yu et al. (2017) proposed online system identification by pre-training a system identification model and verified its ability against domain randomisation. All the proposed work in this direction view system identification as pure dynamics fitting or parameter searching, which is computationally costly and none of them has leveraged feedback in this layer. To the best of our knowledge, we are the first to incorporate preferences for parameter estimation of system identification problems.

**Hierarchical Control Framework** High-dimensional control tasks can be often challenging, like interactive tasks that requires to maintain contact forces in a proper range and that needs to adapt to uncertain external environmental parameters. A vision servo-based adaptive controller for motion and force tracking Cheah et al. (2009) and combining fuzzy logic with traditional sliding mode control Ravandi et al. (2018) are proposed to deal with this problem. Hierarchical control methods are widely used to effectively reduce the search space and improve adaptability in an unstructured environment. For example, The LAPPLAND Lin et al. (2022) is based on one-shot demonstration trajectory alignment to organise low level policies. A single force controller or single policy cannot solve complicated tasks and as previous work showing adding imitation learning and hierarchical structure improves performance Jiang et al. (2021), introducing different kinds of human advice may further expand its potential. There are also other hierarchical reinforcement learning frameworks such as FeUdal neural networksVezhnevets et al. (2017), Option Critic ArchitectureBacon et al. (2017) and HAC Levy et al. (2017) that allows a more stable temporal abstraction in different levels.

## 3 Background

**Reinforcement Learning:** Reinforcement Learning is a machine learning methodology that enables agents to find a proper policy under a certain environment. It is usually defined under a traditional Markov Decision Process (MDP), which is denoted by a quintuple as $M = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma\}$, where $\mathcal{S}$ denotes the agent's state space, $\mathcal{A}$ is the agent's action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the environmental dynamics transition probability, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the function that gives an immediate reward, $\gamma$ is a discount factor. In an MDP setting, the goal of RL is to optimise $\pi$ through maximise the expected accumulated reward $\mathbf{E}_{\tau\sim\pi}[\Sigma_{t=0}^{T} R_t]$.
**Co-teaching for de-noising:** In order to tackle noisy labels, Han et al. (2018) proposed Co-teaching for the supervised learning. Co-teaching adds another neural network and lets the two supervised models select data batches based on the losses and feed into each other reciprocally, as shown in the right part of Figure 1. Following the intuition that the model first tends to learn the easier patterns in data and overtime gradually overfits to the dataset, including noisy labels, the Co-teaching models selects the minibatch that gives the smallest loss and hence filter out noisy labels based on each other's model learning ability.

## 4 Algorithm and Methodology

Our proposed framework incorporating human advice at two levels is shown in Figure 1. The problem is decomposed into (1) a high-level task on an abstract state representation whose policy is $\pi^h$ and (2) a low-level task on the original state action representation whose policy is $\pi^l_\phi$. Human advice is incorporated into the high-level task in the form of discrete feedback, denoted by $f$, and into the system identification model as preference over its parameters $\phi$, denoted by $y$. We also introduce a pair of *noise filtering networks* that can handle noisy preference given by scripted teachers. Finally, using the goal proposed by the high-level goal selector and the refined parameters of the system-identification model, the low-level controller can learn its policy faster and with higher accuracy. The details of the main components of our framework are as follows, except for low-level controller design, which is available in Appendix A.7.

**High-level goal selector:** The high-level goal selector chooses a goal for the low-level task, which will contains information of the high-level state and action. The high-level task can be represented as an MDP $M^h = \langle \mathcal{S}^h, \mathcal{A}^h, \mathcal{T}^h, r^h, \gamma \rangle$, where $\mathcal{S}^h$ is the state space, $\mathcal{A}^h$ is the action space, $\mathcal{T}^h$ is the transition probability, $r^h$ is the immediate reward, and $\gamma$ is the discount factor. However, in our learning from feedback setting, $\gamma$ and $r^h$ is not used. The high-level policy is defined as $\pi^h_\theta : S^h \rightarrow \mathcal{A}^h$, which is parameterised by $\theta$ and maps high-level states to goals.

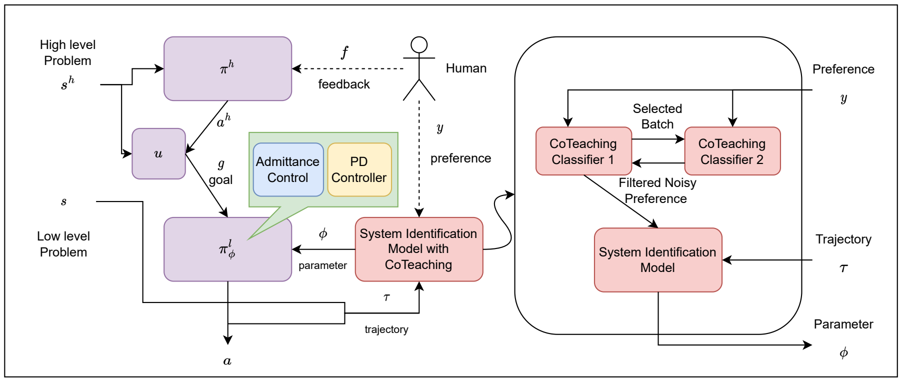


Figure 1: The framework integrates human feedback ($f$) for the abstract problem and human preferences ($y$) for system identification model.

