Counterfactual Behavior Cloning: Offline Imitation Learning from Imperfect Human Demonstrations

SHAHABEDIN SAGHEB and DYLAN P. LOSEY, Virginia Tech, USA

Learning from humans is challenging because people are imperfect teachers. When everyday humans show the robot a new task they want it to perform, humans inevitably make errors (e.g., inputting noisy actions) and provide suboptimal examples (e.g., overshooting the goal). Existing methods often learn by matching some or all of the human's behavior — but this approach is fundamentally limited because the demonstrations themselves are imperfect. In this work we advance offline imitation learning by enabling robots to extrapolate across nearby actions, instead of only considering what the human actually showed. We achieve this by hypothesizing that all of the human's demonstrations are trying to convey an underlying policy, while the noise and suboptimality within their behaviors obfuscates the data and introduces unintentional complexity. To recover the underlying policy and learn what the human teacher meant, we introduce *Counter-BC*, a generalized version of behavior cloning. Counter-BC expands the given dataset to include actions close to behaviors the human demonstrated (i.e., counterfactual actions that the human teacher could have intended, but did not actually show). During training Counter-BC autonomously modifies the human's demonstrations within this expanded region to reach a simplified policy that explains the underlying trends in the human's dataset. Theoretically, we prove that Counter-BC can extract a simple and similar-to-demonstration policy from imperfect data, multiple users, and teachers of varying skill levels. Empirically, we compare Counter-BC to state-of-the-art alternatives in simulated and real-world settings with noisy demonstrations, standardized datasets, and real human teachers. Overall, we find that trying to extrapolate what the human teacher meant by considering nearby actions — as opposed to rigorously following what actions the human actually demonstrated — can lead to more proficient learning from humans. See videos of our work here: https://youtu.be/XaeOZWhTt68

CCS Concepts: • **Computing methodologies** → **Learning from demonstrations**.

Additional Key Words and Phrases: Imitation Learning, Behavior Cloning, Human-Robot Interaction

1 INTRODUCTION

Robots can learn new tasks by imitating human examples. Consider the system in Figure 1: by watching how a human plays air hockey, this robot arm should learn how to block and hit the puck. But one fundamental limitation when learning from humans is that people are imperfect teachers [11, 28, 36]. Human demonstrations of the task will inevitably contain mistakes, noise, and suboptimal actions, particularly in real-world settings with inexperienced human operators. For instance, when teaching the robot how to play hockey the human might accidentally miss the puck, press the wrong buttons on the joystick, or take convoluted paths towards the target. These sorts of mistakes can arise for multiple reasons: the task may be physically demanding or the user may not be paying sufficient attention, the teleoperation interface may introduce input delays or mapping difficulties, or the data may be collected from heterogeneous users whose skill levels and intentions differ. Imperfect demonstrations are a problem because — when robots mimic suboptimal examples — they learn to perform the task incorrectly (e.g., only hitting the puck occasionally). Moreover, poor demonstrations often cannot simply be discarded and re-recorded. At scale, this would require a human expert to review thousands of demonstrations from many users, and in many deployment scenarios (e.g., data collected passively or from remote users) re-recording may not be feasible.

To advance learning from humans we need robots that can extrapolate what the user actually *meant*, not just what the human *showed*. This is inherently challenging because the human is the

This work was supported in part by NSF Grant #2337884.
Authors' address: Shahabedin Sagheb, shahab@vt.edu; Dylan P. Losey, losey@vt.edu, Virginia Tech, Department of Mechanical Engineering, 635 Prices Fork Rd, Blacksburg, VA, 24060, USA.

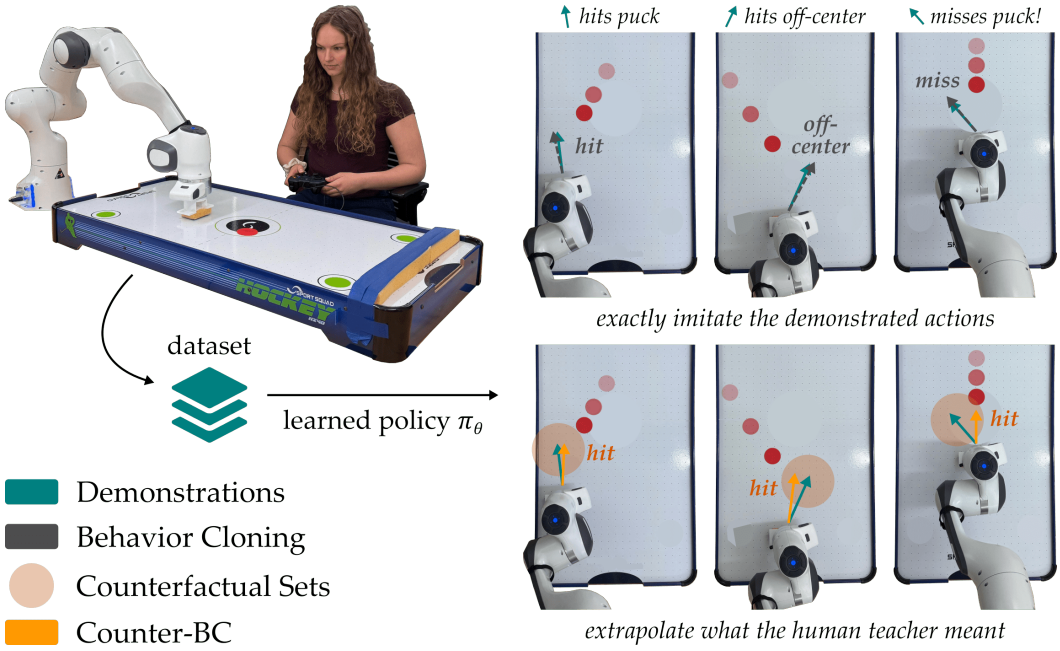

Fig. 1. Learning from imperfect human demonstrations. (Left) The human teaches a robot arm to play air hockey by showing examples of hitting the puck. (Right) The robot learns a control policy based on the human's data. Imitation learning approaches often try to mimic (at least a subset) of the actions demonstrated by the human — but this approach is fundamentally limited when human teachers make mistakes (like missing the puck). Counter-BC tries to imitate what the human *meant*, not necessarily what the human *showed*. More specifically, Counter-BC can modify the human's demonstrations within counterfactual sets in order to extrapolate underlying behaviors across the entire dataset. This leads to robots that learn consistent explanations of the human's demonstrations, e.g., learning to hit the puck at every state.

teacher: the robot infers its task based on behaviors the human demonstrates. Humans can make outright errors (e.g., pressing the wrong joystick button) as well as suboptimal demonstrations (e.g., taking a longer path to the goal). If we acknowledge that human teachers make these types of mistakes, then how should the robot reason over their potentially incorrect demonstrations in order to extract the intended task? Recent research attempts to resolve this question by introducing additional human guidance [15, 22, 24, 27, 43, 44, 48–50, 53] and by iteratively testing policies in the environment [6, 7, 9, 16, 25, 39, 46, 47, 52]. Within these prior works the robot asks a human expert to label the quality of some or all of their demonstrations [24, 43]. If the robot knows which trajectories are proficient and which contain errors, then it can bias its learning to mimic the high-quality examples while ignoring low-quality behaviors. Unfortunately, asking people to watch and label demonstrations is time consuming and subjective [24, 43] — particularly for large-scale datasets that contain thousands of examples collected across diverse scenarios [4, 21, 40]. This brings us back to the fundamental problem: given only a dataset of human examples, how should the robot extract a control policy to autonomously complete the desired task?

In this work we propose a shift in perspective when learning from imperfect humans. Instead of trying to determine how good or bad each individual human demonstration is, we recognize that (in many scenarios) the human teacher has an underlying policy that all of their examples are trying to convey. The human's imperfect teaching *obfuscates* this policy — i.e., makes it seem more

complex or variable than it really is — by unintentionally adding noise and suboptimal actions into the dataset. To recover what the human actually meant, we accordingly hypothesize that:

The robot learner should adjust the human teacher's demonstrations to reach a policy that simply and clearly explains their underlying trends.

Applying this hypothesis we introduce *Counter-BC*, a variant of behavior cloning that learns from noisy and imperfect human teachers. Our offline imitation learning approach does not require additional human labels or knowledge about the environment. Instead, Counter-BC automatically expands the demonstrations to consider nearby actions. These *counterfactual actions* capture the behaviors an imperfect human teacher might have been trying to show the robot. Returning to our hockey example in Figure 1, if the human inputs a motion that hits the puck off-center, the counterfactuals could include all actions within a radius Δ of that motion (including actions that miss the puck or hit the puck's center). Equipped with counterfactual actions, the robot can now modify the human's demonstrations (i.e., hypothesize that the human meant to show slightly different actions) in order to reach a simple and consistent explanation of the given dataset. This can result in control policies that align with the underlying behaviors the human teacher intended to convey, instead of trying to match any of the actions the human actually demonstrated. For instance: a robot arm trained with Counter-BC learns to hit the puck at each iteration, even though our given demonstrations occasionally missed that puck.

Overall, this work is a step towards robots that learn proficient policies from datasets provided by everyday human teachers. We make the following contributions:

Generalizing Behavior Cloning. We analyze the loss function used to train the robot's policy in standard behavior cloning algorithms, and show why that loss is ill-posed when learning from imperfect human data. To address this issue, we generalize the loss to include both counterfactual actions that expand the human's demonstrations and a probabilistic classifier that determines which counterfactual actions the robot learner should emulate. State-of-the-art approaches — including standard behavior cloning — can be viewed as simplified instances of this more general loss.

Deriving Counter-BC. Given the generalized loss function, we next derive our Counter-BC algorithm by selecting the counterfactuals and classifier. Our choices here cause the robot to learn a control policy which simultaneously does two things: i) the learned policy minimizes entropy to confidently output actions at each state, while ii) constraining those actions to remain close to the behaviors demonstrated by the imperfect human. We provide implementation details and public code for Counter-BC, and highlight that this offline imitation learning algorithm does not require any additional information beyond the human's demonstrations.

Finding Simple Explanations. A key feature of Counter-BC is the size of the counterfactual set. Designers can tune this size along a continuous spectrum by adjusting its radius Δ. We theoretically and empirically prove that increasing Δ (i.e., making the counterfactual sets larger) causes the robot to learn simpler policies, but these policies may increasingly diverge from the behaviors the human provided. When learning from expert users, we decrease Δ to closely mimic the demonstrated actions; when learning from inexperienced users we increase Δ to extrapolate underlying patterns.

Testing with Synthetic Demonstrations. We compare Counter-BC to state-of-the-art baselines across multiple environments. Within these tests we first collect synthetic human demonstrations with controlled noise distributions (e.g., Gaussian, uniform). We then explore how effectively Counter-BC and the baselines learn the desired task as we vary the noise within the human's demonstrations. In general, we find that Counter-BC outperforms existing methods regardless of what type of noise or suboptimal actions the synthetic human provides.

Learning from Real Humans. We conclude by learning from actual human data across simulated environments, standardized datasets, and an air hockey experiment. We again compare Counter-BC to existing offline imitation learning methods, and study how proficiently the robot learns the desired task based on examples from $N = 20$ human demonstrators. Our results indicate that robots which try to extrapolate what the human teacher meant by considering counterfactual actions (as opposed to copying or ignoring what the human actually did) are able to learn more effective policies from real human data.

