2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802106

Self Supervised Detection of Incorrect Human Demonstrations: A Path Toward Safe Imitation Learning by Robots in the Wild

Noushad Sojib and Momotaz Begum

***Abstract*— A major appeal of learning from demonstrations or imitation learning (IL) in robotics is that it learns a policy directly from lay users. However, Lay users may inadvertently provide erroneous demonstrations that lead to learning of policies that are inaccurate and hence, unsafe for humans and/or robot. This paper makes two contributions in the endeavour of recognizing human errors in demonstrations and thereby helping to learn a safe IL policy. First, we created a dataset – Layman V1.0 – with 15 lay users who provided a total of 1200 demonstrations for three simulated tasks – *Lift, Can* and *Square* in the simulated Robosuite environment – and two real robot tasks with a Sawyer robot, using a custom designed Android app for tele-operation. Second, we propose a framework named Behavior Cloning for Error Detection (BED) to autonomously detect and discard erroneous demonstrations from a demonstration pool. Our method uses a Behavior Cloning method as self-supervised technique and assigns binary weight to each demonstration based on its inconsistencies with the rest of the demonstrations. We show the effectiveness of this framework in detecting incorrect demonstrations in the Layman V1.0 dataset. We further show that state-of-the-art (SOTA) policy learners learns a better policy when bad demonstrations, identified through the proposed framework, are removed from the training pool. Dataset and Codes are available in `https://github.com/AssistiveRoboticsUNH/bed`**

I. INTRODUCTION

We envision IL-enabled robots to serve lay humans in their own homes for a wide range of activities of daily living. In such settings, lay users will have to teach robots – e.g through teleoperation – how to perform a new task. Some of these task demonstrations may get spoiled – partially or fully – with inadvertent errors caused by factors such as a user's unfamiliarity with robots/teleoperation interfaces, fatigue, distractions, etc. For example, it is not unusual for a novice lay user to slam the door while tele-operating a robot to show how to pick a milk-can from the refrigerator, or release an object before reaching the goal and thereby causing it to break. When IL-enabled robots are in the wild and robotics experts are not around to curate the data, the policy learning algorithm needs a mechanism to deal with such erroneous demonstrations. Otherwise, learned policies will not only be inaccurate but also unsafe – causing physical and/or financial harms to end-users. Although IL is increasingly becoming ubiquitous [1], topics relevant to deployment safety – such as the impact of human errors on learned policies – are nascent in the IL literature. A comprehensive study on the type of errors lay users may make while giving demonstrations and the impact of those errors on the policy accuracy and safety is non-existent. The closest group of IL works that deal with a similar issue is policy learning from sub-optimal demonstrations. This

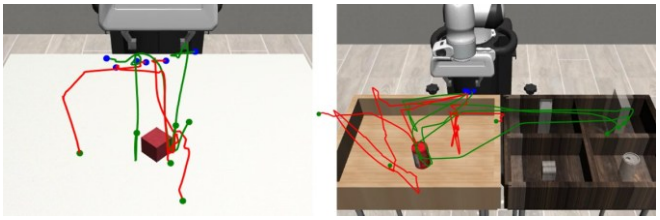

Fig. 1: Errors in human demonstrations: High-quality demonstrations exhibit consistent pattern (green), whereas poor demonstrations do not follow any nominal pattern (red). (Left) The *Lift* task in the Robosuite environment [2]: lift the block up, (Right) The *Can* task: pick up the can and place it in the bottom-right corner

group of works forgoes a long-standing assumption in the IL literature that demonstrations are optimal [3] and attempts to learn a high accuracy policy from demonstrations that are noisy or sub-optimal. However, the way these works model sub-optimality is not a faithful representation of the way errors occur in lay users' demonstrations. For example, adding a fixed number of random actions to a RL-generated expert policy (RL: reinforcement learning) or adding random actions at random states – two contemporary methods of creating sub-optimal demonstrations [2] – is not how a human errs while doing a goal-directed task. The seminal work [2] published in 2021 made the first attempt to create a diagnostic dataset – termed as "Multi-human dataset" – that captured some examples of humans errors in demonstrating a number of tasks in simulated *Robosuite* environment. This dataset primarily considers inefficient task trajectories – e.g. long path length – that eventually succeeded in completing the task as examples of human errors. There is however no coordinated effort to understand the reality of human errors in demonstrations, the way these errors impact policy safety-accuracy, and necessary measures to mitigate this issue. The proposed work bridges that gap through making the following contributions:

First, we have launched an Institutional review board (IRB) approved study to collect human demonstrations from **lay users** for various robotics tasks, both in simulation and in real-world. This paper releases the first dataset from this ongoing study, **Layman V1.0**, that contains 1200 demonstrations by 15 lay users of three simulation tasks – *Can, Lift,* and *Square*, in the Robosuite environment – and two real world tasks by

Authors are with the Department of Computer Science, University of New Hampshire, USA noushad.sojib@unh.edu, Momotaz.Begum@unh.edu

a Sawyer robot – *Drawer Closing* and *Spoon Picking*. An intuitive Android App that we designed for teleoperation to collect these data are also made available to the research community (Section III).