**Incorporating feedback on the high-level task**: feedback is often defined as a scalar value describing the teacher's judgment, which can be a human or a scripted teacher, of the agent's current behavior. Like COACH Warnell et al. (2018) and TAMER Knox & Stone (2009), we define feedback as $f \in \{-1, 0, 1\}$, where $-1$ means the teacher discourages certain behavior, 0 means the teacher is indifferent, and 1 encourages the agent's behavior. Learning from feedback has been most effective in smaller domains like Mountain Car, Atari, Tetris Celemin & Ruiz-del Solar (2015); Knox & Stone (2009); Arumugam et al. (2019); Warnell et al. (2018) with limited cases of it's application to complicated high-dimensional tasks. Similar to COACH, we treat feedback as an estimate of advantage and use it to update the policy. We use feedback $f_t$ as proxy for advantage function as in Deep-COACH Arumugam et al. (2019) as Equation 1:

$$\nabla_{\theta_t} J(\theta_t) = \mathbf{E}_{a \sim \pi^h_{\theta_t}(\cdot|s^h)}[\nabla_{\theta_t} log(\pi^h_{\theta_t}(a^h|s^h)) \cdot f_t]. \quad (1)$$

**Preference Enhanced System Identification**: For the low-level task, we assume there are unknown parameters, $\phi$, that describe the environmental dynamics, influencing the performance of the low level policy $\pi^l_\phi$. To estimate $\phi$, we may pre-train a system identification model $SI : \tau \rightarrow \mathbb{P}(\phi \in \Phi)$, which maps from a trajectory slice $\tau : (s_{t-h}, a_{t-h}, s_{t-h+1}, a_{t-h+1} ... s_{t-1}, a_{t-1})$ of length $h$, to the distribution of $\phi$. The pre-training is done on a dataset $\mathcal{D} = \{(\tau_i, \phi_i)|i = 1, ..., N\}$, which is collected from a series of domain-randomised environments. This is a regression problem and the mean squared loss is used to fit the parameters. Details are in the Appendix A.4.

With a pre-trained model $SI$, we further incorporate human preferences into this model and collect preference $y_{\phi_0,\phi_1}(\phi) \in \{-1, 1\}$ over $\pi^l_{\phi_0}$ and $\pi^l_{\phi_1}$. 1 means the $\phi_0$ is preferred over $\phi_1$ and vice versa. Based on preferences, $SI$ is updated via Equation 2.

$$\begin{aligned}\mathcal{L}_{PESI} = -[&log(Pr(\phi_1))(Pr_(\phi_1) + p \cdot y_{\phi_1,\phi_2}(\phi_1))+ \\ &log(Pr(\phi_2))(Pr\phi(\phi_2) - p \cdot y_{\phi_1,\phi_2}(\phi_2)))],\end{aligned} \tag{2}$$

where $p$ is the encourage percentage controlling the update size, how confident we are about the preferences. This is a weighted cross entropy loss, where a preferred parameter candidate will be encouraged and the dis-preferred ones will be discouraged. The algorithm for the preference enhanced system identification (PESI) can be found in Appendix A.2.

**Algorithm 1** **Co**-teaching **P**reference-**E**nhanced **S**ystem **I**dentification

**Input**:System identification model $SI$ parameterised by $\xi$, learning rate $\alpha$, encourage percentage $p$, trajectory slice length $h$, candidate parameter set $\Phi$, preference buffer $D$, Co-teaching Classifier $M_0$, Co-teaching Classifier $M_1$, Co-teaching Start Data Size $s$, Co-teaching Rounds $t$, Forget Number $n_{forget}$, Model Refresh Frequency $f_{refresh}$

```
1: for $i \leftarrow 0, 1, 2, ...$ do
2:   Execute policy with $\phi_0$, $\phi_1$ sampled from $\Phi$
3:   Sample trajectory slice $\tau$
4:   Ask for human preference $y$ and append it to preference buffer $D$
5:   if $|D|$ is larger than $s$ then
6:     if Model Refresh Frequency $f_{refresh}$ met then
7:       Randomly initialise $M_0$ and $M_1$
8:     end if
9:     Sample Batch $b$ from $D$
10:    Select $b_0$ from Loss of Co-teaching Classifier $M_0$, forgetting $n_{forget}$ samples
11:    Select $b_1$ from Loss of Co-teaching Classifier $M_1$, forgetting $n_{forget}$ samples
12:    Update $M_0$ with $b_1$
13:    Update $M_1$ with $b_0$
14:    Update system identification model $SI$ by Eqn. 2 with selected batch data $b_0$ (or $b_1$)
15:  end if
16: end for
```

**Co-teaching for noisy preferences**: Preferences are normally easier to collect than demonstrations. Intuitively, determining which trajectory in a pair is better can be easier than performing the task itself, requiring less expertise. However, since humans provide these preferences, they are inherently noisy. Co-Teaching Han et al. (2018) is an algorithm for filtering noisy labels for deep neural networks. We propose Algorithm 1, namely CO-PESI, an enhanced version of Algorithm 2, by using co-teaching to handle noisy preference advice. In CO-PESI, two additional neural network classifiers are introduced so that co-teaching can help handle noisy preferences. In CO-PESI, a mini-batch $b$ of noisy preferences is sampled and then fed into co-teaching classifiers $M_0$ and $M_1$. The two classifiers compute the cross entropy loss over the batch $b$. Given the forget number $n_{forget}$, the corresponding amount of examples that gives the highest loss is forgotten, resulting in a filtered batch $b_0$ from $M_0$ and $b_1$ from $M_1$ (lines 10 and 11). The two classifiers will teach each other using the mini-batch selected by the other: $M_0$ is updated with mini-batch $b_1$ and $M_1$ is updated with mini-batch $b_0$ (lines 12 and 13). Next, we fine-tune the SI model via the data filtered by $b_0$ (or $b_1$ due to the symmetry of the learning process). In order to avoid overfitting, we refresh the preference classifiers after a certain frequency $f_{refresh}$ as shown in Algorithm 1, line 7, by letting it learn from scratch.

## 5 Experiments

We aim to answer the following research questions:

**R1:** Can incorporating advice at both the high- and low-level help train a robotics task?

**R2:** How will noises influence the performance of the proposed framework?
**R3:** Can integrating co-teaching help handle noisy preferences and yield better performance?

**Setting**: We experiment in the MuJoCo simulator Todorov et al. (2012) using an UR5e robot arm. The two tasks are described below.

**1. Desktop wiping:** In this domain (see Figure 2(a)), the robot arm has a sponge attached to the end effector. The goal is to control the robot to clean the desktop within a desired range of contact force and distance to the desktop. While we want to keep contact between the sponge and the desktop, we also want the robot to apply a force that is not too strong or too weak, within a safe range. To safely perform this task, the desktop's height must be accurately estimated so that the sponge touches the desktop but the robot arm does not make contact.

**2. Pushing:** This task (see Figure 2(b)) requires the robot to push an object to a desired position. The initial position and the target position of the object are set randomly such that they are on the table but reachable by the robot. We want to keep the object as close to the target location as possible. In this domain, the parameter we want to estimate is the object's mass so that the force controller successfully moves the object.