Counter-BC is most applicable to scenarios where: (1) the robot learns from a fixed, offline dataset that cannot be re-collected or curated; (2) the task requires a continuous control policy with a relatively low-dimensional action space; and (3) the human teachers share a common intent (i.e., they are all trying to teach the same task, even if their skill levels differ). Concrete examples include assistive robotics applications where caregivers of varying skill demonstrate desired motions, manufacturing settings where workers teach robot manipulators new assembly steps, and imitation learning pipelines that aggregate crowd-sourced demonstrations.

2 RELATED WORK

Learning from noisy and imperfect human demonstrations is a core challenge for imitation learning [3, 30, 51]. Humans inevitably make errors when they show robots how to perform tasks — if robots naively mimic these mistakes, they will learn to perform the tasks incorrectly. The emergence of large-scale datasets for robot learning has made this issue increasingly important. To collect vast amounts of data, systems such as [4, 21, 40] rely on multiple human teachers of varying skill and proficiency. Given a heterogeneous dataset, how should the robot extract the correct behavior?

Improving Human Teaching. One approach is to help the human become a better teacher. By giving the human offline guidance before they demonstrate the task, or real-time feedback during their demonstration, the robot may improve the quality of the human's examples [11, 17, 36]. For instance, the robot can indicate when the human's current trajectory deviates from previous behaviors. Related works have accordingly leveraged augmented reality [33], haptic interfaces [42], physical markers [12], or computer screens [14] to assist the human teacher. For example, in robot-centric learning from demonstration [23, 26, 32, 35] the robot takes an active role during data collection by querying the human for additional demonstrations or feedback at states it finds uncertain. Once demonstrations are collected, the robot can also post-process the resulting data to filter out poor examples and human errors [8, 18]. Overall, these works are complementary but orthogonal to our efforts: instead of assisting the human during online data collection, we will advance how the robot learns from a fixed, offline, human-provided dataset.

A variety of learning algorithms have already been proposed for imperfect human demonstrations. Below we divide this related research into three groups: methods that rely on environment interactions, methods that rely on rankings, and methods that only consider the offline dataset.

Leveraging Environment Interactions. The first way in which robots can learn from noisy demonstrations is by extracting a reward [6, 9, 39] or discriminator [7, 46, 47, 52] from those demonstrations. The robot then interacts with the environment online (i.e., the robot arm attempts to complete the task) using reinforcement learning algorithms [6, 9, 39] or adversarial networks [7, 46, 47, 52]. Through these interactions the robot finds policies which maximize the reward, or which the discriminator cannot distinguish from expert human examples. One advantage of this paradigm is that — given ranked demonstrations — the robot can learn behaviors that are better than the best human demonstrations [6, 9]. But the downside is that this approach is *online*, and requires access to the environment dynamics through simulations or real-world rollouts. For

practical implementation, we instead focus on imitation learning algorithms which are purely *offline*, and do not require any environment interactions during training.

Leveraging Supervised Labels. Some research achieves offline imitation learning from imperfect demonstrations by labeling (or scoring) the dataset. These labels reflect the relative quality of an example: e.g., a trajectory that goes precisely to the goal might be labeled as an "expert" demonstration and given a high score, while a trajectory that overshoots the goal could be labeled as "novice" and given a low score. Based on this labeled dataset, the robot learns a policy offline which aligns with high-quality demonstrations while down-weighting or ignoring low-quality inputs. The key challenge here is how to label the dataset; i.e., how to discern expert and novice demonstrations. When the robot has access to its reward function, this process is straightforward [16, 25]; trajectories with higher rewards are higher quality. More generally, recent works focus on settings where a portion of the dataset is labeled, and the rest is unlabeled [15, 22, 24, 27, 43–45, 48–50, 50, 53]. The exact way that the labels are collected can vary. For instance, in [43] it is assumed that a human has marked some subset of the data as "expert." Alternatively, in [24] the human indicates the relative quality of a few demonstrations along a continuous or discrete scale. Regardless of how the labels are obtained, these works leverage the known labels to autonomously determine the quality of the remainder of the (unlabeled) dataset, and then train a policy to align with the high-quality examples. Of course, the downside to this framework is that labeling the demonstrations takes time and effort; a human expert must watch multiple demonstrations to grade their relative quality. We therefore seek a method which *does not require* any additional human inputs, rankings, or labels.

Leveraging Only Human Demonstrations. Most related to our proposed approach are methods that learn a policy given only the human's noisy and imperfect demonstrations. These methods do not assume access to environment interactions or supervised labels; instead, they seek to extrapolate what the human meant based only on trends in the human's data. In [34] the authors modify behavior cloning so that the robot's learned policy concentrates on the modes of the human's demonstrations. Put another way, the robot learns to consistently mimic the behaviors which are most common in the dataset. When these modes are correct, the robot learns the desired task — but if the majority of the dataset is not expert behavior, [34] falls short. Similarly, [2] assumes that human demonstrations are high-quality when those demonstrations output a consistent action at a given state, and low-quality when the demonstrated actions have high variance. We will experimentally compare our proposed algorithm to both of these state-of-the-art baselines. In general, the key difference between our method and [34] or [2] is that we enable the robot to incorporate counterfactuals (i.e., actions that the imperfect human could have intended, but did not actually demonstrate) to build a control policy that consistently explains the data. We note that [38] also uses counterfactuals, but applies them in settings where the robot has access to expert labels.

3 PROBLEM FORMULATION

We consider scenarios where a robot is learning a new task from one or more human teachers. Offline, the human teacher(s) provide examples of how the robot should complete the desired task. Because humans are noisy and imperfect, their demonstrations will inevitably contain mistakes (i.e., suboptimal actions that they do not want the robot to imitate). The robot's goal is to learn how to perform the task correctly, despite imperfect or suboptimal examples in the dataset.

Robot. Let $s \in \mathcal{S}$ be the system state and let $a \in \mathcal{A}$ be the robot's action. For instance, within our motivating example, s contains the robot's joint position, end-effector pose, and an image of the hockey table; the action a is the robot's joint velocity. The robot's actions cause the system state to transition according its dynamics. We do not assume access to these environment dynamics. Put

another way, when the robot arm moves to hit the hockey puck, the robot cannot explicitly model or simulate of how the puck will respond to its actions.

Human. The human has in mind a task they want the robot to learn. To teach the robot, the human demonstrates how to complete that task. Prior works have established multiple forms of demonstration: the human might kinesthetically guide the robot through the desired motion [1], teleoperate the robot along a trajectory [29], or employ graphical user interfaces, augmented reality, motion capture, passive observation, or natural language inputs to indicate the correct action at various states [10]. Regardless of how they are collected, the human's demonstrations result in a dataset $\mathcal{D}$ of examples. Within this dataset the human teacher labels each system state s with an action a, such that $\mathcal{D} = \{(s_1, a_1), \ldots, (s_n, a_n)\}$ is a collection of n state-action pairs. In our experiments we will focus on *offline datasets* (i.e., datasets collected prior to robot learning), but our approach also applies to settings where the human shows new state-action pairs online as the robot attempts to complete its task [37].

It is inevitable that the human will make mistakes when giving demonstrations. These mistakes may occur because the task is hard to perform, because the human is distracted or moving quickly, or because it is challenging to perfectly orchestrate the motion of a dexterous robot arm [5, 11, 28, 36]. To formulate teaching errors we define a^* as the *ideal action* the human meant to show the robot. If the human acted perfectly, they would have provided an ideal dataset $\mathcal{D}^* = \{(s_1, a_1^*), \ldots, (s_n, a_n^*)\}$. In practice, however, the noisy human actually demonstrates suboptimal actions a:

$$a = a^* + \epsilon, \quad \epsilon \sim P(\cdot \mid s) \tag{1}$$

where ϵ is the error between the ideal action a^* and the actual action a. At each state in the collected dataset $\mathcal{D}$, this action error is sampled from some unknown distribution $P(\epsilon \mid s)$.

Policy. Despite the noise inherent in the human's examples, the robot's objective is to learn a *control policy* that autonomously completes the demonstrated task. This control policy is a mapping from observed states s to robot actions a:

$$a \sim \pi_\theta(\cdot \mid s) \tag{2}$$

We instantiate the control policy π as a model (e.g., a neural network) parameterized by weights $\theta \in \Theta$. In practice, there are three different control policies that our formulation must consider: i) the ideal policy π^* that the human is trying to convey to the robot, ii) the noisy and imperfect policy π that the human actually follows when providing demonstrations, and iii) the robot's learned policy π_θ. Ideal actions a^* and the ideal dataset $\mathcal{D}^*$ are sampled from the ideal policy π^*, while the demonstrated actions $a = a^* + \epsilon$ and the collected dataset $\mathcal{D}$ are sampled from the imperfect policy π. We emphasize that the robot does not know either π^* or π. Instead, based on the dataset $\mathcal{D}$ induced by π, the robot seeks to learn a control policy π_θ that matches the ideal policy π^*.

Behavior Cloning. Recent works have found that methods founded on behavior cloning are a promising way to learn π_θ from suboptimal demonstrations [13, 30, 31, 34]. To better understand these baselines — and set the stage for our own approach — we here derive the loss function used to train the robot's policy in behavior cloning algorithms. Intuitively, the robot seeks to learn θ such that its control policy π_θ matches the expert policy π^*. We can measure the distance between distribution π_θ and distribution π^* using Kullback–Leibler (KL) divergence. Applying KL divergence to conditional probabilities, we reach:

$$D_{KL}(\pi^*, \pi_\theta) = C - \int_{s \in \mathcal{S}} \int_{a \in \mathcal{A}} \Big[\rho^*(s) \pi^*(a \mid s) \log \pi_\theta(a \mid s) \Big] \tag{3}$$

$$= C - \mathbb{E}_{s \sim \rho^*(\cdot), a^* \sim \pi^*(\cdot \mid s)} \Big[\log \pi_\theta(a^* \mid s) \Big] \tag{4}$$

where C is a constant term that does not depend on θ, and $\rho^*(s)$ is the distribution over states induced by the expert policy $\pi^*(a \mid s)$. Accordingly, to reduce the error between π_θ and π^* the robot should update θ to minimize the right side of Equation (4). This leads to the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s\sim\rho^*(\cdot),a^*\sim\pi^*(\cdot|s)}\Big[-\log \pi_\theta(a^* \mid s)\Big] \tag{5}$$

Approximating this expectation by sampling from the expert policy, Equation (5) becomes:

$$\mathcal{L}(\theta) \propto \sum_{(s,a^*)\in\mathcal{D}^*}\Big[-\log \pi_\theta(a^* \mid s)\Big] \tag{6}$$

which forms the standard loss function for behavior cloning algorithms [3, 20, 34]. Training the model weights θ to minimize the loss in Equation (6) corresponds to learning a control policy π_θ that matches the expert policy π^* across states in the dataset.