Second, we propose a weighted regression-based self-supervised framework to automatically identify incorrect demonstrations in a training pool. The proposed framework, that we term BED, leverages policy learning as a pretext task to learn poor weights for demonstrations that are inconsistent with the rest of the demonstrations in the training pool. During optimization the weights are forced to take values towards either 1 or 0 that later can be treated as binary value for good and bad respectively. A unique attribute of the proposed BED is that it assigns a weight to an entire demonstration, instead of state-action pairs in a demonstrations pool. (Section IV).

Third, we demonstrate the performance of BED in detecting erroneous demonstrations in the **Layman V1.0** dataset. We further demonstrate that a SOTA IL algorithm learn a better policy when erroneous demonstration are masked using the proposed BED framework (Section V).

II. Related Works

IL literature traditionally followed an algorithmically convenient assumption that demonstrations are always optimal [3]. Considering demonstrations to be sub-optimal is a very recent trend in IL research where the primary focus is to learn a policy with high task-accuracy despite the sub-optimality in the training data. We categorize the existing IL literature that deals with sub-optimal demonstrations into two groups based on how the sub-optimal demonstrations are created. **The first group** creates sub-optimality synthetically – such as, taking a random action with a small probability at arbitrary states. A simulator is needed for such a synthetic generation of sub-optimal demonstrations and typically, the sub-optimality is introduced to a trained RL policy, instead of a real human-demonstrations [4], [5], [6], [7], [8]. By nature, these synthetic errors do not capture the nuances of human errors in demonstrations and therefore trivialize the problem. Access to a simulator allows generation of an abundant amount of training data which can often help with compensating for the loss of state-space-coverage caused by incorrect/sub-optimal demonstrations [9], [10], [11], [12], [13]. However, availability of a simulator for every task to be taught in the wild is a too restrictive assumption for deployment of IL-agents. The most prevalent approach in this group of research for learning a high-accuracy policy from sub-optimal demonstrations is weighing the demonstration based on its correctness. A vast majority of work requires a labeled/ranked dataset [14], [15], [16], [17], [12] to train a machine learning model for generating poor weights or confidence scores for incorrect demonstrations [17], [12], [18], [19], [20]. This is unrealistic – especially, in the context of IL in the wild – due to the burden imposed on the user for manual ranking/labeling of incorrect demonstrations. From this group, BCND [20] is the closest to our work since it does not require any labeled data to generate weight. BCND [20] uses an already learned Behavioral Cloning (BC) policy to generate weights for state-action pairs in a demonstration pool. Through multiple iterations, subsequent BC policies converge toward the mode actions observed in the demonstrations. Although intuitive, this strategy of convergence to the mode actions will face difficulty when dealing with human demonstrations. For example, if one state is visited multiple times due to the incompetency of a user in demonstrating a task, the assumption – that the number of correct action at a state outnumber the incorrect action at a given state – becomes invalid, causing the algorithm to fail. We experimentally demonstrate such failures in Section V. Note that BCND in [20] has been tested only with synthetically generated sub-optimal data for a few simulated Mujoco tasks.

TABLE I: A summary of the datasets typically leveraged by IL research.

Data Source	Tasks	Demo by Human?
Robomimic [21], [7]	Lift,Can,Square,Tool Hang	Yes
MuJoCo [22], [23], [5], [6], [24], [16]	Ant, HalfCheetah, Walker, Humanoid,Reacher,Swimmer, Beam Rider, Seacost	No
D4RL[5], [6]	Door, Hammar, Relocate	No
Minigrid [4]	DoorKey, FourRoom	No
Atari[24]	Pong, Space Invader	No
Real Robot [23], [16]	Custom tasks	Yes

The second group of work make a rare attempt to understand the reality of errors in human demonstrations [2], [21]. The seminal work in [2] introduces two datasets – *Multi human* and *Can-Pair* – that captures some human errors in demonstrations of a number of tasks in the simulated *Robosuite* environment. However, the *Multi human* dataset only captures human demonstrations with inefficient trajectories that are eventually successful in completing the tasks. The *Can-Pair* dataset simply mixes demonstrations from a target task with those from a non-target task. Despite this lack of diversity in human-generated sub-optimality, the policy accuracy of standard behavior cloning algorithm dropped when tested with these two datasets [2] , indicating that human errors are fundamentally different from synthetically introduced sub-optimality. The only other work in this group is ILEAD [21] which leverages the expertise level of demonstrators to learn a better policy. However, it does not address the issue that the expertise level of lay users changes over time. Table I summarizes the datasets that the contemporary IL research community relies on for policy learning from sub-optimal demonstrations. There is only one dataset that includes human demonstrations. Overall, understanding human errors in demonstrations and their impact on the learned policy is a heavily under-explored area in imitation learning. This paper attempts to bridge that gap.