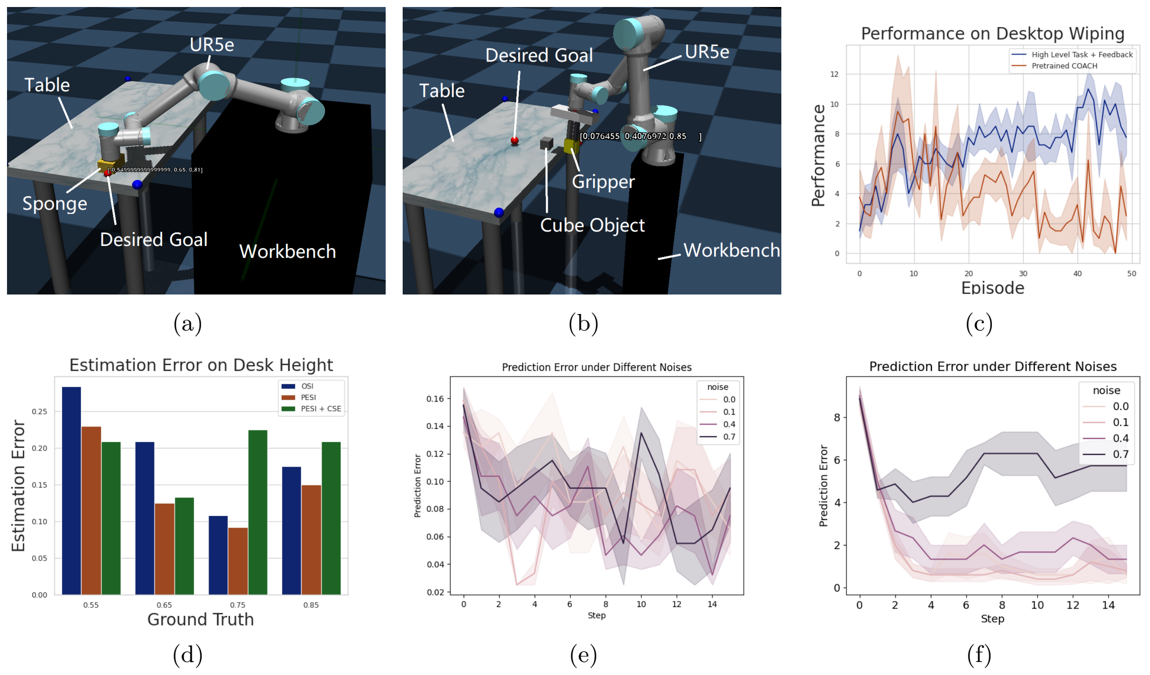
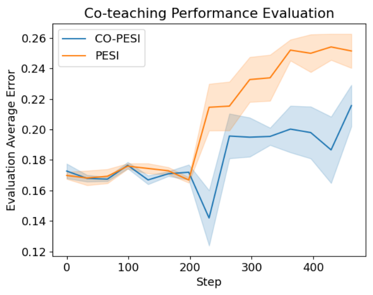


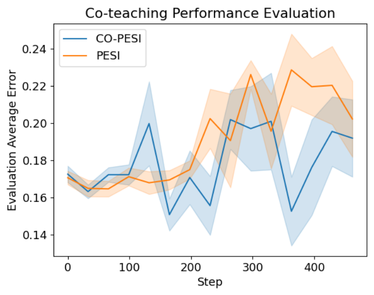
Figure 2: (a) Desk wiping environment: the robot arm is trying to move the cleaning sponge to the desired next goal (a red dot). (b) Pushing environment: the robot arm with a gripper is trying to push the cube object to the desired goal (a red dot).(c) High level policy performance (count of wiped grids) over episodes compared with Pretrained COACH. (d) Parameter estimation error of Online System Identification, our method PESI, and PESI but with standard cross entropy loss with different ground truth desk heights. (e)(f) Prediction error on domain desktop wiping and object pushing, with different levels of noisy preference. It is observed the better preference provided, the better prediction precision achieved.

**Teacher details and baselines:** In our experiments, we have two kinds of human advice: feedback and preference. The simulated agent provides feedback based on a hand-coded expert policy. If the action is the same with the expert's action, a positive feedback is provided and vice versa. For preferences, the simulated agent will determine which of the candidates are closer to the ground truth parameter of the environment and determine the favoured candidate. Furthermore, in the case of noise, a certain percentage of the preference labels will be flipped.

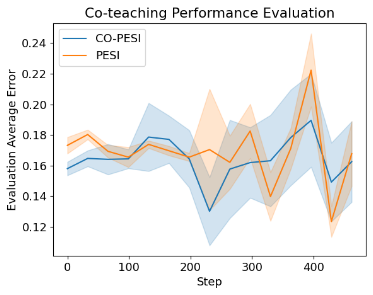
**Performance metrics:** We consider two performance metrics for estimating the effect of the system-identification model and the overall task execution: 1) average parameter estimation error and 2) success rate of the task. Both metrics are evaluated after every 33 episodes of learning, freezing the learning and repeating for 10 episodes. Details are in the Appendix A.1,A.7.



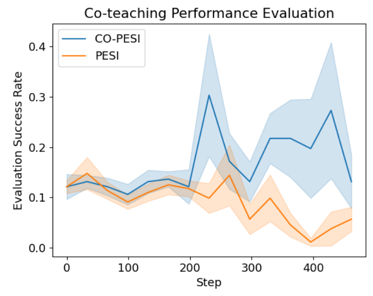
(a)



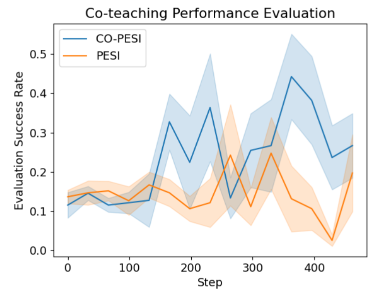
(b)



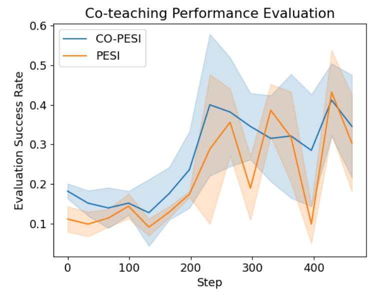
(c)



(d)



(e)



(f)

Figure 3: (a)(b)(c) Prediction Error of height (desktop wiping) over time with faked human preference under different noise scales (10%, 20%, 30% from left to right) (d)(e)(f) Success rate in desktop wiping over time against different noise scales. As Co-teaching generally shows performance improvement under 10% and 20% noise, it is noticeable that under 30% noise the Co-teaching reaches its limit on our tasks.