Unfortunately, this standard behavior cloning loss has a significant issue when applied to noisy and imperfect data. Based on the derivation in Equation (6), to learn the intended policy the robot should reason over the ideal dataset $\mathcal{D}^*$. But in reality the robot does not have access to $\mathcal{D}^*$. Instead, humans provide $\mathcal{D}$ — a dataset with noisy, imperfect, and suboptimal examples. One naive solution is just to replace $\mathcal{D}^*$ with $\mathcal{D}$ in Equation (6) and then train the robot's policy (we test this baseline in our experiments). However, doing so means that our robot is being trained to imitate incorrect human examples as opposed to the desired behaviors. We must therefore develop an approach that modifies Equation (6) to extrapolate from the noisy human demonstrations in dataset $\mathcal{D}$.

4 COUNTERFACTUAL BEHAVIOR CLONING

In this section we present counterfactual behavior cloning (Counter-BC), a modification of behavior cloning designed to learn from noisy and imperfect human demonstrations. Our core hypothesis is that human teachers have in mind an underlying control policy they are trying to convey, and the noise in their demonstrations obfuscates this control policy by introducing additional variation and complexity. As such, robot learners should reason over the human's demonstrations — and even modify those demonstrations — in order to reach a *simple explanation* of the given data. Counter-BC achieves this by expanding the demonstrations to include counterfactual actions: i.e., actions that the suboptimal human did not actually demonstrate, but may have intended to show the system. The robot learns a policy that minimizes entropy over these counterfactual actions; this policy is a *simple explanation* because the learned function is confident about what action to take at each state in the dataset (preventing the robot from overfitting to the variability and complexity introduced by the human's noisy behavior). In Section 4.1, we expand the standard behavior cloning loss to now account for counterfactual actions, and explain how this generalizes methods from prior work. Next, in Section 4.2, we derive the key terms of the new loss function, and show how the resulting policy selects counterfactuals that lead to minimal entropy. Finally, in Section 4.3, we outline the Counter-BC algorithm and practical implementation details.

4.1 Generalizing the Loss with Counterfactuals

Human teachers inevitably provide imperfect and suboptimal demonstrations. In Equation (1), we formulated the impact of those demonstrations: instead of showing actions a^* (sampled from the ideal policy π^*), everyday users input actions a (sampled from their suboptimal policy π). Put another way, at each (s, a) pair in the dataset $\mathcal{D}$, the human may have actually meant to show some other behavior. Our proposed method captures these alternative actions through *counterfactual sets*. Given a state-action pair, let $a' \in C(s, a)$ be the set of counterfactual actions that the human could have intended to demonstrate. By leveraging counterfactual sets C, we can expand the standard

behavior cloning loss from Equation (6) into a more general form:

$$\mathcal{L}(\theta) \propto \sum_{(s,a)\in\mathcal{D}} \sum_{a'\in C(s,a)} \Big[-R(s,a') \cdot \log \pi_\theta(a' \mid s) \Big] \tag{7}$$

Here C iterates through the actions the human might have wanted to show, and $R(s,a) \in [0,1]$ classifies these hypothetical actions along a continuous spectrum from expert ($R \to 1$) to non-expert ($R \to 0$). Equation (7) reduces to Equation (6) when: i) the counterfactuals include the ideal action $a^* \in C(s,a)$ at each state, and ii) $R(s,a) = 1$ for $a = a^*$, and $R(s,a) = 0$ otherwise. Introducing counterfactual actions is a shift in the robot's perspective: instead of strictly learning to match the behaviors in $\mathcal{D}$ demonstrated by the imperfect human teacher, now the robot can learn to mimic actions in $C(s,a)$ which the human never actually showed.

We highlight that the loss functions used by related works can also be viewed as simplifications of Equation (7). As summarized below, we reach these simplifications by making different choices for the counterfactual sets C or the classifier R in Equation (7):

- **Behavior cloning** [20]: $C(s,a) = \{a\}$, $R(s,a) = 1$ for all states.
- **Advantage-filtered behavior cloning** [16, 25]: $C(s,a) = \{a\}$, $R(s,a)$ is based on the advantage function or value associated with each state-action pair.
- **Discriminator-weighted behavior cloning** [48]: $C(s,a) = \{a\}$, $R(s,a)$ is a discriminator that assigns $R(s,a) \to 1$ to expert behavior and $R(s,a) \to 0$ to novice behavior.
- **ILEED** [2]: $C(s,a) = \{a\}$, $R(s,a)$ is the expertise level of the current human teacher where $R(s,a) \to 1$ when the human is an expert at the current state.
- **BCND** [34]: $C(s,a) = \{a\}$, $R(s,a) = \pi_{prev}(a \mid s)$ is the learned control policy from the previous training iteration.

One trend we observe is that — across each of these state-of-the-art methods — the robot always sets $C(s,a) = \{a\}$. This means that no actions besides the ones demonstrated by the human teacher can be explicitly treated as "expert" behavior (i.e., the robot does not reason over counterfactuals). As we will show, using counterfactuals is a fundamental change that enables the robot to interpret and explain the human's demonstrations in ways not available when the system is constrained to only imitating ($R \to 1$) or ignoring ($R \to 0$) the behaviors provided by the human teacher.

4.2 Learning Policies that Consistently Explain the Counterfactuals

Given the generalized loss in Equation (7), our next steps are to select the counterfactual set C and classifier R used by our algorithm. Our choices of C and R should enable the robot learner to reason over the counterfactuals and reach a simple explanation of the underlying control policy.

Counterfactual Set. The counterfactual set accounts for noise and error in the human's demonstrations by considering alternative actions that are close to the human's actual behavior. For each $(s,a) \in \mathcal{D}$, let $C(s,a)$ contain all actions within a radius Δ of the demonstrated a:

$$C(s,a) = \{a' \mid \|a - a'\| - \Delta \leq 0\} \tag{8}$$

The radius $\Delta \geq 0$ in Equation (8) is a hyperparameter specified by the designer. Decreasing Δ means that the robot will have a smaller counterfactual set, i.e., the learner assumes that the ideal action a^* is close to the demonstrated action a. Conversely, increasing Δ results in a larger counterfactual set, i.e., the learner can treat a variety of actions as a^* even if those actions are dissimilar from the human's demonstration a. We empirically find that the best performing values of Δ lie between these extremes, enabling the robot to extrapolate without discarding the human's underlying policy. Smaller values of Δ are appropriate for expert teachers (where the errors in a are slight), and larger values of Δ should be used with novice teachers (where the errors in a may be significant).

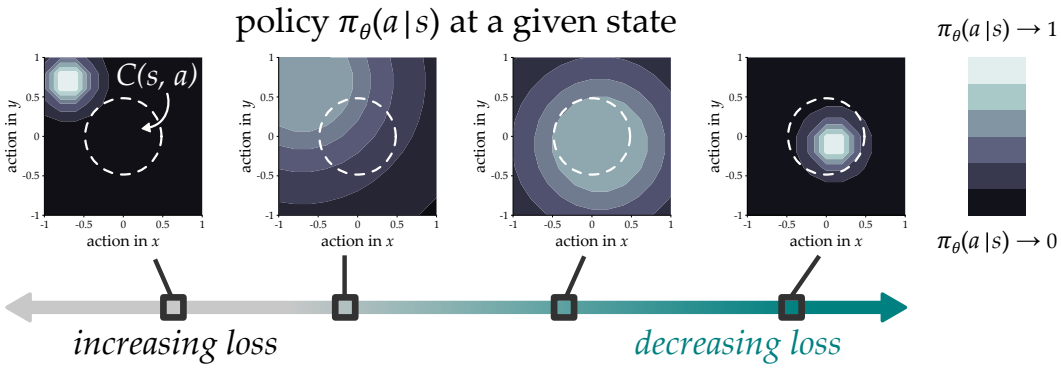

Fig. 2. Visualizing the loss in Equation (13) for a single state. Here the actions are two dimensional (i.e., in an x-y plane), and the demonstrated human action is at the origin. The counterfactual set lies within the dashed white circle ($\Delta = 0.5$). We show the probability distribution over actions for four different policies $\pi_\theta(a \mid s)$, where lighter regions indicate that the action is more likely. (Far Left) The maximum loss occurs when the policy confidently predicts an action outside the counterfactual. (Far Right) The minimum loss occurs when policy confidently predicts an action inside the counterfactual. Note that the predicted action does not need to be the demonstrated action; specifying any action within $C(s, a)$ can minimize the proposed loss.

We note that the counterfactual set $C(s, a)$ should be constrained to the action set $\mathcal{A}$, such that $C(s, a) \subseteq \mathcal{A}$. Put another way, the robot should not consider counterfactuals that do not belong to its set of feasible actions. We here assume that the designer knows the action space. We further note that the counterfactual assumption is most natural when the action space is continuous and Euclidean proximity reflects physical similarity (e.g., joint velocity or end-effector velocity commands). In settings where this assumption does not hold (for instance, when nearby values in the action space correspond to semantically distinct behaviors) designers should reduce Δ or consider a domain-specific distance metric when defining $C(s, a)$. Alternatively, redesigning the action parameterization so that physically similar motions are represented as numerically nearby values can recover the benefits of Counter-BC in such settings.

Probabilistic Classifier. The probabilistic classifier $R : \mathcal{S} \times \mathcal{A} \to [0, 1]$ determines which actions the control policy should emulate. Given a proposed counterfactual a', increasing $R(s, a')$ indicates that a' is likely the intended human action. We have already introduced a term that operates similarly to R: the control policy $\pi_\theta(a \mid s)$ estimates the likelihood that a given action is part of the desired policy at state s. Recognizing this alignment, we define the probabilistic classifier as a restriction of probability distribution π_θ with support over the counterfactual set:

$$R(s, a') = \hat{\pi}_\theta(a' \mid s), \quad \hat{\pi}_\theta(a' \mid s) = \frac{\pi_\theta(a' \mid s)}{\sum_{z \in C(s,a)} \pi_\theta(z \mid s)} \tag{9}$$

Because the normalizer of $\hat{\pi}_\theta$ considers only the counterfactual set C, by definition $\hat{\pi}_\theta$ is a probability distribution that sums to one across the counterfactuals. Put another way, $R(s, a')$ in Equation (9) expresses the probability that counterfactual a' is really the action a^* the human teacher meant to demonstrate. For instance: in the special case where $C(s, a) = \{a\}$ and the robot does not consider counterfactuals, then $R(s, a) = 1$ and the robot is confident that the demonstrated action a is the intended action a^*. In what follows we will prove that this choice for probabilistic classifier R (when combined with counterfactual sets C) biases the robot learner towards policies that provide simple explanations for the human's noisy demonstrations.