III. Understanding Errors in Human Demonstrations: Layman V1.0 Dataset

The famous quote of Tolstoy that '*All happy families are alike; each unhappy family is unhappy in its own way*' could be an uncanny characterization of human errors in demonstrations. Errors in human demonstrations manifest in diverse ways, while correct demonstrations tends to exhibit consistency (Fig. 1). Not all errors have negative consequences on the quality of the learned policy. For example, the sub-optimal demonstrations in the *Multi-human* dataset actually helped BC-RNN to learn a robust policy than the baseline [2] – primarily because of the higher state-space coverage achieved through the exploration performed by the inefficient trajectories. To understand the nuances of human errors during demonstrations, we have launched an IRB approved study where lay users are asked to provide task demonstrations through teleoperating robots in simulated and real environments. This paper releases the first version of this dataset: **Layman V 1.0**.

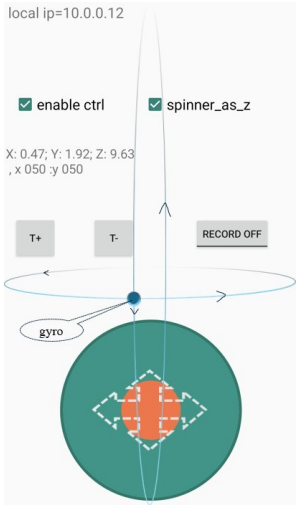

Fig. 2: SixDOF Android App interface. A robot end-effector is controlled through the sliding button and the physical rotation of the phone

User Demographics: Fifteen lay users participated in this study. None of the participants had any previous exposure to robotics. However, participants had varying level of familiarity with computer games: '1' indicates no familiarity at all and '10' indicates proficiency in playing various computer games using keyboard/mouse/game-controller. Among the participants, 7 were male and 8 were female; 4 participants had self-reported expertise level between 1-3, 5 participants reported between 4-6, and 6 other reported between 7-10; 8 participants were in the age group 18-22 years and 7 were in the age group 23-30 years.

Teleoperation interface: We designed an Android app, that we term SixDOF, for teleoperation of a Sawyer robot and the robot in the simulated Robosuite environment. Unlike similar apps [2], SixDOF is openly available for the research community (Github Link). Fig. 2 shows the SixDOF interface. The App uses the gyroscope of an Android phone for an easy and intuitive control of the robot.

Data: The **Layman V1.0** dataset consists of a total 1200 demonstrations for three tasks in simulation – *Lift, Can, Square* in Robosuite – and two tasks in real world by a Sawyer robot – *Drawer Closing* and *Spoon Picking*. The *Drawer Closing* task involves closing the top drawer of a multi-drawer shelf, while the *Spoon Picking* task entails picking up a spoon from a holder and dropping it into a coffee cup. An expert demonstration of both tasks is shown in Figure 5. For the simulated tasks, 12 participants were involved and each provided 25 demonstrations for each task. For the real robot tasks, 5 participants were involved and each provided 20 demonstrations for each task. We recorded observations from two cameras (a wrist camera and a front-view camera), end-effector position, joint positions, joint velocities, and gripper status. Observation data can be considered as states while the delta changes in the end-effector position can be considered as actions. All demonstrations were recorded at a frequency of 20Hz. The dataset contains both naturally occuring mistakes (Type-1 and Type-2 as defined later) and intentionally making mistakes (Type-3 as defined later).

Summary of Human errors: Analysis of the **Layman V1.0** dataset reveals two primary type of errors in human demonstrations. It is important to note that inclusion of more participants with diverse demographics and diverse tasks requiring finer manipulation skills will inevitably reveal other type of human errors that are not currently present in **Layman V1.0**.

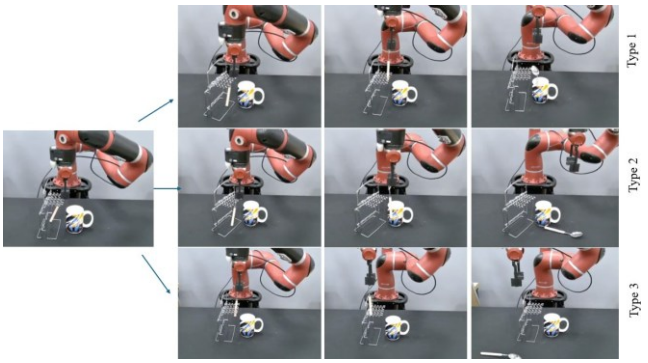

Fig. 3: Errors in human demonstrations for the *Spoon picking* task. Type 1: a user displaces the holder while picking up the spoon but managed to put the spoon in the cup. Type 2: a user displaced the holder and failed to put the spoon in the cup. Type 3: the expert picked up the spoon efficiently but intentionally threw it on the table.

- **Type 1**: A user struggles to control the robot using the interface, primarily due to unfamiliarity, resulting in the robot visiting states that are not relevant to the task goal. However, the user eventually completes the task. Type 1 errors may result in trajectories that are fundamentally different from an expert demonstration of the task. Note that, state visitation may differ among different runs of the same task, even for the same user.
- **Type 2**: This is similar to Type 1 but the user is unsuccessful in completing the task at the end. Type 2 errors also include trajectories that are similar to an expert trajectory for the most part but failed at the end to complete the task.