## Experimental results and analysis

**Effectiveness of incorporating advice on two levels:** We first evaluate the effectiveness of the high-level task representation on the desktop wiping domain to answer **R1**. Here, the abstract problem is defined as a gridworld, discretizing the desktop's state space. MDP states are grid coordinates and actions are four directions: up, down, left, and right.

The high-level policy must control the agent moving and traversing the grids to successfully wipe the desktop as much as possible. The reward function for this MDP can be defined as the total area the agent traverses in an episode. Accordingly, the actual problem would be how to control the robot arm to a desired position on the desktop, within a safe range of force applied. The performance of different learning episodes, guided by a simulated teacher's feedback, is shown in Figure 2 (c). We use a tabular COACH MacGlashan et al. (2017) algorithm to learn the high-level policy. In this experiment, since we are focusing on the high-level problem, *performance metric* refers to how many grids cells the policy wiped in an episode, and helps evaluate the teacher's feedback. Furthermore, in order to make it a fair comparison, we use pretrained COACH as the baseline so that we can apply feedback on a neural network. In Figure 2(c), Pretrained COACH refers to directly applying feedback to a policy that is pretrained with a thousand demonstration trajectories from the desktop wiping domain, that is collected from a predefined force controller. The reason for using a pretrained COACH policy is to ensure a fair comparison by using the same pretraining data, while ours comes with a similarly predefined goal-conditioned low level controller. As can be seen from the Figure 2(c), our method incorporating human feedback is better than Pretrained COACH (also incorporating the same feedback). Pretrained COACH even shows worse performance over time, suggesting that binary feedback directly applied to a flat policy can be hard to give, which might even mislead the agent. The reason behind this is that the actual action, for example, torques for robot arms, are high-dimensional and cannot be easily evaluated with binary signals. This partially answers **R1** affirmatively that for complex tasks: the high-level problem can effectively incorporate teacher's feedback.

Next, we evaluate the effectiveness of incorporating preferential advice in the SI model, which impacts the learning of the low-level controller. We compare our algorithm to Online System Identification Yu et al. (2017), estimating desktop height in the wiping task and object mass in the pushing task. Noticeably, PEBBLE Lee et al. (2021) does not serve as a proper baseline here as it requires access to preference over trajectories and in our setting of system identification, the preference is acquired over different parameter candidates. As shown in Figure 1, the teacher watches two video clips, each illustrating $\pi^l$ parameterized by different candidate parameters and then gives a relative ranking (preference). We see from Figure 2(d) that the system identification gives a more accurate prediction of the parameter after taking preferences, denoted as PESI, compared to OSI (Online System Identification, without preferences). This answers R1 affirmatively.[1]

**Evaluation on Noisy Preferences** To answer **R2**, Figure 2(e) &(f), shows the effect of various noise level on the performance of PESI, for desktop wiping and object pushing. The parameter estimation precision is notably impacted by noise. With perfect preferences, the prediction error can be reduced and with noisy preferences, the performance is significantly deteriorated. When the noise exceeds 50%, we observe that the model unlearns over time. This answers **R2**: noise in preferences can significantly hurt the precision of the system identification module.

**Evaluation CO-PESI to handle noisy preference** As concluded from the previous section, the noisy preferential advice significantly hinders the fine-tune process of SI model. The experimental results of the CO-PESI on desktop wiping domains is shown in Figure 3 and the other domain results are in Appendix A.1. The experiments are conducted under 10%, 20% and 30% noises (flipping the preference) and show the CO-PESI successfully improves the success rate and reduces the prediction error in the presence of noise. Results for 10% and 20% noise are more promising than 30% noise. In our problem setting, the noise is randomly applied to preferences and the algorithm does not have access to ground truth labels. Therefore, noise filtering is built upon the co-teaching model's initialisation and generalisation ability. The experiments above suggest that CO-PESI is effective in filtering noisy preference up to 20% noise, thus answering **R3** affirmatively.[2]

## 6 Conclusion and future work

In this paper, we learn from simple teacher's feedback for complicated high-dimensional tasks, and integrate feedback at higher task representation and preferences into system identification for better control. Furthermore, our proposed framework handles noisy preference advice in the system identification module, even with unknown proportion of noises. There are several avenues for future work. Firstly, improving the framework by integrating symbolic planning in the higher-level to give more environmental adaptability and generalization is an exciting future work. Secondly, applying similar de-noising techniques to high level control might also further strengthen the framework's robustness under noisy advice. Additionally, introducing preferences from multiple teachers can be a way to improve sample efficiency. Lastly, experiments in real-world robotics domain and larger-scale user study remains interesting future extensions of this work.

[1]Additional experimental evaluations related to PESI with simulated agents, and PESI+CSE can be found in Appendix A.5.

[2]The evaluation results of the second domain, and the results with unknown proportion of noise can be found in Appendix A.6,A.9

## References

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L Littman. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257*, 2019.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

Serena Booth, W Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5920–5929, 2023.

Carlos Celemin and Javier Ruiz-del Solar. Coach: Learning continuous actions from corrective advice communicated by humans. In *2015 International Conference on Advanced Robotics (ICAR)*, pp. 581–586. IEEE, 2015.

Chien Chern Cheah, Saing Paul Hou, Yu Zhao, and Jean-Jacques E Slotine. Adaptive vision and force tracking control for robots with constraint uncertainty. *IEEE/ASME Transactions on Mechatronics*, 15(3):389–399, 2009.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979. IEEE, 2019.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *Conference on Robot Learning (CoRL)*, 2021.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 2021.