Counter-BC Loss. To derive the loss function for Counter-BC we plug Equation (8) and Equation (9) back into Equation (7). Starting with our given choices of the counterfactual set and classifier, and then manipulating the terms, we eventually reach the cross entropy between $\hat{\pi}_\theta$ and π_θ:

$$L(\theta) \propto \sum_{(s,a)\in\mathcal{D}} \sum_{a'\in C(s,a)} \Big[-\hat{\pi}_\theta(a' \mid s) \cdot \log \pi_\theta(a' \mid s) \Big] \tag{10}$$

$$= \sum_{(s,a)\in\mathcal{D}} \sum_{a'\in C(s,a)} \left[-\hat{\pi}_\theta(a' \mid s) \cdot \log \frac{\pi_\theta(a' \mid s) \cdot \hat{\pi}_\theta(a' \mid s)}{\hat{\pi}_\theta(a' \mid s)} \right] \tag{11}$$

$$= \sum_{(s,a)\in\mathcal{D}} \sum_{a'\in C(s,a)} \left[-\hat{\pi}_\theta(a' \mid s) \cdot \log \hat{\pi}_\theta(a' \mid s) + \hat{\pi}_\theta(a' \mid s) \cdot \log \frac{\hat{\pi}_\theta(a' \mid s)}{\pi_\theta(a' \mid s)} \right] \tag{12}$$

$$= \sum_{(s,a)\in\mathcal{D}} H\Big(\hat{\pi}_\theta \mid s, C(s,a)\Big) + D_{KL}\Big(\hat{\pi}_\theta, \pi_\theta \mid s, C(s,a)\Big) \tag{13}$$

Here H is the conditional entropy over the restricted policy's actions given state s and counterfactual set C, and D_{KL} is the KL divergence between $\hat{\pi}_\theta$ and π_θ given state s and counterfactual set C. In practice, minimizing Equation (13) does two things. The first term drives our restricted policy $\hat{\pi}_\theta$ to have low entropy over the counterfactual set. The second term forces the robot's full policy π to match the restricted policy $\hat{\pi}_\theta$. Putting these terms together means that the robot will learn a policy π_θ that confidently outputs actions (i.e., minimizes entropy), while constraining those actions to be close to the behaviors demonstrated by the imperfect human.

To better visualize this loss function, we provide an example in Figure 2. The loss is *highest* when the policy π_θ confidently predicts an action outside of the counterfactual set (far left). Conversely, the loss is *lowest* when π_θ confidently predicts an action within $C(s,a)$ (far right).

Viewed at a single state s, minimizing Equation (13) results in a robot policy $\pi_\theta(a \mid s)$ that confidently outputs an action $a \in C(s,a)$. We next consider what happens when we apply this loss function across the entire dataset $\mathcal{D}$ of state-action pairs. At each state in this dataset, the robot can learn to output any action within the corresponding counterfactual $C(s,a)$. The radius Δ in Equation (8) defines the size of this counterfactual set. When $\Delta = 0$ the counterfactual set is simply the action demonstrated by the human, and the robot tries to precisely imitate the human's actions. As Δ increases, the size of the counterfactual set also increases to include a wider range of actions — at the extreme, the counterfactual set is $\mathcal{A}$. So given this increasing range of options, which actions will a policy trained with Equation (13) learn to output? If we assume that the model π_θ is locally Lipschitz [19], then for small changes in the state s, the model is inherently structured to output small changes in the action a. In other words — if the robot is free to choose any action within this counterfactual set — the policy is likely to select an action at state s' that is similar to the action at nearby state s. Expanding the radius Δ therefore means that the robot learns a "simpler" policy, in the sense that the learned policy maintains consistent actions across nearby states.

Intuition. By extrapolating across the counterfactual set, our policy seeks to *explain* the human's data, and can *modify* the expert actions (i.e., change the selected counterfactual) in order to reach a more efficient explanation of the dataset. Overall, the design of Equation (13) formalizes our hypothesis that there is an underlying function behind the human's behaviors, and the robot may need to tweak the dataset in order to remove suboptimal actions and recover that intended function.

In Figure 3, we show an instance of using the Counter-BC loss to recover the underlying task — here a simulated human is demonstrating the absolute value function. Within this example we also highlight how tuning Δ affects the recovered policy. In general: *increasing Δ means the robot will recover a simpler function, but this function may increasingly diverge from the dataset.*

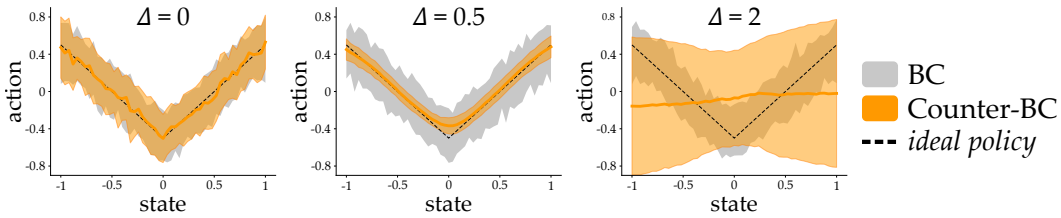

Fig. 3. Recovering the underlying function with Counter-BC. In a 1-dimensional environment the simulated human noisily demonstrates the absolute value function $a = |s| - 0.5 + \epsilon$, where $\epsilon \sim \mathcal{U}(-0.5, +0.5)$. We plot the robot's learned policy averaged across 50 runs; shaded regions show the standard deviation. With behavior cloning (BC) the robot tries to precisely match the noisy data, resulting in a more complex explanation of the human's demonstrations. (From left to right) Counter-BC with different values of hyperparameter Δ. Increasing Δ causes Counter-BC to recover increasingly simple explanations of the data by reasoning over larger counterfactual sets, but the resulting policies may diverge from the human's examples.

Algorithm 1 Counterfactual Behavior Cloning (Counter-BC)

Input dataset $\mathcal{D} = \{(s_1, a_1), \ldots, (s_n, a_n)\}$
Initialize control policy $\pi_\theta(a \mid s)$ with weights θ
Specify the radius $\Delta \geq 0$ used for sampling counterfactual actions
Sample counterfactual actions $a' \in C(s, a)$ for each $(s, a) \in \mathcal{D}$ ▷ Equation (8)
for epoch $\in 1, 2, \ldots, K$ **do**
    $\mathcal{L}(\theta) \leftarrow 0$ ▷ Initialize loss
    for $(s, a) \in \mathcal{D}$ **do**
        $\log \pi_\theta \leftarrow \left[\log \pi_\theta(a' \mid s) \text{ for each } a' \in C(s, a)\right]$ ▷ Compute vector of length $|C(s, a)|$
        $\hat{\pi}_\theta \leftarrow \text{softmax}(\log \pi_\theta)$ ▷ Equation (9)
        $\mathcal{L}(\theta) \leftarrow \mathcal{L}(\theta) - \hat{\pi}_\theta \cdot \log \pi_\theta$ ▷ Equation (10) with dot product between vectors
    end for
    $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$ ▷ Update model weights to minimize loss
end for
Return trained control policy $\pi_\theta(a \mid s)$

When hyperparameter $\Delta \to 0$ then $C(s, a) = \{a\}$ and minimizing Equation (13) is equivalent to standard behavior cloning. Hence, as $\Delta \to 0$ the robot learns a policy that is sufficiency complex to match the given state-action pairs as precisely as possible. At the other extreme, when $\Delta \to \infty$ the counterfactual set is equal to the entire action set $C(s, a) = \mathcal{A}$. This in turn means that $\hat{\pi}_\theta = \pi_\theta$ in Equation (9), and Equation (10) becomes the conditional entropy of $\pi_\theta(a \mid s)$. Robots can minimize this conditional entropy with any deterministic function that outputs a consistent action at each state — for example, simply outputting a line in Figure 3. Accordingly, as $\Delta \to \infty$ the robot can fit the provided dataset with increasingly simplistic but inaccurate policies.

4.3 Implementation

Equipped with our understanding of the loss function, we are now ready to present Counter-BC (see Algorithm 1). Counter-BC is an offline imitation learning approach that inputs a dataset of state-action pairs $\mathcal{D}$ and the designer-specified hyperparameter Δ. During training, Counter-BC learns a control policy with parameters θ to minimize Equation (10): this training phase does not

require any human labels or environment interactions. Code for implementing Counter-BC is located here: https://github.com/VT-Collab/Counter-BC

In the experiments reported below we instantiate π_θ as a Gaussian policy with independent covariance. This policy is constructed from a fully connected neural network with two hidden layers and ReLU(·) activation functions. Actions are sampled from the policy using the reparameterization trick. To train the policy network we employ an Adam optimizer with a learning rate of $1e^{-3}$. We selected the learning rate, batch size, and number of epochs through preliminary testing with standard behavior cloning (see our public repository for additional hyperparameter values). Unless otherwise stated, we normalized the actions to a unit sphere, and used $\Delta = 0.4$ in our experiments. This value is based on the tradeoff observed in Figure 3.

When designers are implementing Counter-BC we recommend normalizing the action space to a unit sphere, and then selecting radius Δ between 0.2 and 0.5. If the robot is learning from a naive user who often makes mistakes, the value of Δ should likely be increased towards 0.5. Conversely, if the users provide high quality demonstrations, then the value of Δ should be decreased towards 0.2. We emphasize that these guidelines are heuristics, and the process of selecting Δ is one of the limitations of Counter-BC. We sample the counterfactual actions $a' \in C(s, a)$ before the training loop (Line 4 of Algorithm 1) for computational speed; these actions do not need to be resampled or updated throughout the learning process. On average, we found that Counter-BC took roughly $5X$ longer to train than standard behavior cloning.

5 SIMULATIONS

In this section we compare Counter-BC to state-of-the-art alternatives in simulated environments. Actual participants and synthetic users provide demonstrations within each environment, and we leverage behavior cloning variants to try and recover the intended task from these imperfect datasets. Our simulations explore i) how different types of noise and suboptimality affect the learner's performance, ii) how the learner's behavior changes as the amount of data varies, iii) how the level of noise in the human's demonstrations impacts the learner, and finally iv) how adjusting hyperparameter Δ alters results with Counter-BC.

Environments. We selected four different environments to perform our simulations: *Intersection*, *Cartpole*, *Car Racing*, and *Robomimic*. Images of the environments are shown in Figure 4. Overall, these environments and tasks span different levels of complexity — from balancing a one-DoF inverted pendulum, to driving a vehicle based on RGB images, to controlling a robot arm that grasps and manipulates objects. In order to collect real human demonstrations, we needed to choose environments where humans can actually control the ego agent (i.e., teleoperate the robot). We also tried to pick environments that were consistent with related works, and leveraged existing datasets for learning from imperfect humans. Below we briefly summarize each environment.

Intersection. A multi-agent driving task from [31]. Two cars are crossing an intersection: the ego agent (which is controlled by a simulated or real human) and another agent (which is fully automated, and updates its motion in response to the ego agent). Both vehicles are attempting to reach their respective goals on the opposite side of the intersection while avoiding collisions with one another. The state $s \in \mathbb{R}^4$ includes the x-y position of both cars, and the action $a \in \mathbb{R}^2$ updates the position of the ego agent. The performance of the robot learner is measured by its total reward across an interaction; this reward is the negative of Equation (12) in [31]. Each demonstrated trajectory contains 20 state-action pairs.

Cartpole. A standard environment for robot learning [41]. The state $s \in \mathbb{R}^4$ captures the pose of a cart and attached pendulum, and actions $a \in \mathbb{R}^1$ move the cart left or right along a continuous action space in order to balance the inverted pendulum. The robot's performance is measured by

the number of timesteps the agent can keep the pendulum from falling over (up to a maximum of 200 timesteps). Related papers on learning from noisy human demonstrations have leveraged this environment or similar variations of the inverted pendulum problem [9, 38, 39, 49, 53]. On average, a human demonstration of the task contains 109 state-action pairs.