The *Can-Pair* dataset in [2] reports a type of error for the *Can* task where a human intentionally performs a series of actions that lead to a task failure – for instance, throwing away the can instead of placing it at the designated location. Although this type of intentional error is not common when lay users train

robots, for the sake of comparison with the *Can-Pair dataset*, we create data spoiled with this type of error and term it **Type 3** error. The first author of this paper, considered as an expert, created Type 3 error dataset for the *Can, Lift,* and *Square* tasks. For the Sawyer dataset lay users also contributed to the Type 3 errors. The same expert also generated **Baseline** demonstrations which complete each task efficiently hence, in the shortest time. **Type 3** and **Baseline** demonstrations, despite not being from lay users, are available in the **Layman V1.0** to facilitate comparison by other similar research. Fig. 3 demonstrates three type of errors corresponding to the *Spoon picking* task.

IV. The Proposed Framework: Behavior Cloning for Error Discovery (BED)

A. Preliminaries

We consider a standard Markov Decision Process (MDP) [25] formalized as a tuple $\{S, A, P, R, d_0, T\}$ where S is the set of states, A is the set of possible actions, $P : S \times A \times S \rightarrow [0, 1]$ is the transition probability, $R : S \times A \rightarrow [0, 1]$ is the reward function, $d_0 : S \rightarrow [0, 1]$ is the initial state distribution and T is the episode horizon. However, the reward function R is unknown in the context of IL and is typically retrieved from human demonstration D.

The proposed BED framework leverages policy learning – specifically, behavior cloning (BC) – as a pretext task to identify errors in human demonstrations. The generalized BC objective function is [6]:

$$\arg\max_{\theta} \mathsf{E}_{(s,a)\sim D} [\log \pi_\theta(a|s) \cdot f(s, a)] \tag{1}$$

Here π_θ is the parameterized BC policy with parameters θ and $f : S \times A \rightarrow [0, 1]$ is an arbitrary weight function. Choices of f generate objective functions for a range of algorithms that deals with suboptimal demonstrations.

B. The BED Framework

Given a set of demonstrations of a task, the goal of the the proposed BED framework is to separate the correct demonstrations from the erroneous/incorrect ones in a self-supervised manner. Such incorrect demonstrations can be spoiled with any type of errors (Type 1, 2, or 3). The BED framework uses policy learning as a pretext task to achieve this. The intuition behind error discovery through the BED framework is based on the following two assumptions:

Assumption 1. *The number of correct demonstrations outnumbers the erroneous demonstrations.*

Assumption 2. *Correct demonstrations exhibit greater consistency among themselves, whereas incorrect demonstrations exhibit variability*.

Both are reasonable assumptions in the context of task learning from a human demonstrator. Learning from an unlabeled dataset that contains both successful and unsuccessful demonstrations is substantially limited without Assumption 1. According to both assumptions, in an unlabeled dataset, the largest consistency group represents the correct demonstrations. Leveraging this fact, the proposed BED defines an optimization problem where the objective function captures the (in)consistency of a demonstration with respect to the rest in the training pool and thereby assigning a poor weight to inconsistent, hence erroneous, demonstrations. In its basic form in equation (2), the BED objective function penalizes action inconsistency – i.e., poorly weighing demonstrations where the evolution of actions is different from the rest in the training pool.

$$\underset{\theta, w}{\operatorname{argmin}} \sum_{i=1}^{|D|} w_i \frac{1}{|D_i|} \sum_{(s,a)\sim D_i} (\pi_\theta(s) - a)^2 \tag{2}$$

$$\sum_{j=1}^{|D|} w_j = m \cdot |D|$$
$$0 \le w_j \le 1$$

Here w_i represents the weights assigned to the i-th demonstration, $|D|$ denotes the total number of demonstrations, and m is a hyperparameter denoting the percentage of demonstrations in the training pool that we hypothesize to be correct. (we present empirical analysis on the sensitivity of BED on m in Section V). Also, π_θ is our pretext task that can be any policy that takes an state s and predict an action a. The constraints ensures that the sum of the weights equals to our hypothesized number of good demonstrations and that weights are in between 0 and 1. Note that the objective in (2) takes inspiration from the generalized BC objective function in 1 and leverages the fact that maximizing log-likelihood is the dual of minimizing the MSE [26], [27]. An important difference of (2) from 1 is that it weighs each demonstration, instead of each state-action pair. We can convert the constrained optimization problem 2 into a soft unconstrained **BED loss function** as follows. Here the third constraint is omitted by clipping w between 0 and 1 during each update.

$$\begin{aligned} L(D, m) = &\sum_{i\in|D|} w_i \frac{1}{|D_i|} \sum_{(s,a)\sim D_i} (\pi_\theta(s) - a)^2 \\ &+ k \left(m\,|D| - \sum_{j=1}^{|D|} w_j\right)^2 \end{aligned} \tag{3}$$

Here, k is a soft constrained multiplier; higher k results in faster weight learning. In practice choice of k does not affect weight learning.

It is possible to incorporate more terms in the BED loss function in (3) to account for inconsistencies other than that in action evolution. For example, both Type 1 and Type 2 errors involve state evolution that is inconsistent with the expert (path consistency); Type 2 errors further involves inconsistency with respect to reaching a desired goal state (goal consistency).