Hao Jiang, Zhanchi Wang, Yusong Jin, Xiaotong Chen, Peijin Li, Yinghao Gan, Sen Lin, and Xiaoping Chen. Hierarchical control of soft manipulators towards unstructured interactions. *The International Journal of Robotics Research*, 40(1):411–434, 2021.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083. IEEE, 2019.

W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16, 2009.

Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pp. 2917–2926. PMLR, 2018.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

Seunghwan Lee, Phil Sik Chang, and Jehee Lee. Deep compliant control. *Trans. of ASME Journal of Dynamic System, Measurement, and Control*, 2022.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022.

Nan Lin, Yuxuan Li, Keke Tang, Yujun Zhu, Xiayu Zhang, Ruolin Wang, Jianmin Ji, Xiaoping Chen, and Xinming Zhang. Manipulation planning from demonstration via goal-conditioned prior action primitive decomposition and alignment. *IEEE Robotics and Automation Letters*, 7(2):1387–1394, 2022.

James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning*, pp. 2285–2294. PMLR, 2017.

Shaunak A Mehta and Dylan P Losey. Unified learning from demonstrations, corrections, and preferences during physical human-robot interaction. *arXiv preprint arXiv:2207.03395*, 2022.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Abbas Karamali Ravandi, Esmaeel Khanmirza, and Kamran Daneshjou. Hybrid force/position control of robotic arms manipulating in uncertain environments based on adaptive fuzzy sliding mode control. *Applied Soft Computing*, 70:864–874, 2018.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 1999.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Jie Tan, Karen Liu, and Greg Turk. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications*, 31(4):34–44, 2011.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Lisa Torrey and Matthew Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1053–1060, 2013.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pp. 3540–3549. PMLR, 2017.

Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *NIPS*, 2012.

Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

## A Appendix

### A.1 Domain Details

MuJoCo Todorov et al. (2012) is a general purpose physics engine that is widely used for robotics simulation. In our domain design, a UR5e robot arm is used to execute the task and gravity compensation is tuned on to better simulate real world. On a desktop of size 120cm × 60cm × 5cm, the robot arm is expected to execute two tasks as mentioned in the paper. Furthermore, the domains can be altered via designating certain parameter to achieve domain randomisation. In the desktop wiping domain, the height of the desktop can be changed, and in the object push domain, the object mass can be altered. A summary of related parameters is shown in Table 1.

Table 1: The parameters of the MuJoCo simulator

| Parameters | **Value** |
|---|---|
| Time step | 2e-3 |
| Gravity | -9.81 |
| Camera Position | [0, 1.7, 1] |
| Camera Euler Angle | [4.71238898, 0, 3.14159265] |
| Default Desktop Size | [1.2, 0.6, 0.05] |
| Default Sponge Size | [0.1, 0.05, 0.07] |
| Default Object Size | [0.05, 0.05, 0.05] |
| Desktop Sliding Friction | 1.0 |
| Desktop Torsional Friction | 0.5 |
| Desktop Rolling Friction | 0.5 |
| Motor Gear | 1.0 |
| Motor Control Range | [-1000, 1000] |
| Arm Joint Range | [-3.14, 3.14] |
| Constraint Solver Time Const | 0.03 |
| Constraint Solver Damp Ratio | 1 |
| Constraint Solver Impedance Midpoint | 0.1 |
| Constraint Solver Impedance Power | 0.01 |

### A.2 Preference from N Candidates

We propose PESI with a new loss function that can integrate preference into fine-tuning system identification model, as illustrated in Algorithm 2

Furthermore, the loss function in Equation 2 can be further generalised for humans or scripted teachers giving preference over $N$ candidates, i.e., $y_{\phi_0,\phi_1,\ldots\phi_N}(\phi) \in \{-\frac{1}{N-1}, 1\}$ as per Equation 3.

$$\mathcal{L}_{PESI} = -\Sigma_{i=1}^{N} log(Pr(\phi_i))(\Sigma_{i=1}^{N-1} Pr(\phi_i) + p \cdot y_{\phi_0,\ldots,\phi_{N-1}}(\phi_i)) \tag{3}$$

### A.3 Using a neural network as low level controller

The proposed framework can be expanded to a version that uses a goal-conditioned deep policy $\pi^l(\bar{q}|s, \phi)$ at low-level, which can be acquired through imitation learning or reinforcement learning.

**Algorithm 2** **P**reference **E**nhanced **S**ystem **I**dentification

```
Input:System identification model $SI$ parameterised by $\xi$, learning rate $\alpha$, encourage percentage $p$, trajectory slice length $h$, candidate parameter set $\Phi$
  for $i \leftarrow 0, 1, 2, ...$ do
    Execute policy $\pi^l_\phi$ with $\phi_0$, $\phi_1$ sampled from $\Phi$
    Sample trajectory slice $\tau$
    Ask for human preference $y$
    Update system identification model $SI$ by Eqn. 2
  end for
```

The Algorithm 3 shows an example of using imitation learning to acquire goal-conditioned deep policy.

**Algorithm 3** Learning $\pi^l$ with imitation learning

```
Input:Position controlled system $P$, Goal space $\mathbb{G}$, Environment candidate parameter set $\lneqq$, differentiable $\pi^l$, replay buffer $R$, batch size $N$
 1: for $i \leftarrow 0, 1, 2, ...$ do
 2:   Randomly sample goal $g$ and environmental parameter $\phi$
 3:
 4:   Set simulator with environmental parameter $\phi$
 5:
 6:   Collect trajectory $\tau$ using $P$ under goal $g$
 7:
 8:   Pushing $\tau$ into $R$
 9:
10:   Update $\pi^l$ with MSE loss, with an $N$-sized batch sampled from $R$
11: end for
```

### A.4 Pre-train the System Identification Model

The algorithm to pre-train the System Identification Model is shown in Algorithm 4, which follows a supervised learning style Online System IdentificationYu et al. (2017). The trained model will map a certain piece of trajectory to a predicted distribution of the parameters, which is further fine-tuned in our paper with preferences.