Car Racing. A single-agent driving along a randomly generated road [41]. Unlike our other environments — where the robot has direct access to the system pose — here the robot observes 96×96 RGB images of the scene. Specifically, the robot's state is a sequence of four frames showing the environment at timesteps t, $t-1$, $t-2$, and $t-3$. The learner must map this sequence of visual inputs into continuous actions $a \in \mathbb{R}^3$ that steer, accelerate, and brake. Humans provide demonstrations where they try to quickly advance while keeping the car on the road; the learner's performance is measured by the distance traveled. For this environment we use a CNN to encode the observed images into state vectors. This CNN is trained alongside the robot's Gaussian policy. On average, a human demonstration of the task contains 2225 state-action pairs.

Robomimic. An existing dataset for learning robot manipulation tasks from real human demonstrations [30]. The dataset includes multiple tasks: we implement the *Can* task and *Multi-Human* dataset. In this task a FrankaEmika robot arm reaches to grasp a cylindrical can, carry it, and then drop it in the appropriate bin. The system state $s \in \mathbb{R}^{23}$ includes the robot's pose, the can's pose, and the can's pose relative to the robot arm. Actions $a \in \mathbb{R}^7$ move the robot's end-effector and actuate its gripper. The multi-human dataset includes 300 *Can* demonstrations collected from six different humans: two "better" operators, two "okay" operators, and two "worse" operators. Similar to [2, 24, 50], we directly leverage this existing multi-human dataset; no additional or synthetic demonstrations are collected for this environment. Performance is measured by the percentage of trials where the robot learner successfully completes the entire manipulation task. On average, a human demonstration of the task contains 209 state-action pairs, but this varies across the human's proficiency level [30].

Human Demonstrations. In *Intersection*, *Cartpole*, and *Car Racing* environments we collected real and synthetic human demonstrations. Within the synthetic demonstrations we injected different types of controlled noise or suboptimal behaviors. A list of demonstrators is provided below:

- **Real Humans.** We recruited $N = 20$ in-person participants (3 female, average age 24 ± 4 years) to teleoperate the ego agent. These participants were Virginia Tech students that provided informed written consent following university guidelines (IRB #23-1237). After practicing the tasks for up to five minutes, users provided demonstrations to convey the desired behavior in each environment. In all our experiments the participants were instructed to do their best when teaching the robot, while acknowledging that mistakes happen and the user does not always need to be perfect. Participants started with *Intersection*, where they completed 50 interactions to provide (1000 state-action pairs). Next, participants teleoperated the simulated *Cartpole* for 1000 timesteps, and finally participants drove *Car Racing* one complete lap around its track. No participant data was excluded. The state-action pairs from all participants were aggregated into a single dataset for each environment, and then the algorithms listed below were trained on a randomly sampled subset of state-action pairs sampled from this aggregated dataset. Participants were compensated at a rate of $20USD/hour for their time. The total study duration was approximately 45–60 minutes per participant, including a consent and briefing period (≈5 minutes), practice trials in each simulated environment (≈15 minutes total), data collection across the three simulated environments (≈15 minutes), and the air hockey robot experiment (≈10 minutes including setup and recorded demonstrations, see Section 6).

- **Uniform.** Synthetic humans demonstrated the task by sampling actions according to Equation (1), where a^* is the optimal action that maximizes task performance and P is a uniform distribution: $\epsilon \sim \mathcal{U}(-\sigma, \sigma)$. Intuitively, these simulated human teachers had uniform noise added to their demonstrations.
- **Gaussian.** Synthetic humans sampled actions according to Equation (1), where a^* is the optimal action and P is an unbiased Gaussian distribution: $\epsilon \sim \mathcal{N}(0, \sigma^2)$.
- **ϵ-Greedy.** With probability $1 - \sigma$ the demonstrator provides the optimal action so that $a = a^*$. Otherwise the synthetic human injects uniform noise: $\epsilon \sim \mathcal{U}(-\sigma, \sigma)$.

We emphasize that collecting actual human data is critical since the purpose of our approach is to learn from humans; *synthetic data may never capture the types of mistakes that real users make*. At the same time, testing with synthetic data is theoretically useful because it provides evidence about which noise distributions are most advantageous or challenging for each method. We performed offline testing to qualitatively compare the human datasets with our synthetic options. In *Intersection* the human-provided demonstrations have an average reward of 12.3, while the synthetic demonstrations have an average reward of 13.1. In *Cartpole* the average human and synthetic reward is 30.3 and 26.6, respectively. These similar average rewards suggest that the synthetic demonstrations are not entirely dissimilar from the human's behavior in terms of task performance. However, we do not claim that this comparison establishes equivalence of noise structure. Testing with real human data remains essential, and the synthetic experiments are intended to provide controlled insights into how different noise types affect each algorithm.

Learning Algorithms. Given a dataset $\mathcal{D}$ of state-action pairs provided by the real or synthetic human, the robot used offline imitation learning to train a policy π_θ. We compared policies learned with **Counter-BC** (Algorithm 1) to three state-of-the-art baselines for learning from noisy humans.

- **BC** [20]: Standard behavior cloning. The robot trains a Gaussian policy to try and exactly match the actions demonstrated by the human teacher.
- **Weighted-BC**: A variant of behavior cloning where the robot learns the relative expertise of different state-action pairs in the dataset. Our approach here is based on ILEED from [2]. Specifically, we introduce a network $\rho_\phi(s) \rightarrow [0, 1]$ that inputs the state and outputs a normalized score. This score estimates the expertise level of the demonstrations, i.e., as $\rho_\phi(s) \rightarrow 1$ the robot is confident the actions shown at s precisely demonstrate the task. The learner then sets $R(s, a) = \rho_\phi(s)$, and simultaneously trains both ρ_ϕ and π_θ using the loss from Equation (7) with no counterfactual actions. In practice, Weighted-BC enables the learner to distinguish between states where the human teachers provide consistent, expert actions, and states where the teachers input noisy, suboptimal actions.
- **BC-RNN** [30]: A version of behavior cloning where the robot reasons over a history of recent states. We combine our standard **BC** approach with a Gated Recurrent Unit (GRU). This GRU inputs a sequence of 5 states, and outputs a latent state representation based on the state history. The Gaussian policy then reasons over the latent state to select actions that match the behaviors of the human teacher. Our intuition is that this history-based approach may be able to disambiguate between local noise and underlying action patterns.
- **BCND** [34]: A behavior cloning variant that shifts the learned policy towards the mode of the distribution. For this approach we set $R(s, a) = \pi_{prev}(a \mid s)$, the policy trained at the previous iteration. In practice **BCND** learns to emulate the human's actions at states where the demonstrations are consistent, and ignores the human's actions with high variability.

Note that we did not include offline reinforcement learning algorithms since the robot learner does not have access to the task reward, and thus these methods cannot be applied to our setting. In our implementation Counter-BC and the baselines are instantiated as Gaussian policies. The

learning rate, batch size, and number of epochs are held constant across all methods to mitigate against biased results. We note that ILEED [2] was originally designed for settings where there are multiple human teachers, and the robot knows which demonstrations belong to which teacher. The **Weighted-BC** baseline — which functions as a simplified version of ILEED — may therefore perform poorly because we do not assume any labels on the dataset, and so **Weighted-BC** cannot connect individual datapoints with any specific human teacher. Additional training details can be found in our public code repository: https://github.com/VT-Collab/Counter-BC

5.1 How Does Counter-BC Perform with Different Types of Imperfect Demonstrations?

The results of our first simulation are shown in Figure 4. Each row of this figure corresponds to a different type of human demonstrator: actual participants, and then synthetic teachers with uniform, Gaussian, or ϵ-Greedy. For the *Robomimic* environment we only collect data from real humans by leveraging the standardized dataset in [30]. In general, we observe that **Counter-BC** learns more proficient policies than the baselines. We particularly highlight the performance of **Counter-BC** in the *Car Racing* environment — where the control policies are based on image observations, and not direct state measurements — as well as the performance across environments with real human demonstrations. By contrast, the one type of imperfect demonstration where **Counter-BC** does not consistently reach the highest reward is ϵ-Greedy; here **BCND** occasionally outperforms **Counter-BC**. This result makes sense because **BCND** is designed to work with ϵ-Greedy. More specifically, **BCND** trains the policy so that it is better aligned with the human teacher's most common actions — and when our simulated humans are ϵ-Greedy, their most common action is the optimal action a^*. For all other types of imperfect demonstrations we find **Counter-BC** leads to improved performance. Taken together, this suggests that **Counter-BC** learns the intended task despite controlled noise or uncontrolled errors in the human's demonstrations.

5.2 How Does Counter-BC Perform with an Increasing Number of Demonstrations?

Within Figure 4 we also measure how the performance of each algorithm changes with varying amounts of human data. The number of demonstrated state-action pairs is shown along the x-axis of each individual plot. Intuitively, we would expect the system to perform better when given more examples: the more times the human shows the robot how to act, the more proficiently it should behave during rollouts. Our results affirm this intuition. Additionally, we see that **Counter-BC** outperforms the alternatives across different levels of demonstrations. This suggests that **Counter-BC** is not limited to settings with a specific number of demonstrations, but can be used across varying dataset sizes. Additional information about the percent improvement between **Counter-BC** and the baselines can be found in the Appendix (Section 8).

5.3 How Does Counter-BC Perform with Varying Levels of Noise?

So far we have varied the type of human teacher and the amount of data; next, we adjust the level of noise within the human's examples. Figure 5 corresponds to a setting where the robot is learning from increasingly suboptimal and imperfect demonstrations. When $\sigma = 0$ the synthetic human teacher perfectly demonstrates the task with optimal actions a^*, and as σ increases the human deviates from the optimal actions with uniform noise $\epsilon \sim \mathcal{U}(-\sigma, \sigma)$. Within the *Intersection*, *Cartpole*, and *Car Racing* environments the maximum magnitude of an action dimension is 1; hence, $\sigma = 1$ means that the size of the noise matches or exceeds the size of the optimal actions. We find that — relative to the baselines — **Counter-BC** is increasingly robust as σ grows and the synthetic human's demonstrations become more noisy and suboptimal. When the human's demonstrations have no noise ($\sigma \rightarrow 0$) then each method is roughly equivalent. But as the size of the noise increases ($\sigma \rightarrow 1$) we see the advantages of **Counter-BC** in reasoning over counterfactual actions, searching