Accordingly, we can define a generalized BED loss term

that includes action, state, and goal consistency.

$$\mathsf{L}(D, m) = c \cdot \sum_{i \in |D|} w_i \cdot \frac{1}{|D_i|} \sum_{(s,a) \sim D_i} (\pi_\theta(s) - a)^2 + h \cdot \sum_{i \in |D|} w_i \cdot \|(\mathbf{G}, \mathbf{g_i})\| + q \cdot \sum_{i \in |D|} w_i \cdot \|(\mathbf{Z}, \zeta_\mathbf{i})\| + k \cdot \Big(m \cdot |D| - \sum_{j=1}^{|D|} w_i\Big)^2 \quad (4)$$

Here, c, h, and q refer to the importance of action consistency, goal consistency, and path consistency, respectively. However, we do not have any knowledge about the goal G except that, the terminal observations of all correct demonstrations should represent the goal. Accordingly, we propose to estimate the goal in the latent space through equation (5) for each demo. Finally, equation (6) helps us to estimate the global goal G.

$$g_i = f_{latent}(state \in \{|D_i|\}) \quad (5)$$

$$G = \mathsf{E}[goal] = \sum_{i \in |D|} \left(\frac{w_i}{sum(w)}\right) \cdot g_i \quad (6)$$

Similarly, we do not have any knowledge of a nominal path Z from which deviation should be punished as an 'inconsistent' path. Rather, we estimate such a path using equation (7) and (8).

$$\zeta_i = f_{latent}(states \in D_i) \quad (7)$$

$$Z = \mathsf{E}[path] = \sum_{i \in |D|} \left(\frac{w_i}{sum(w)}\right) \cdot \zeta_i \quad (8)$$

Note that the estimation of the latent goal and the latent path are a weighted sum of the latent goal and latent path, respectively. The pretext policy requires to have a latent encoder f_{latent} which we can extract from any intermediate layer of a policy network. We can use this latent layer to generate the latent goal and latent path. As the length of the path ζ_i can vary between demonstrations, we use linear interpolation to make the length same. When the path consistency term is consider in (4), the goal consistency term is redundant since the goal is inherently included as the last step of the path. Accordingly, we can set h=0.

C. Training BED

For visual BC it only practical to train model using GPU. As the BED loss iterates over all demonstrations, it takes considerable amount of GPU memory before it can update the model parameter θ, w. To mitigate this resource issue, we devised a novel idea of mini-batching over demonstrations, instead of state-action pairs. Each mini-batch contains $n << |D|$ demonstrations and the weights w corresponding to those demonstrations are updated. The joint training of θ and w is described in Algorithm 1. Figure 4 depicts the idea of mini-batching over demonstrations in the BED framework. It is important to note that the BED loss learns 'a policy' as a pretext task but the BED training is designed to optimize

Algorithm 1 BED Training

Input Mixed Dataset D, Percent of demos to keep m
Output Binary Mask $w_{|D|}$

Randomly initialize policy θ and $w_i = 0.5 \forall i \in |D|$
Estimate G using (6) and Z using (8)
di = demo indices $[1, 2, ..., |D|]$, b=batch_size
while epoch ≤ epochs or sum(round(w)) $\neq m \cdot |D|$ **do**
  for $i = 1$ to $\lceil |D|/\text{batch\_size} \rceil$ **do**
    $bi = di[i \cdot \text{b} : i \cdot \text{b} + \text{b}]$
    calculate loss using equation (4)
    $w_{\text{clipped}} = \max(0, \min(w, 1))$
    update $w[bi]$ and θ as standard gradient update.
  end for
  Update G using (6) and Z using (8)
end while
return round(w)

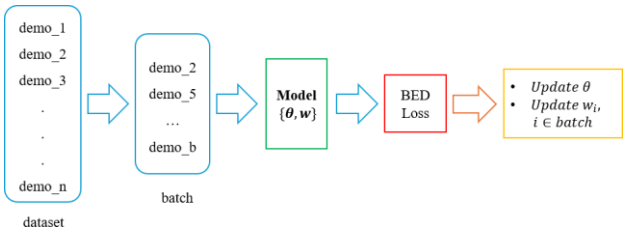

Fig. 4: The BED pipeline: a mini-batch of demonstrations, not state-action pairs, is used to learn weights that discriminate correct demonstrations from incorrect ones

different type of inconsistencies, not the policy. We use the weights w generated through BED as a binary mask that can filter out incorrect demonstrations before policy learning happens by any model architecture.

V. Experiments and Results

We conducted a set of experiments to shed light on the following: (1) The impact of human errors captured in **Layman V1.0** on the policy accuracy of a SOTA policy learning algorithms namely, BC-RNN [2], (2) The performance of BED in detecting incorrect demonstrations in **Layman V1.0** (3) The sensitivity of BED's performance to the choice of the hyperparameter m (4) Comparison with BCND when incorrect demonstrations are masked using the BED framework. Additionally, we also experimented with three MuJoco tasks to facilitate a direct comparison with BCND [22] which reported performance only with MuJoco tasks.