### A.5 PESI Evaluation with Simulated Agent

Due to the limitation of human reaction speed, collecting real feedback is rather slow and costly, and we also performed experiments using a scripted teacher with different levels of noise, which represents the percentage of randomly flipped preferences. The experimental results are shown in Figure 4(a)(b) to show the effectiveness of using preference to fine-tune the system identification

**Algorithm 4** Pre-train System Identification Model

```
Input:System identification model $SI$ parameterised by $\psi$, trajectory dataset $\mathcal{D}$, learning rate $\alpha$, trajectory slice length $h$, batch size $l$
1: for $i \leftarrow 0, 1, 2, ...$ do
2:   Sample batch $\{(\tau_j, \phi_j)|j = 1, ..., l\}$ from $\mathcal{D}$
3:
4:   Update system identification model $SI_\psi$ using MSE Loss
5:
6: end for
```

model. For both domains, it is observed that the preference input helps reduce the prediction error over time.

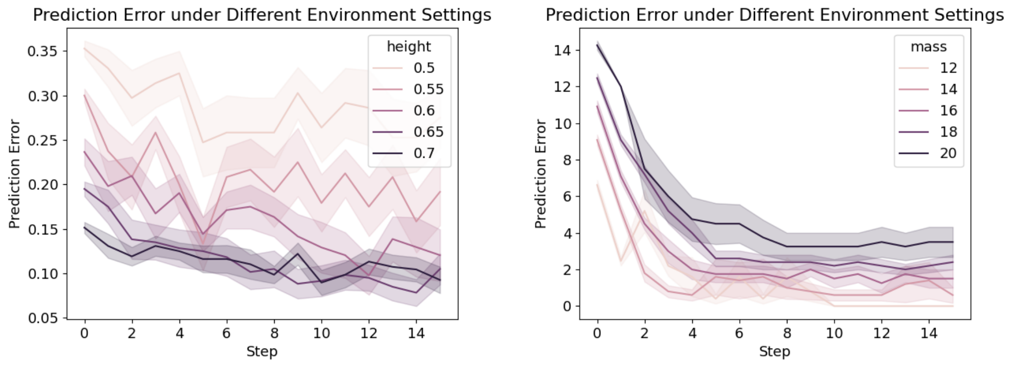


Figure 4: (a)(b) Prediction Error over time with perfect faked human preference on desk height/object mass and we can see the preference helps to reduce the prediction error over time.

## A.6 Additional CO-PESI Evaluation Results

The CO-PESI evaluation results on our second domain, object pushing, are shown in Figure 5. In our second domain, results suggest slightly difference results, as this task appears to be easier to solve. It is observed that 10% noise is not strong enough to hinder the learning process, so both of the algorithms perform well and under 20% noise, CO-PESI shows a better success rate than PESI. Similarly, under 30%, the limit of CO-PESI's de-noising ability is reached, hence no significant performance difference can be found.

## A.7 Implementation and Hyperparameters

**Details of High Level** Noticeably, there exists a certain linear coordinate transformation (namely $u$ shown in Fig. 1) between the grid world to the actual world coordinate system on the desktop. Furthermore, the grid world may not strictly align with the actual world, especially when the $\pi^l$ fails to reach the goal. In that case, a reset of the grid world state is required. The grid world is aligned to the actual desktop with grid size of 0.05m by 0.05m, which is slightly smaller than the sponge size to ensure the granularity is of proper range. In the high level policy, the grid world will receive expert feedback every step for 50 episodes and each episode's max length is 290. The feedback provider at high level does not involve noise in our setting.

**Details of Low Level** Our design of admittance force controller is shown in Fig 6 as described in our paper. The low-level controller chooses a low-level action in the original high-dimensional state and action space. It can be represented as a goal-conditioned MDP $M^l = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \mathcal{G} \rangle$ where $\mathcal{S}$ denotes state space, $\mathcal{A}$ denotes the action space, $\mathcal{T}$ is the transition probability, $r$ is the immediate reward, $\gamma$ is the discount factor, and $\mathcal{G}$ is the goal space. The low-level policy, $\pi^l_\phi : \mathcal{S} \to \mathcal{A}$, maps the low-level states to actions. $\phi$, from the parameter candidate set $\Phi$, is the parameter predicted by the system-identification module $SI$, which represents important parameters that describe environmental dynamics. We further define a mapping function $U : \mathcal{S}^h \times \mathcal{A}^h \to \mathcal{G}$ that maps the high-level states and actions to low-level goals. The low-level controller can be a predefined force controller or a learned neural network.

For the low level controller, we implemented a force-based controller and later extended it to a learning-based version Lee et al. (2022). While the robotic arm is an open-chain articulation with the fixed root, its configuration can be expressed by its joint positions $q \in \mathbb{R}^n$, joint velocities $\dot{q} \in \mathbb{R}^n$, and joint accelerations $\ddot{q} \in \mathbb{R}^n$ in generalized coordinates. The equations of motion describing the