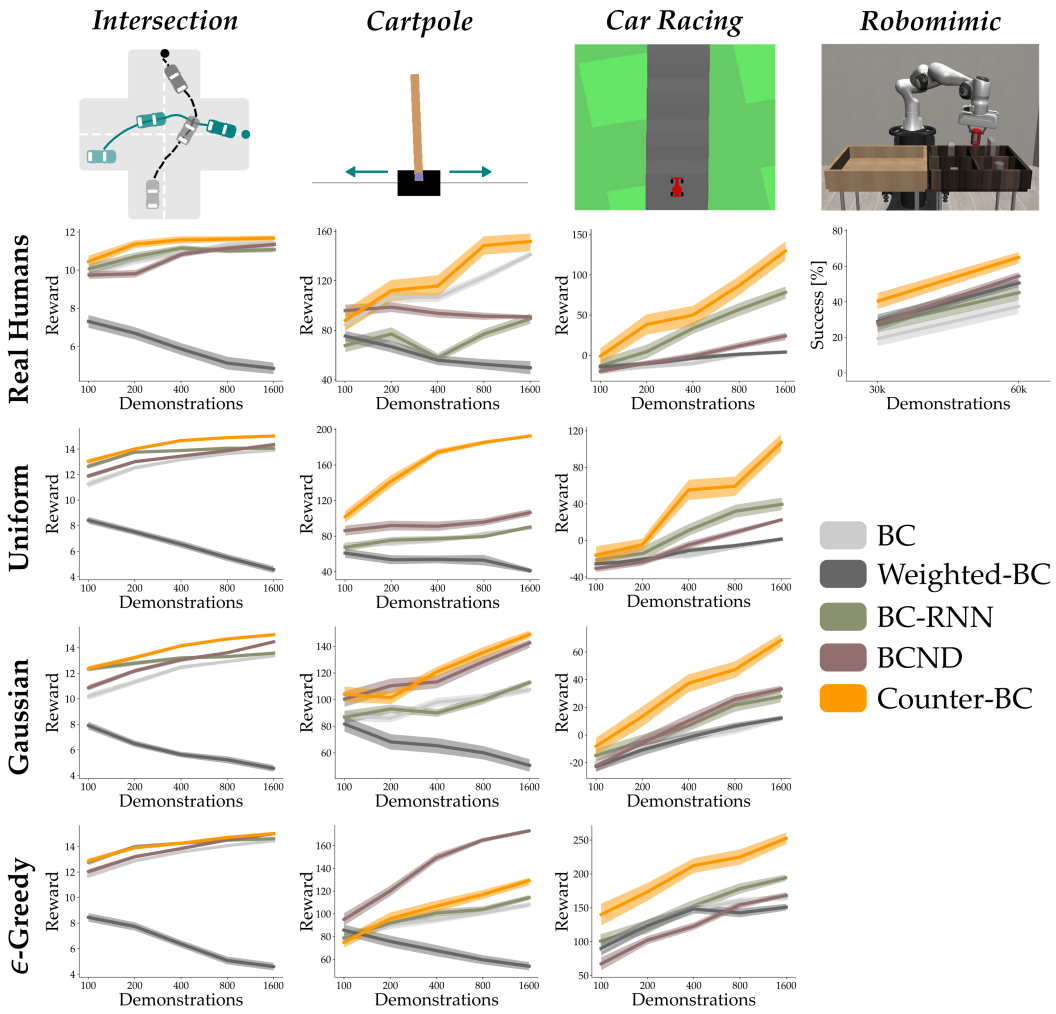

Fig. 4. Learning from real and simulated humans (see Sections 5.1 and 5.2). Every column shows a different environment, and the rows correspond to demonstrations from real humans, or synthetic operators ~~with~~ uniform, Gaussian, and ϵ-Greedy. Here "demonstrations" refers to the number of state-action pairs in the training dataset. Plots illustrate the performance of the policy learned by each algorithm as a function of the number of demonstrated state-action pairs (higher reward or success is better). For *Robomimic* we tested the *Can* task with standardized demonstrations from the Multi-Human dataset [30]. Results are averaged across 50 runs with different seeds; shaded regions show standard error. A repeated measures ANOVA shows that algorithm type had a statistically significant effect on reward across all environments and human teachers. Post hoc tests confirm that **Counter-BC** led to higher rewards on average as compared to **BC** ($p < .01$), **Weighted-BC** ($p < 0.001$), **BC-RNN** ($p < .01$) and **BCND** ($p < .01$).

for simple policy explanations, and extrapolating the underlying task. We note that for *Car Racing* at high noise levels ($\sigma \rightarrow 1$), all methods approach chance performance. This environment is particularly sensitive to noise because: (i) the policy maps high-dimensional image observations to actions, requiring a reliable association between visual features and correct steering/braking commands; and (ii) the training datasets used in this experiment (400 state-action pairs) represent

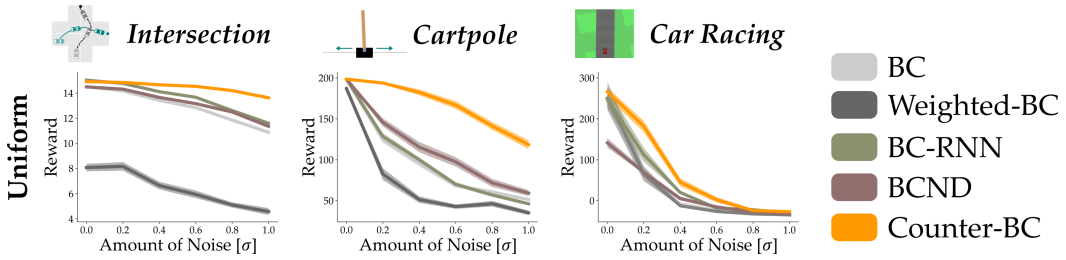

Fig. 5. Learning from increasingly noisy demonstrations (see Section 5.3). To regulate the amount of noise, we simulated a human teacher. This teacher selected actions $a = a^* + \epsilon$, where a^* is the optimal action and $\epsilon \sim \mathcal{U}(-\sigma, \sigma)$ is uniform noise. Increasing σ led to more noisy, imperfect, and suboptimal human teaching. The policies learned by each algorithm degrade as σ increases, but **Counter-BC** is more robust than the baselines. Results are averaged across 50 runs with demonstrations containing 400 state-action pairs. We conducted a repeated measures ANOVA using this data. Post hoc tests show that **Counter-BC** led to higher average rewards than **BC** ($p < .01$), **Weighted-BC** ($p < 0.01$), **BC-RNN** ($p < .01$) and **BCND** ($p < .01$).

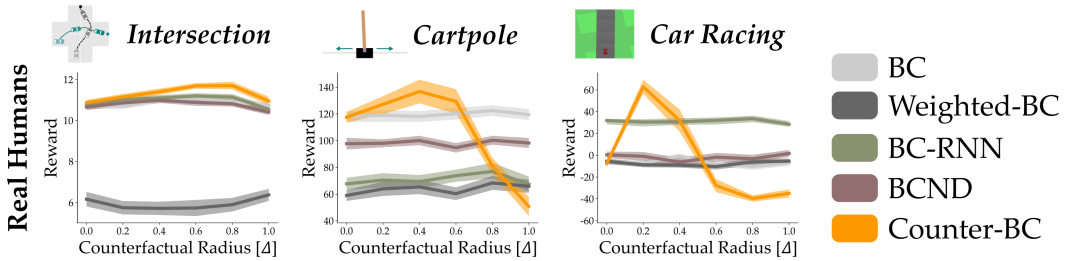

Fig. 6. Learning from demonstrations while varying hyperparameter Δ (see Section 5.4). Note that the hyperparameter Δ only applies to **Counter-BC**, but the performance of the baselines slightly varies across the x-axis because the training dataset and testing start states are sampled randomly at each run. Consistent with our theoretical analysis from Section 4.2, when $\Delta = 0$ then **Counter-BC** is equivalent to **BC**. Increasing Δ enlarges the counterfactual sets and leads to simpler explanations of the human's demonstrations. Once Δ is increased too much, the robot begins to oversimplify the underlying human policy, resulting in degraded performance. Shaded regions show the standard error across 50 end-to-end runs with demonstrations that have 400 state-action pairs. These demonstrations were collected from $N = 20$ in-person participants.

only a fraction of a full lap, making it difficult to learn a globally consistent control policy under heavy noise. At lower noise levels and with more data, **Counter-BC** continues to outperform the baselines in this environment (see Figure 4 and Figure 5).

5.4 How Does Adjusting Hyperparameter Δ Affect Counter-BC?

We conclude our simulations by testing the effect of hyperparameter Δ on **Counter-BC**. In Section 4.2 and Figure 3 we theoretically showed that $\Delta \to 0$ causes the robot to exactly imitate all of the human's actions, reducing **Counter-BC** to standard **BC**. Conversely, as Δ increases the robot should learn simpler functions by imitating actions close to (but not the same as) the behaviors actually demonstrated by the human teacher, resulting in learning capabilities beyond the current state-of-the-art. In Figure 6 we put this theory to the test with real human data. For datasets from $N = 20$ participants in *Intersection*, *Cartpole*, and *Car Racing*, we learn robot policies while adjusting the value of Δ. As expected, when $\Delta = 0$ the performance of **Counter-BC** equals the behavior of **BC**. Increasing Δ enables **Counter-BC** to expand the human's demonstrations and extrapolate

underlying functions, improving performance of the learned policy. However, when Δ is increased too far the robot eventually recovers policies that are simpler than what the human intended (i.e., missing nuances in the correct behavior), harming the performance of the autonomous robot. This suggests both a trade-off and guidelines for implementation. We recommend that designers start with low values of Δ, and gradually increase that hyperparameter to improve performance. We also recognize that one downside of **Counter-BC** is that — if designers initialize with values of Δ that significantly exceed the noise in the teacher's demonstrations — the performance of our algorithm may actually fall below the **BC** baseline.

6 USER STUDY

Our previous experiments focused on simulated environments with real and synthetic human teachers. In our final study, we now test Counter-BC on a real-world robot arm (Franka FR3) with demonstrations provided by $N = 20$ in-person participants. The task matches our running example from Figure 1: human teachers teleoperate the robot arm to repeatedly hit an air hockey puck across a table. Based on the participants' demonstrations, the robot must learn to align its paddle with the oncoming puck, and then move forward to strike that puck with the correct timing. See examples of this game in our supplemental video: https://youtu.be/XaeOZWhTt68

Data Collection. Our experiment has two phases: data collection and evaluation. During *data collection* each individual participant sits next to the robot and uses a joystick to teleoperate the arm in real-time. The arm's end-effector is constrained to move along the planar surface of an air hockey table. A red puck glides along this table; participants try to control the robot to hit that puck so it bounces off the opposite side and returns towards the robot arm. Based on the speed and location where participants hit the puck, the puck may move at various angles and velocities (e.g., colliding with the table's sides). Hence, each iteration naturally forces users to vary the way they move the robot arm. In total, $N = 20$ users provided demonstrations for the air hockey task; these were the same users that taught the simulated robots in Section 5. Participants practiced teleoperating the robot arm with a joystick for 2 to 5 minutes. Users then controlled the robot to play air hockey for 2 minutes of recorded demonstrations. This corresponds to roughly 2400 state-action pairs per user. These participants are not perfect teachers — their motions include examples where they miss the puck or accidentally hit it off-center.

Throughout data collection the robot records the position of the puck with an overhead camera (see Figure 7, Left). The robot also saves its end-effector locations and the user's commanded actions. Overall, the robot's state $s \in \mathbb{R}^6$ consists of the x-y position of its end-effector, the x-y position of the puck at the current timestep, and the puck's x-y position at the previous timestep. Actions $a \in \mathbb{R}^2$ are end-effector velocities along the surface of the table normalized between 0 and 1. Each timestep of data collection yields a new (s, a) pair for our aggregated dataset $\mathcal{D}$.