As the pretext policy for BED we used a BC policy architecture from [2], we also used the BC-RNN from robomimic. Both of them used default hyperparameters.

The primary performance metric for BED is the number of False Positive (FP: incorrect demonstrations reported as correct through assigning a weight w=1) and False Negative (FN: correct demonstrations are reported as incorrect through assigning a weight w=0). The performance goal of BED is to keep FP and FN as low as possible.

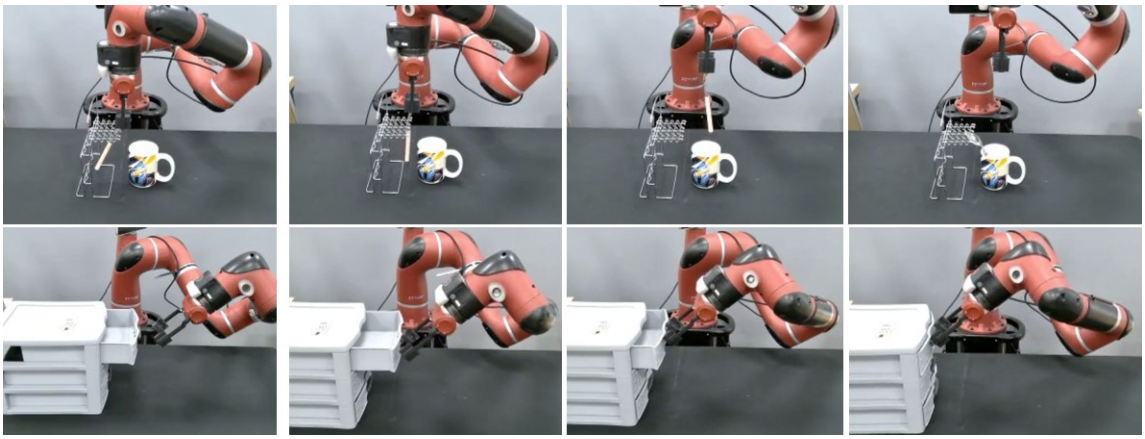

Fig. 5: *Spoon picking* tasks (top row) involves picking up a spoon from a spoon holder and placing it inside a coffee cup. The *Drawer closing* task (bottom row) involves closing an open drawer

A. Mixed dataset preparation for experiments

Based on complexity of the tasks, each task has different N number of demonstrations (Lift=100, Can=150, Square=100, Spoon=100, Drawer=70). However, percentage of correct and incorrect data are consistent in all the tasks. For each task we used $N = N_c + N_i$ demonstrations, where N_c and N_i are the number of correct and incorrect demonstrations, respectively. Baseline is when 100% ($N = N_c + 0$) are correct. Incorrect demonstrations are spoiled with Type 1, 2, or 3 errors to prepare three datasets namely, Type 1 dataset, Type 2 dataset, and Type 3 dataset. 20% spolied means 20% data are replaced with incorrect data. For 20% spoiled data, Lift, Square and Spoon has 80 expert demos and 20 incorrect demos, Can task has 120 expert demos and 30 incorrect demos, Drawer task has 56 expert and 14 incorrect demos. For all the tasks except square first author provided N expert demonstrations. For the square task, we borrowed N expert demonstration from Robomimic dataset [2].

B. Experiment 1: Impact of human errors on policy accuracy

We evaluate BC-RNN policy accuracy – i.e., mean success rate over 3 means where each mean was calculated over 50 rollouts using different seeds. Results are reported in Table II. Here 'Baseline' indicates policy accuracy with the **Baseline** dataset (see Section III). The accuracy drops for all tasks when the policy is learned from demonstrations spoiled with human errors. It is important to note that Type 1 and Type 2 errors (genuine errors made by lay users) have stronger negative impact than the Type 3 error (intentional errors made by an expert). In general, the real robot tasks suffer the most due to incorrect demonstrations. These results affirm the need for separating erroneous human demonstrations before policy learning.

TABLE II: Impact of human errors in demonstrations on BC-RNN policy. Baseline contains no incorrect demo while other types replaced 20% with incorrect demos.

Task	Baseline	Type 1	Type 2	Type 3
Lift	0.99	0.73	0.71	0.78
Can	0.69	0.60	0.59	0.64
Square	0.57	0.25	0.23	0.45
Drawer	1.0	0.2	0.0	1.0
Spoon	1.0	0.0	0.2	0.1

C. Performance of BED in detecting erroneous demonstrations

The proposed BED framework is applied to 20% incorrect data of all tasks with the following hyperparameter settings. $m = 0.8$, $c = 20$, $h = 0$, $q = 5$, $k = 1$. Table III reports the results in terms of FP rates. The proposed BED shows low FP when dealing with Type 1 and Type 2 errors. Type 3 error identification, however, often exhibits a high FP rate. This is because Type 3 errors include demonstrations from experts that share certain commonalities with the correct demonstrations (e.g., for the can task, they may be pursuing a different goal but are efficient in maneuvering).