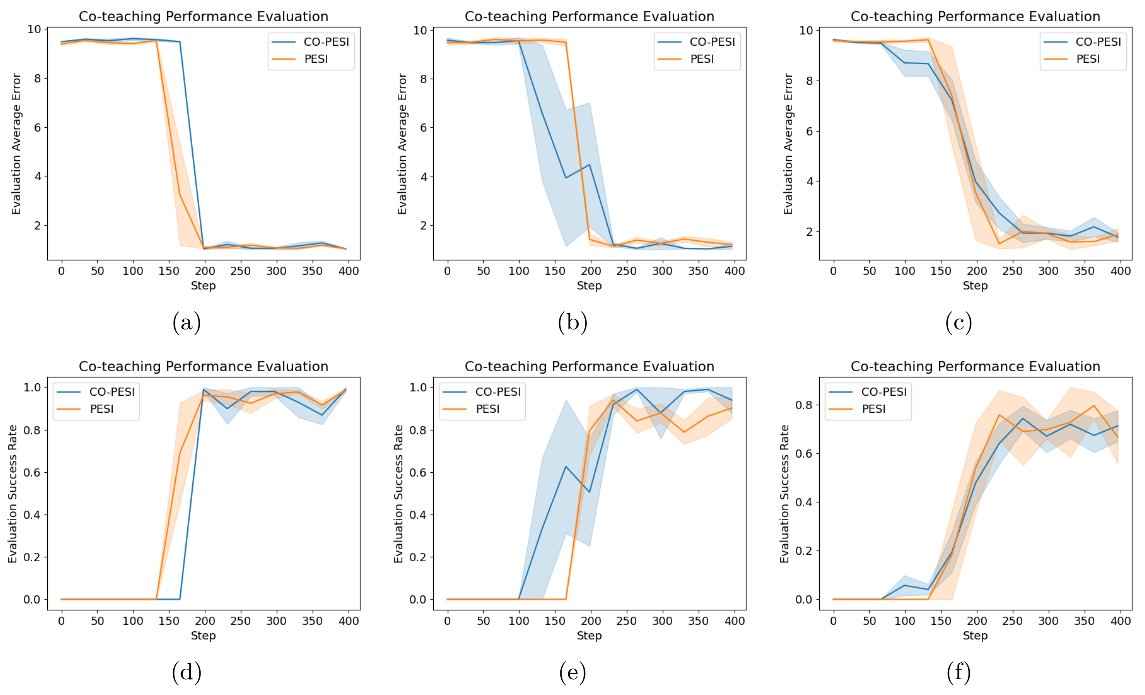


Figure 5: (a)(c)(e) Prediction Error of mass over time with faked human preference under different noise scales (10%, 20%, 30% from left to right) (b)(d)(f) Success rate over time against different noise scales. This task is easier to solve and Co-teaching generally shows slightly better performance 20% noise, it is noticeable that under 30% noise Co-teaching reaches its limit on our tasks, and under 10% noise both methods learn well with no noticeable difference.

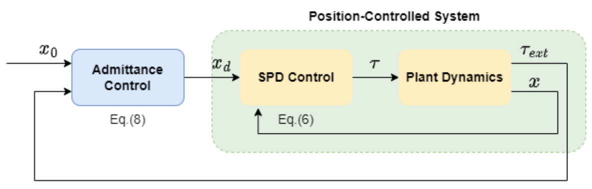


Figure 6: Our admittance control based implementation.

dynamic system is as follows:

$$M(q)\ddot{q} + c(q,\dot{q}) = \tau + \tau_{ext} \tag{4}$$

where $M(q)$ is the mass matrix, $c(q,\dot{q})$ is Coriolis and gravitational force, and $\tau_{ext}$ is the sum of external forces that includes contact force and other external perturbations.

The PD controller calculates the control force $\tau$ via a stable proportional derivative (SPD) formulation Tan et al. (2011):

$$\tau = -k_p(q^n + \dot{q}\Delta t - \bar{q}^{n+1}) - k_d(\dot{q}^n + \ddot{q}^n \Delta t) \tag{5}$$

where both $k_p$ and $k_d$ are diagonal matrices that indicate the gains and damping coefficients, and $\Delta t$ is the time step. SPD computes the control forces using the next time step state $q^{n+1}$, which can be expanded as $q^n + \Delta t \dot{q}^n$ via Taylor series, and the same to $\dot{q}^n$. The acceleration can be written as:

$$\ddot{q}^n = (M + k_d \Delta t)^{-1}(-c - k_p(q^n + \dot{q}^n \Delta t - \bar{q}^{n+1}) - k_d \dot{q}^n + \tau_{ext}) \tag{6}$$

The explicit Euler method integrates to the next time step.

Table 2: The hyper parameters of PESI and CO-PESI

| Parameters | **Value** |
|---|---|
| System Identification Learning Rate | 1e-3 |
| System Identification Model Hidden Size | [256, 128, 64, 32] |
| System Identification Model Activation | TanH |
| System Identification Model History Length | 3 |
| Co-teaching Learning Rate | 1e-2 |
| Co-teaching Classifier Model Hidden Size | [64, 64] |
| Co-teaching Classifier Model Activation | ReLU |
| Height Randomisation Range | [0.5, 0.9] |
| Height Estimation Batch Size | 16 |
| Height Estimation Refresh Frequency | 16 |
| Mass Randomisation Range | [8, 20] |
| Mass Estimation Batch Size | 32 |
| Mass Estimation Refresh Frequency | 32 |
| Evaluation Frequency | 33 |
| Evaluation Repetition | 15 |

The mathematical expression of a single degree-of-freedom system in which a mass interacts with an environment is defined as $m\ddot{x} = \tau + \tau_{ext}$, where $m$ and $x$ are the inertia and displacement of the mass, respectively. The Admittance Control is designed to control the force $\tau$ that will establish a given relationship between $\tau_{ext}$, a desired trajectory $x_0$, and a desired position $x_d$. Typically, a linear second-order relationship

$$M_d(\ddot{x}_d - \ddot{x}_0) + D_d(\dot{x}_d - \dot{x}_0) + K_d(x_d - x_0) = \tau_{ext} \tag{7}$$

is considered, where the positive constants $M_d$, $D_d$, and $K_d$ represent the desired inertia, damping, and stiffness, respectively. In Admittance Control, the plant is position-controlled and can be implemented using the SPD controller as per Eqn. 5 mentioned above.

Besides the force controller, the system identification models are trained with trained with 100 trajectories, each of length 100 states and actions, that are collected under domain randomisation of a certain parameter that we want to estimate. The randomisation is done with a uniform distribution and the range is $[0.5, 0.9]$ metres for height and $[8, 20]$ kilograms for the object mass. Both models consist of four fully connected hidden layers with size of 256, 128, 64, 32, with hyperbolic tangent as its activation function. The input history length is set to 3 and the encourage percentage is 0.25. The finetuning process is conducted for learning rate of 0.01, batch size of 16 for (height estimation) and 32 (for mass estimation) and refresh frequency of 16 and 32. Furthermore, the remember rate is set up to corresponding to same level of the noise level. The evaluation are done every 33 episodes and and will be repeated for 15 times. A summary of can be found in Table 2.

## A.8 Ablation Study

In this section, we do ablation study to verify the effectiveness of our algorithm design. Recall Equation 2, instead of using cross entropy loss as Christian et al. Christiano et al. (2017), we exploit human preferences by a soft cross entropy loss. This is due to the fact that preferences are given to sampled parameter candidates while there are still other possible parameter candidates. The soft cross entropy loss is adopted to illustrate the idea that a preferred parameter candidate may not be a "positive class" or the right prediction, but only a signal to shape its distribution. The results of the ablation study using standard cross entropy loss are illustrated in Fig. **??**(b) denoted by standard CSE We conclude that soft preferences show low estimation error as compared to standard CSE. However, OSI is still the worst, given the fact that it does not involve any preferences.