Independent Variables. After collecting dataset $\mathcal{D}$ we next move to the *evaluation* phase. During evaluation the robot trains its policy using subsets of the aggregated demonstrations, and then rolls out this learned policy to autonomously play air hockey. We modulate two variables: the *algorithm* the robot uses to train its control policy, and the *amount of data* leveraged by that algorithm. We compare the same offline imitation learning methods as in Section 5, with the exception of BC-RNN. These include our proposed **Counter-BC** approach, as well as state-of-the-art baselines **BC** [20], **Weighted-BC** [2], and **BCND** [34]. Each algorithm learns from the same human demonstrations. We sample these demonstrations from the aggregated dataset $\mathcal{D}$; specifically, we measure the robot's performance when trained on 200, 400, 800, 1600, and 3200 state-action pairs. For reference, a dataset size of 200 corresponds to 10 seconds of human demonstration, and a dataset size of 3200 corresponds to 160 seconds of human demonstration. We chose these dataset sizes to be

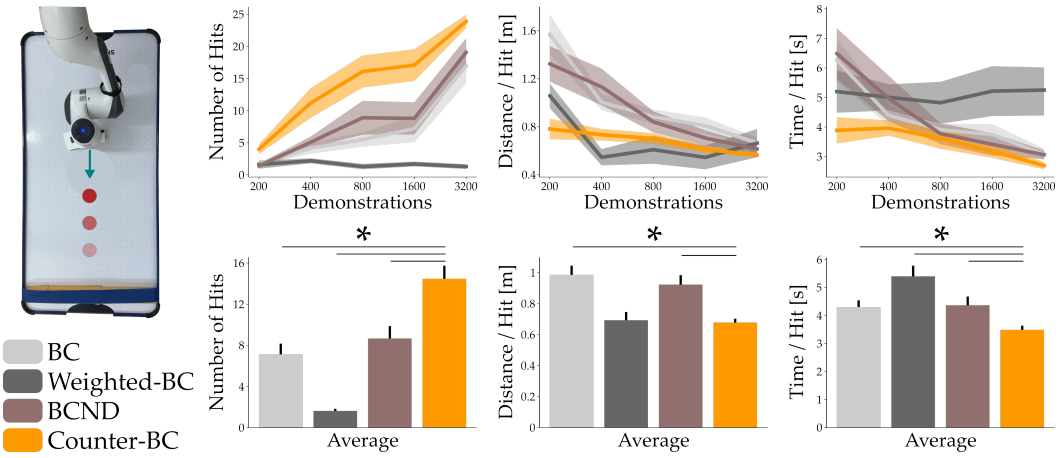

Fig. 7. Results from our air hockey experiment in Section 6. In-person participants provided demonstrations where they teleoperated the robot arm to repeatedly hit a hockey puck (also see Figure 1). The top row plots the performance of the robot's learned policy when trained on an increasing number of state-action pairs (i.e., "demonstrations"); the bottom row shows the average performance across all dataset sizes. *Number of Hits* is the number of times the robot autonomously hit the puck (higher is better). *Distance per Hit* is path length of the robot's end-effector divided by the number of hits (lower is better), and *Time per Hit* is the total interaction time divided by the number of hits (lower is better). Our results suggest that robots which use **Counter-BC** to learn from human demonstrations outperform other offline imitation learning approaches that account for imperfect teaching. Shaded regions and error bars show the standard error, and an $*$ denotes statistical significance ($p < .05$).

roughly consistent with the simulations in Section 5 and Figure 4; in our preliminary testing, 100 state-action pairs was not sufficient for any method to learn a meaningful air hockey policy.

During evaluation we perform 10 end-to-end runs for each dataset size. We emphasize that the robot policies are re-trained from scratch at each run, so the results presented below show performance across 50 separate rounds of training and testing per algorithm. In what follows we describe a single one of these runs. To evaluate the performance with k datapoints, we first sample k state-action pairs from dataset $\mathcal{D}$ and then train all four algorithms with those same datapoints. For each method (**BC**, **Weighted-BC**, **BCND**, **Counter-BC**) we place the hockey puck directly in front of the robot's end-effector and start executing the learned policy.

Dependent Measures. Ideally, the robot will learn to repeatedly hit the puck while moving quickly and efficiently. We therefore count the *Number of Hits*, i.e., the number of times the robot uses its end-effector to push the puck across the table. To assess efficiency, we record the *Distance per Hit*. This corresponds to the distance the robot's end-effector travels during an interaction (in meters) divided by the number of hits; if the robot never hits the puck, then this is just the total distance traveled. Similarly, the *Time per Hit* is the total interaction time (in seconds) divided by the number of hits. As before, if the robot never hits the puck, *Time per Hit* becomes the total interaction time. A proficient robot should repeatedly hit the puck (maximizing *Number of Hits*) while minimizing the distance traveled (*Distance per Hit*) and the time between hits (*Time per Hit*).

Hypotheses. We hypothesized that:

> *Robots that use **Counter-BC** to learn from human demonstrations will extract more proficient control policies than the baselines.*

Results. Our experimental results are summarized in Figure 7. In the top row we plot our dependent measures as a function of the number of demonstrations; the bottom row displays the average outcomes across all 50 trials for each algorithm. As a reminder, for the *Number of Hits* **higher** values indicate better task performance. By contrast, for the *Distance per Hit* and the *Time per Hit* **lower** values mean the robot is playing the game more quickly and efficiently.

To capture overall trends in the learned policies we will analyze the average results. Repeated measures ANOVAs with Greenhouse-Geisser corrections determine that algorithm type has a statistically significant effect on the *Number of Hits* ($F(2.74, 134.24) = 39.8$, $p < .001$), the *Distance per Hit* ($F(2.42, 118.78) = 12.6$, $p < .001$), and the *Time per Hit* ($F(1.89, 92.48) = 8.3$, $p < .001$). To better understand these differences, we next conduct post hoc tests with a Bonferroni adjustment. Here we see that **Counter-BC** averages a higher *Number of Hits* across all demonstrations as compared to **BC** ($p < .001$), **Weighted-BC** ($p < .001$), and **BCND** ($p < .001$). Similarly, the *Time per Hit* for **Counter-BC** is lower than **BC** ($p < .01$), **Weighted-BC** ($p < .001$), and **BCND** ($p < .05$). For the *Distance per Hit* we gather mixed findings: **Counter-BC** is again more performant than **BC** ($p < .001$) and **BCND** ($p < .01$), but comparisons with **Weighted-BC** are not statistically significant ($p = 1.0$). We can explain this result by watching the learned behavior for **Weighted-BC**. *Distance per Hit* reflects the distance the robot traveled during each interaction. In our tests, robots trained with **Weighted-BC** remained close to their initial position, often missing the puck (as reflected by the low *Number of Hits*) or hitting that puck at a low velocity (as reflected by the high *Time per Hit*). Accordingly, while the **Weighted-BC** robots learned concise motions similar to **Counter-BC**, these motions were not effective at completing the overall task.

Discussion. In summary, our results from Figure 7 support our hypothesis and suggest that **Counter-BC** is an effective algorithm for learning from real human data on physical robot arms. Across each axis of evaluation (*Number of Hits*, *Distance per Hit*, and *Time per Hit*), we find that **Counter-BC** outperforms or matches the baselines, and these trends hold regardless of the number of state-action pairs leveraged to train the robot learner.

More generally, our experiments in Section 5 and Section 6 included learning tasks from images, learning tasks with high-dimensional states, and learning tasks on real-world robot arms. The combined results indicate that Counter-BC often learns more proficient policies than state-of-the-art baselines. In particular, we found that Counter-BC is robust to different types of human data with varying levels of noise, and that Counter-BC is also more performant when trained on real human demonstrations. One of the strengths of Counter-BC is its relatively simple implementation (see https://github.com/VT-Collab/Counter-BC), and its theoretical connection to other behavior cloning variants. We believe that Counter-BC is well suited for scenarios where the robot is given a single, aggregated dataset: this dataset could be provided by one human teacher, or the result of multiple different users interacting with the system. We also highlight that Counter-BC can be used in combination with other approaches (e.g., diffusion policies) by simply adding a counterfactual within those approaches and then minimizing the entropy of the learned policy. On the other hand, Counter-BC may struggle with tasks that require multimodal policies; see our Limitations below.

7 CONCLUSION

Existing imitation learning algorithms seek to emulate the "optimal" actions of the human teacher. When the human's teaching is of varying quality, these approaches attempt to separate "optimal" and "noisy" actions, and then train a policy to exactly match these optimal behaviors. By contrast, this work introduced a different perspective. Our core hypothesis was that all of the human's data is based on the underlying task they are trying to convey. Suboptimality inherent to human teaching adds unintended complexity to the dataset, but we can develop learning algorithms to work around

this noise and recover the underlying function. In doing so, the robot is no longer constrained to imitate the human's *exact* datapoints; instead, the robot can now propose and imitate *counterfactual* actions to extrapolate what the human meant. We formalized this concept by deriving Counter-BC, an offline imitation learning algorithm that accounts for imperfect human teachers. We proved that the loss function used to train Counter-BC generalized prior works; put another way, current methods can be viewed as instances of our overarching approach. In practice, by optimizing this generalized loss Counter-BC trains the policy to minimize entropy over actions (i.e., reaching a confident prediction) while ensuring that the actions output by the policy are close to the actions demonstrated by the human (i.e., remaining within the counterfactual set). Our analysis indicated that Counter-BC can extract the desired policy from imperfect data, multiple users, and teachers of varying skill. To support these findings, we tested Counter-BC across several environments with real human teachers, synthetic demonstrations, and standardized datasets. Overall, our theoretical and empirical contributions are a step towards robots that learn what their human teachers actually meant, not just what the human teacher showed.

Limitations. One advantage and disadvantage of Counter-BC is the hyperparameter $\Delta \geq 0$ in Algorithm 1. This hyperparameter specifies the size of the counterfactual set; intuitively, increasing Δ means the robot can extrapolate more from the human's data, but the resulting policy may contain greater deviations from the human's demonstrations. The advantage of Δ is that designers can tune this parameter along a continuous spectrum so that Counter-BC works with expert teachers (low values of Δ) and inexperienced teachers (high values of Δ). The disadvantage is that — if designers choose too large a Δ — Counter-BC may oversimplify the policy and miss important nuances in the task. Within our manuscript we tried to mitigate this potential issue by providing recommended values and by testing with a range of Δ. In future work, we would like to automate the selection of Δ based on the offline dataset of human demonstrations.

A second limitation of Counter-BC is its approach to multimodal human behaviors. For some tasks the learned policy should output multiple, distinct actions. Imagine a mobile robot driving towards an obstacle: ideally, the robot has learned to pass this obstacle both on the right (one mode) and on the left (a second mode). Counter-BC inherently struggles with these multimodal scenarios because it is trained to minimize action entropy within its counterfactual set, thereby penalizing any policy which tries to match more diverse human demonstrations. Instead of learning to pass on both sides of the obstacle, for instance, the policy trained with Counter-BC will likely collapse to just one of these demonstrated modes (e.g., always passing on the right). Our experimental results suggest that Counter-BC can handle some of the action diversity that is inherent to human demonstrations (e.g., our participants in Section 6 teleoperated the robot with different timings, speeds, and movement patterns). However, we do not think Counter-BC is well suited for tasks which require multimodal policies. We plan to explore this limitation in future work.