TABLE III: False Positive rate of BED in detecting errors in demonstrations. Here, 1/20 means 1 out of the 20 incorrect demonstrations were wrongfully assigned a weight of w=1.

Task	Type 1	Type 2	Type 3
Lift	1/20	1/20	0/20
Can	2/30	4/30	13/30
Square	3/20	7/20	4/20
Drawer	2/14	2/14	3/14
Spoon	0/20	1/20	7/20

We noticed that running the same experiment without path consistency results in similar performance for simulation but lower performance for real robot in detecting Type-1 error. Since the goal/task-completion is not well-defined for the two real robot tasks, meaning that some lay users did not stop recording immediately after completing the task, ends up in non-consistent last (goal) time-step. Because of this, here we did not use goal consistency loss ($h = 0$). BED performance for all real robot tasks are slightly higher than that for all simulation tasks, presumably because inconsistencies in real

world are far more prominent – hence, recognizable – than those in the simulation.

D. Sensitivity of BED performance to the choice of the hyperparameter m

Given a training pool, the BED essentially identifies percent of correct demonstrations as defined by m, making the hyperparameter m an important design choice. To understand the dependency of BED's performance on the choice of m, we conducted a series of experiments with the Can task with different percent of mixed dataset (10%, 20%, 30% , 40%) of Type-2. The total number of demonstrations remained 150 for all experiments. Table IV reports FP rate. The results show that irrespective of the choice of m and the actual number of incorrect demonstrations in the training pool, the FP rate remains close to zero. However, there is an opportunity cost associated with the choice of m, as it wants to assign a fixed number of demos to weight 1. If actual bad demos are less than $N - m \cdot |D|$ it detect some other good demos as bad.

TABLE IV: The sensitivity (FP rate) of BED to the hyper-parameter m (Can task, N_i: the actual number of incorrect demonstrations, $\bar{N}_i$: the number of incorrect demonstrations BED will attempt to find based on the choice of m)

m ($\bar{N}_i$)	N_i			
	15	30	45	60
0.8 (30)	1	3	1	0
0.6 (60)	0	0	0	5

So, we run another experiment without having this constraint (setting $m = 0$ while keeping $q > 0$). Table V shows the results. It suggests that, without any assumption of how many are correct, using action and path consistency, BED can detect good and bad demonstration with a very good accuracy while not mislabelling any good data.

TABLE V: BED on various mixture datasets (Can Type-2) ranging from 10% to 40%. (N_i: the actual number of incorrect demonstrations

N_i	15	30	45	60
FP	5	0	1	0
FN	2	3	2	3

It raises the question of whether we can completely remove the m from BED loss, the answer is keeping m gives us a generalized version of the loss that gives flexibility to select a certain number of demonstration as good or bad.

For example, in the following subsection with MuJoco tasks where we did not use goal and path consistency ($h = q = 0$) because of the nature of the dataset, without the m term the policy optimization gives $w_i = 0$ for all w because it gives the most minimized loss. On the otherhand keeping m forces sum of the weights equal to $m * |D|$ hence minimizing the objective function will assign $w = 1$ to the consistent demos and $w = 0$ to the inconsistent demos as they tends to contribute higher loss in the loss function.

E. The role of BED in improving policy accuracy

We evaluate the policy accuracy of BC-RNN and BCND on Type 2 datasets for all five tasks. Table VI reports the result. There is significant increase in BC-RNN policy accuracy when errors are masked using BED. Note that BCND is designed to learn from sub-optimal data. However, BCND performs quite poorly as compared to BC-RNN, both in masked and unmasked cases. The reason is that BCND is based on BC that does not benefit from history and different types of policy like Gaussian Mixture Model as used by BC-RNN. This indicates the incapability of algorithms that learns from sub-optimal data to deal with sub-optimality triggered by human errors. Note that no hyperparameter tuning was performed for BCND due to its extensive time requirement especially for visual imitation learning.

TABLE VI: Policy accuracy on Type-2 dataset.

Task	BC-RNN	BCND	**BED Masked-BC-RNN**
Lift	0.71	0.22	0.90
Can	0.59	0.0	0.67
Square	0.23	0.0	0.52
Drawer	0.0	0.0	1.0
Spoon	0.2	0.2	1.0

F. Other Experiment: Comparison with BCND in Mujoco

BCND is the most closely related algorithm to the proposed BED in a sense that it operates without requiring any labeling, instead it relies on only a few simple assumptions. However, BCND has only been tested with tasks in Mujoco environment [22]. In order to make a direct comparison with BCND, we also ran experiments with three Mujoco tasks using action consistency loss only ($m = 0.8, k = 10, h = q = 0$). Table VII reports the results. Both BC and BCND adopt a Gaussian policy architecture, featuring two MLP layers as an intermediate layer and two additional MLP layers for calculating the mean and variance. Each task consists of 25 demonstrations. In case of Baseline (column 2), all 25 are expert data. Accuracy with BC (column 3) and BCND (column 4) are generated with 20% incorrect (20 correct and 5 incorrect) demonstrations where incorrect data are created by introducing random actions at arbitrary states. Finally, column 5 shows accuracy with BC after the 5 incorrect demonstrations are masked using BED. Results show that masking incorrect/sub-optimal data actually increases policy accuracy, as compared to learning from those incorrect data.