We also conducted experiments to illustrate the benefits of using admittance control. We use two different low-level controllers including the SPD Controller and the Admittance to manipulate the robotic arm directly. We let the robotic arm start from the same initial position and execute the same random trajectory on the table. At the same time, the data of the torque sensor installed at the end are collected. In the desktop wiping, the height of the table surface is 0.81 m. We try to set the desired height of environment to 0.72m, 0.74m, 0.76m, 0.78m and 0.8m. The mean and standard deviation of the z-axis force during robotic arm execution has been shown in Fig.7. We can see that the force exerted by the end-effector on the table can be very large, which may cause damage to the robotic arm or the table in the real world, hence warranting the admittance control.

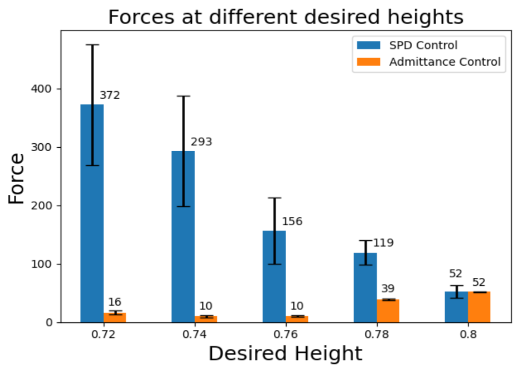


Figure 7: Admittance control gives smaller contact forces compared to SPD controller.

## A.9 Noise Percentage Estimation

The Co-teaching algorithm requires a previously calibrated forget number $n_{forget}$, which represents how many samples we want to filter out. Ideally, this hyperparameter is set according to the actual percentage of noise in the sampled data. In this section, we briefly introduce a simple algorithm (Algorithm 5) to estimate the noise percentage in the preference data, without requiring any prior knowledge. The general idea of this algorithm is to search conflicts of preferences. For example, among preferences to estimate the desktop height, a conflict occurs when there's two preferences A and B, where A suggests 0.5m is a better parameter candidate than 0.6m while B suggests vice versa. The algorithm will search such conflicts among data entries and will deem those data entries with more conflicts than average, to be potential noises. Noticeably, a noisy preference may not necessarily have conflicts with other correct ones and this method merely provides an approximation of the noise. A simulated experiment is repeated for 100 times to see its average performance in estimating the noise percentage. It is observed in a randomly generated preference dataset of size 256, the algorithm often tends to overestimate the noise by about 6%, as shown in Table 3. The overestimation of noise, in comparison to underestimation, is less of a problem when we use the preferences to finetune the neural network model as the later is prone to ignore more noisy labels. Furthermore, this algorithm tends to perform better when the ground truth noise is larger.

Experiments are conducted to verify if we apply this noise percentage algorithm to the CO-PESI, with no knowledge of the groundtruth noise level. The results are illustrated in Fig 8 and Fig 9. It is shown that under noise of 20% and 30%, a similar performance is achieved even when we do

**Algorithm 5** Estimate Preference Noise Percentage

```
Input:Preference dataset $\mathcal{D}$
Output:Noise Percentage $p_{noise}$
1: Initialise conflicts counting array $C[0...|\mathcal{D}|-1]$
2: for $i \leftarrow 0, 1, 2, ..., |\mathcal{D}|-1$ do
3:
4:     for $j \leftarrow i+1, 1, 2, ..., |\mathcal{D}|-1$ do
5:         if $\mathcal{D}_i$ is conflicting with $\mathcal{D}_j$ then
6:
7:             $C[i] \leftarrow C[i]+1$
8:
9:             $C[j] \leftarrow C[j]+1$
10:        end if
11:    end for
12:
13: end for
14: Find the average $\mu$ of conflicts counting array $C$
15:
16: Let $N = 0$
17: for $i \leftarrow 0, 1, 2, ..., |\mathcal{D}|-1$ do
18:     if $C[i] \geq \mu$ then
19:         $N \leftarrow N+1$
20:     end if
21: end for $\frac{N}{|\mathcal{D}|}$
```

| Groudtruth Noise | Estimated Noise |
|---|---|
| 0.1 | 0.19 |
| 0.15 | 0.23 |
| 0.2 | 0.27 |
| 0.25 | 0.32 |
| 0.3 | 0.35 |
| 0.35 | 0.38 |
| 0.4 | 0.44 |

Table 3: The average estimated noise against different scale of noises

not know the groundtruth scale. However, under 10% noise, it performance is hindered by the fact the our algorithm overestimate the noise level too much by about 9%. Furthermore, since the noise level estimation algorithm tends to overestimate, it slows down the learning speed.

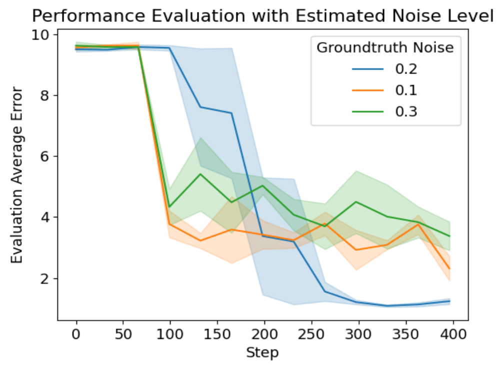


Figure 8: Average error on mass estimation with estimated noise level

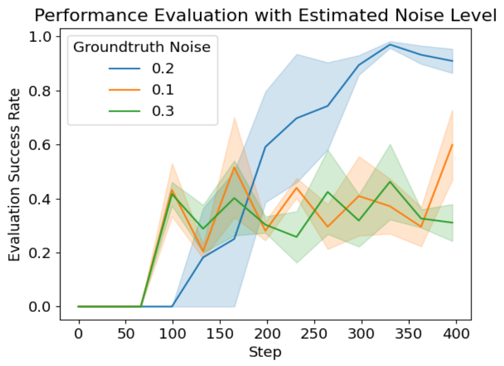


Figure 9: Success rate on mass estimation with estimated noise level