A more severe form of the multimodal limitation arises when heterogeneous demonstrators have genuinely different goals (rather than different styles for achieving the same goal). Here Counter-BC would likely collapse to one task rather than representing both. Resolving this limitation likely requires first segmenting the dataset by demonstrator intent — for example, conditioning on task — before applying Counter-BC with the relevant state conditioning.

REFERENCES

[1] Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L Thomaz. 2012. Keyframe-based learning from demonstration: Method and evaluation. International Journal of Social Robotics 4 (2012), 343–355.

[2] Mark Beliaev, Andy Shih, Stefano Ermon, Dorsa Sadigh, and Ramtin Pedarsani. 2022. Imitation learning by estimating expertise of demonstrators. In International Conference on Machine Learning. 1732–1748.

[3] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. 2023. Data quality in imitation learning. Advances in Neural Information Processing Systems 36 (2023), 80375–80395.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. 2024. π_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv preprint arXiv:2410.24164 (2024).

[5] Andreea Bobu, Andi Peng, Pulkit Agrawal, Julie A Shah, and Anca D Dragan. 2024. Aligning human and robot representations. In ACM/IEEE International Conference on Human-Robot Interaction. 42–54.

[6] Daniel S Brown, Wonjoon Goo, and Scott Niekum. 2020. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In Conference on Robot Learning. 330–359.

[7] Xingchen Cao, Fan-Ming Luo, Junyin Ye, Tian Xu, Zhilong Zhang, and Yang Yu. 2024. Limited preference aided imitation learning from imperfect demonstrations. In International Conference on Machine Learning.

[8] Annie S Chen, Alec M Lessing, Yuejiang Liu, and Chelsea Finn. 2025. Curating Demonstrations using Online Experience. arXiv preprint arXiv:2503.03707 (2025).

[9] Letian Chen, Rohan Paleja, and Matthew Gombolay. 2021. Learning from suboptimal demonstration via self-supervised reward regression. In Conference on Robot Learning. 1262–1277.

[10] Sirui Chen, Chen Wang, Kaden Nguyen, Li Fei-Fei, and C Karen Liu. 2024. ARCap: Collecting high-quality human demonstrations for robot learning with augmented reality feedback. arXiv preprint arXiv:2410.08464 (2024).

[11] Sonia Chernova and Andrea L Thomaz. 2022. Robot Learning from Human Teachers. Springer Nature.

[12] Yinlong Dai, Robert Ramirez Sanchez, Ryan Jeronimus, Shahabedin Sagheb, Cara M Nunez, Heramb Nemlekar, and Dylan P Losey. 2025. CIVIL: Causal and Intuitive Visual Imitation Learning. arXiv preprint arXiv:2504.17959 (2025).

[13] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. 2022. Implicit behavioral cloning. In Conference on Robot Learning. 158–168.

[14] Kanishk Gandhi, Siddharth Karamcheti, Madeline Liao, and Dorsa Sadigh. 2023. Eliciting compatible demonstrations for multi-human imitation learning. In Conference on Robot Learning. 1981–1991.

[15] Udita Ghosh, Dripta S Raychaudhuri, Jiachen Li, Konstantinos Karydis, and Amit K Roy-Chowdhury. 2024. Robust offline imitation learning from diverse auxiliary data. arXiv preprint arXiv:2410.03626 (2024).

[16] Jake Grigsby and Yanjun Qi. 2021. A closer look at advantage-filtered behavioral cloning in high-noise datasets. arXiv preprint arXiv:2110.04698 (2021).

[17] Soheil Habibian, Antonio Alvarez Valdivia, Laura H Blumenschein, and Dylan P Losey. 2025. A survey of communicating robot learning during human-robot interaction. The International Journal of Robotics Research 44, 4 (2025), 665–698.

[18] Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. 2025. Robot Data Curation with Mutual Information Estimators. arXiv preprint arXiv:2502.08623 (2025).

[19] Matt Jordan and Alexandros G Dimakis. 2020. Exactly computing the local lipschitz constant of relu networks. Advances in Neural Information Processing Systems 33 (2020), 7344–7353.

[20] Liyiming Ke, Sanjiban Choudhury, Matt Barnes, Wen Sun, Gilwoo Lee, and Siddhartha Srinivasa. 2021. Imitation learning as f-divergence minimization. In Workshop on the Algorithmic Foundations of Robotics. 313–329.

[21] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. 2024. DROID: A large-scale in-the-wild robot manipulation dataset. In Robotics: Science and Systems.

[22] Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. 2021. DemoDICE: Offline imitation learning with supplementary imperfect demonstrations. In International Conference on Learning Representations.

[23] W Bradley Knox and Peter Stone. 2008. Tamer: Training an agent manually via evaluative reinforcement. In 2008 7th IEEE international conference on development and learning. IEEE Monterey, CA, 292–297.

[24] Sachit Kuhar, Shuo Cheng, Shivang Chopra, Matthew Bronars, and Danfei Xu. 2023. Learning to discern: Imitating heterogeneous human demonstrations with preference and representation learning. In Conference on Robot Learning. 1437–1449.

[25] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. 2022. When should we prefer offline reinforcement learning over behavioral cloning? arXiv preprint arXiv:2204.05618 (2022).

[26] Michael Laskey, Caleb Chuck, Jonathan Lee, Jeffrey Mahler, Sanjay Krishnan, Kevin Jamieson, Anca Dragan, and Ken Goldberg. 2017. Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations. In 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 358–365.

[27] Ziniu Li, Tian Xu, Zeyu Qin, Yang Yu, and Zhi-Quan Luo. 2023. Imitation learning from imperfection: Theoretical justifications and algorithms. Advances in Neural Information Processing Systems 36 (2023), 18404–18443.

[28] Dylan P Losey, Andrea Bajcsy, Marcia K O'Malley, and Anca D Dragan. 2022. Physical interaction as communication: Learning robot objectives online from human corrections. The International Journal of Robotics Research 41, 1 (2022),

20–44.

[29] Dylan P Losey, Hong Jun Jeon, Mengxi Li, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, Jeannette Bohg, and Dorsa Sadigh. 2022. Learning latent actions to control assistive robots. Autonomous Robots 46, 1 (2022), 115–147.

[30] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. 2021. What matters in learning from offline human demonstrations for robot manipulation. arXiv preprint arXiv:2108.03298 (2021).

[31] Shaunak A Mehta, Yusuf Umut Ciftci, Balamurugan Ramachandran, Somil Bansal, and Dylan P Losey. 2025. Stable-BC: Controlling covariate shift with stable behavior cloning. IEEE Robotics and Automation Letters (2025).

[32] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 627–635.

[33] Maram Sakr, Zhikai Zhang, Benjamin Li, Haomiao Zhang, HF Van der Loos, Dana Kulic, and Elizabeth Croft. 2023. How can everyday users efficiently teach robots by demonstrations? arXiv preprint arXiv:2310.13083 (2023).

[34] Fumihiro Sasaki and Ryota Yamashina. 2020. Behavioral cloning from noisy demonstrations. In International Conference on Learning Representations.

[35] Mariah L Schrum, Erin Hedlund-Botti, Nina Moorman, and Matthew C Gombolay. 2022. Mind meld: Personalized meta-learning for robot-centric imitation learning. In 2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI). IEEE, 157–165.

[36] Aran Sena and Matthew Howard. 2020. Quantifying teaching behavior in robot learning from demonstration. The International Journal of Robotics Research 39, 1 (2020), 54–72.

[37] Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge, and Sidd Srinivasa. 2022. Expert intervention learning: An online framework for robot learning from explicit and implicit human feedback. Autonomous Robots (2022), 1–15.

[38] Zexu Sun, Bowei He, Jinxin Liu, Xu Chen, Chen Ma, and Shuai Zhang. 2023. Offline imitation learning with variational counterfactual reasoning. Advances in Neural Information Processing Systems (2023), 43729–43741.

[39] Voot Tangkaratt, Bo Han, Mohammad Emtiyaz Khan, and Masashi Sugiyama. 2020. Variational imitation learning with diverse-quality demonstrations. In International Conference on Machine Learning. 9407–9417.

[40] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. 2024. Octo: An open-source generalist robot policy. arXiv preprint arXiv:2405.12213 (2024).

[41] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulao, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. 2024. Gymnasium: A standard interface for reinforcement learning environments. arXiv preprint arXiv:2407.17032 (2024).

[42] Antonio Alvarez Valdivia, Soheil Habibian, Carly A Mendenhall, Francesco Fuentes, Ritish Shailly, Dylan P Losey, and Laura H Blumenschein. 2023. Wrapping haptic displays around robot arms to communicate learning. IEEE Transactions on Haptics 16, 1 (2023), 57–72.

[43] Qiang Wang, Robert McCarthy, David Cordova Bulens, Kevin McGuinness, Noel E O'Connor, Francisco Roldan Sanchez, Nico Gürtler, Felix Widmaier, and Stephen J Redmond. 2023. Improving behavioural cloning with positive unlabeled learning. In Conference on Robot Learning. 3851–3869.

[44] Qiang Wang, Robert McCarthy, David Cordova Bulens, Francisco Roldan Sanchez, Kevin McGuinness, Noel E O'Connor, and Stephen J Redmond. 2023. Identifying expert behavior in offline training datasets improves behavioral cloning of robotic manipulation policies. IEEE Robotics and Automation Letters 9, 2 (2023), 1294–1301.

[45] Yunke Wang, Minjing Dong, Yukun Zhao, Bo Du, and Chang Xu. 2023. Imitation learning from purified demonstrations. arXiv preprint arXiv:2310.07143 (2023).

[46] Yunke Wang, Chang Xu, Bo Du, and Honglak Lee. 2021. Learning to weight imperfect demonstrations. In International Conference on Machine Learning. 10961–10970.

[47] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. 2019. Imitation learning from imperfect demonstration. In International Conference on Machine Learning. 6818–6827.

[48] Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. 2022. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In International Conference on Machine Learning. 24725–24742.

[49] Mengjiao Yang, Sergey Levine, and Ofir Nachum. 2021. TRAIL: Near-Optimal Imitation Learning with Suboptimal Data. In International Conference on Learning Representations.

[50] Sheng Yue, Jiani Liu, Xingyuan Hua, Ju Ren, Sen Lin, Junshan Zhang, and Yaoxue Zhang. 2024. How to leverage diverse demonstrations in offline imitation learning. In International Conference on Machine Learning.

[51] Maryam Zare, Parham M Kebria, Abbas Khosravi, and Saeid Nahavandi. 2024. A survey of imitation learning: Algorithms, recent developments, and challenges. IEEE Transactions on Cybernetics (2024).

[52] Songyuan Zhang, Zhangjie Cao, Dorsa Sadigh, and Yanan Sui. 2021. Confidence-aware imitation learning from demonstrations with varying optimality. Advances in Neural Information Processing Systems 34 (2021), 12340–12350.
[53] Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. 2020. Offline learning from demonstrations and unlabeled experience. In Offline Reinforcement Learning Workshop at Neural Information Processing Systems.

8 APPENDIX

Table 1. Counter-BC relative improvement (%) over each baseline across tasks. See Figure 4.

Baseline	Interaction	Cartpole	Car Racing	Robomimic
BC	10%	39%	2,370%	21%
Weighted-BC	144%	129%	1,179%	13%
BC-RNN	8%	51%	276%	15%
BCND	9%	38%	563%	11%