TABLE VII: MuJoCo tasks, mean reward over three mean success, each mean calculated over 50 rollouts with different seeds.

	Baseline	BC	BCND	**BED Masked BC**
Ant-v3	4810.11 ±67.82	4435.16 ±29.16	2414.35 ±1145.73	4752.81 ±55.09
HalfCheetah-v3	4070.67 ±53.11	3812.50 ±211.72	2946.26 ±56.34	4098.58 ±11.25
Walker2d-v3	5452.36 ±74.04	2445.95 ±55.49	422.55 ±45.08	5159.09 ±517.57

VI. CONCLUSIONS

This paper makes two contributions toward understanding the impact of human errors on IL policy. First, it reports a first-of-its-kind dataset capturing the nuances of lay user errors in demonstrating robots, both in simulation and real world. Second, it proposes a novel framework to identify demonstrations spoiled with various type of human errors in a self-supervised manner. The proposed framework uses policy learning as a pretext task and hence does not require any labeling of the data. Extensive experiments demonstrate the performance of the proposed method in simulation and real world. The proposed BED can be used as a wrapper with any policy learning algorithm to identify and block incorrect demonstrations from reaching the policy learner.

ACKNOWLEDGMENT

This was supported in part by the National Science Foundation IIS 1830597 and OIA 2218063

REFERENCES

[1] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[2] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *Conference on Robot Learning (CoRL)*, 2021.

[3] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[4] T. Zhao, W. Yu, S. Wang, L. Wang, X. Zhang, Y. Chen, Y. Liu, W. Cheng, and H. Chen, "Skill disentanglement for imitation learning from suboptimal demonstrations," *arXiv preprint arXiv:2306.07919*, 2023.

[5] G.-H. Kim, S. Seo, J. Lee, W. Jeon, H. Hwang, H. Yang, and K.-E. Kim, "Demodice: Offline imitation learning with supplementary imperfect demonstrations," in *International Conference on Learning Representations*, 2021.

[6] H. Xu, X. Zhan, H. Yin, and H. Qin, "Discriminator-weighted offline imitation learning from suboptimal demonstrations," in *International Conference on Machine Learning*. PMLR, 2022, pp. 24 725–24 742.

[7] J. Hejna, J. Gao, and D. Sadigh, "Distance weighted supervised learning for offline interaction data," *arXiv preprint arXiv:2304.13774*, 2023.

[8] M. Yang, S. Levine, and O. Nachum, "Trail: Near-optimal imitation learning with suboptimal data," *arXiv preprint arXiv:2110.14770*, 2021.

[9] B. Burchfiel, C. Tomasi, and R. Parr, "Distance minimization for reward learning from scored trajectories," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[10] V. Tangkaratt, B. Han, M. E. Khan, and M. Sugiyama, "Variational imitation learning with diverse-quality demonstrations," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 9407–9417.

[11] D. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *International conference on machine learning*. PMLR, 2019, pp. 783–792.

[12] Y. Wang, C. Xu, B. Du, and H. Lee, "Learning to weight imperfect demonstrations," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 961–10 970.

[13] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse reinforcement learning from failure," 2016.

[14] S. Choi, K. Lee, and S. Oh, "Robust learning from demonstrations with mixed qualities using leveraged gaussian processes," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 564–576, 2019.

[15] B. Hertel and S. R. Ahmadzadeh, "Learning from successful and failed demonstrations via optimization," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7807–7812.

[16] S. Zhang, Z. Cao, D. Sadigh, and Y. Sui, "Confidence-aware imitation learning from demonstrations with varying optimality," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[17] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama, "Imitation learning from imperfect demonstration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6818–6827.

[18] M. Hussein, B. Crowe, M. Clark-Turner, P. Gesel, M. Petrik, and M. Begum, "Robust behavior cloning with adversarial demonstration detection," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7858–7864.

[19] M. Hussein and M. Begum, "Detecting incorrect visual demonstrations for improved policy learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 1817–1827.

[20] F. Sasaki and R. Yamashina, "Behavioral cloning from noisy demonstrations," in *International Conference on Learning Representations*, 2020.

[21] M. Beliaev, A. Shih, S. Ermon, D. Sadigh, and R. Pedarsani, "Imitation learning by estimating expertise of demonstrators," in *International Conference on Machine Learning*. PMLR, 2022, pp. 1732–1748.

[22] F. Sasaki and R. Yamashina, "Behavioral cloning from noisy demonstrations," in *International Conference on Learning Representations*, 2021.

[23] Z. Cao and D. Sadigh, "Learning from imperfect demonstrations from agents with varying dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5231–5238, 2021.

[24] D. S. Brown, W. Goo, and S. Niekum, "Better-than-demonstrator imitation learning via automatically-ranked demonstrations," in *Conference on robot learning*. PMLR, 2020, pp. 330–359.

[25] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[26] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[27] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*. MIT press Cambridge, MA, USA, 2017, vol. 1.