Robust Imitation of a Few Demonstrations with a Backwards Model

Anonymous Author(s)
Affiliation
Address
email

Abstract

Behavior cloning of expert demonstrations can speed up learning optimal policies in a more sample-efficient way over reinforcement learning. However, the policy cannot extrapolate well to unseen states outside of the demonstration data, creating covariate shift (agent drifting away from demonstrations) and compounding errors. In this work, we tackle this issue by extending the region of attraction around the demonstrations so that the agent can learn how to get back onto the demonstrated trajectories if it veers off-course. We train a generative backwards dynamics model and generate short imagined trajectories from states in the demonstrations. By imitating both demonstrations and these model rollouts, the agent learns both the demonstrated paths and how to get back on to these paths. With optimal or near-optimal demonstrations, the learned policy will be both optimal and robust to deviations, with a wider region of attraction. On continuous control domains, we evaluate the robustness when starting from different initial states unseen in the demonstration data. While both our method and other imitation learning baselines can successfully solve the tasks for initial states in the training distribution, our method exhibits considerably more robustness to different initial states.

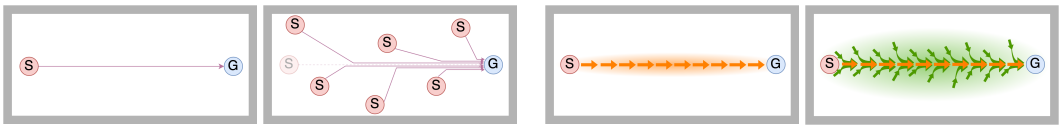
1 Introduction

While reinforcement learning (RL) has shown remarkable success in many challenging domains [18, 35, 33], tasks with sparse rewards and long horizons still remain extremely difficult to solve. In such tasks, a positive reward is only encountered after the RL agent reaches the goal after a long sequence of actions, meaning that it cannot learn any useful signals until this occurs (typically the agent must reach the goal several times to learn reliably as well). Furthermore, the learning signal decreases exponentially with the horizon, which combined with slow gradient-based updates, can lead to catastrophic forgetting even after the agent learns to reach the goal.

Expert demonstrations can help RL agents solve difficult tasks [27, 39, 19]. These demonstrations can be used in the supervised setting where the agent imitates the expert's behavior, termed imitation learning (IL). However, naively cloning (BC) of the expert's trajectories suffers from covariate shift: the agent's policy drifts away from the expert's, which leads to compounding errors due to RL's sequential nature. Furthermore, the distribution of states given in the demonstrations often has low-dimensional support with respect to the entire state space. As such, the agent cannot extrapolate correctly when outside of the demonstration data. Many approaches to solve this issue have been proposed [30, 36, 13], but such approaches require an interactive expert to query the correct actions.

Another way to use demonstrations is to combine imitation learning with reinforcement learning in a form of learning from demonstrations (LfD) [32, 11, 7]. In this case, demonstrations do not simply act as supervised labels and can guide the agent's exploration, and also act as augmentations to good data samples. These LfD approaches either use demonstrations to pretrain the policy [32, 7], use an

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) Training (b) Evaluation (c) BC (d) BMIL

Figure 1: **(a)**, **(b)**: Robustness: The policy is trained on specified set of initial start and goal states and is evaluated at different start states. **(c)**, **(d)**: BMIL uses generated reverse-time rollouts from a backwards model (green arrows originating from the demonstration) to learn a wider region of attraction (green) around the demonstration data (orange) than BC.

auxiliary imitation loss in conjuction with the policy update [27, 19], or modify the reward function such that the agent is rewarded when it imitates the demonstrations [45, 28].

One primary concern of relying on demonstrations is that they are costly to obtain, especially in real-world applications. Requiring an operator providing corrections in-the-loop to handle covariate shift is often prohibitive as well. Given only a few offline trajectories demonstrating successful task completion, an agent ought to be able to replicate the behavior from similar starting conditions, even if there are small perturbations and variations along the way, and correct itself when necessary. We are primarily interested in this setting. In this work, we aim to minimize the number of demonstrations necessary for sparse-reward tasks, while preserving successful task completion. To tackle covariate shift, we seek an approach that will be *robust*, in the sense of Figures 1a and 1b: the agent is only trained with a few demonstrations from a single starting state, but at evaluation time, it must generalize its behavior to a variety of unseen starting states. If the agent can successfully complete the task from states adjacent to the demonstrated trajectory, then it can recover from small deviations from the demonstration.

Inspired by the notion of "funnels" in robotics [17] and feedback control [2, 16], we introduce a reverse-time generative model that can generate possible paths leading the agent back onto the demonstrations. These reverse rollouts provide useful information because every rollout ends within the support of the demonstration data. Assuming that the demonstrations lead to the goal, imitating these generated reverse rollouts and the original demonstration data allow the agent to reach the goal from more starting conditions, including unseen ones. As illustrated in Figures 1c and 1d, typical behavior-cloning (BC) methods focus learning on the small number of demonstrated states, whereas our proposed approach, Backwards Model-based Imitation Learning (BMIL), uses the reverse rollouts to learn a wider region of attraction around the demonstration. We validate our approach on a number of long-horizon, sparse-reward continuous-control tasks. Even from 5–20 demonstrations, BMIL provides a significant increase in the region of attraction and robustness on many domains compared to BC, or to using a forward dynamics model.

We summarize our contributions as follows:

- We propose an imitation learning method that pairs a backwards dynamics model with a policy and train on both demonstrations and imagined model rollouts.
- In the restrictive setting of an offline expert and no access to environment interactions, we show that a backwards model can improve robustness over behavior cloning.
- Our experiments on a variety of long-horizon, sparse-reward domains demonstrate that BMIL can noticeably extend the region of attraction around the demonstration data, even when trained on a very small subsets of the state space.

2 Related Work

Imitation learning has a long history [24, 32] and is well-studied, as documented in various surveys [1, 22, 9]. The challenges of covariate shift and compounding errors are also well-known [24, 29]. Most solutions involve on-policy imitation learning, where environment interaction and interactive querying of the expert allow for the agent's distribution to match that of the expert [30, 36]. More closely related to our approach are methods that modify or augment the demonstrations to increase robustness. Laskey et al. [13] inject noise into the supervisor's policy during training to force the demonstrator to provide corrections. Luo et al. [15] learn value function and dynamics model from demonstrations,

using this to conservatively extrapolate a value function that encourages the agent to return to the expert data distribution. Generative approaches have also been used in imitation learning [8, 42], but their focus is not on robustly following a few goal-reaching demonstrations.

Time-reversibility has been explored in RL, often as a form of regularization [37, 20, 44, 26, 31]. Reverse-time dynamics models, also called predecessor models, have also been used in RL [3, 4, 34, 12, 14, 43, 5]. However, in all cases, the reverse-time dynamics model is used as either an alternative to the forward-time dynamics model, or as an auxiliary model in addition to the standard forward-time dynamics model, in order to mitigate model-compounding errors. The result is that the reverse-time dynamics model can accelerate RL and enable greater sample-efficiency. In this work, we take a different perspective where the reverse-time dynamics model used to generate possible, unseen paths that can lead back to the expert demonstration and thus to the goal, thereby improving robustness in following the demonstration. Our work is also similar to [41], where a reverse-time dynamics model is used to generate possible trajectories; however, their focus is on offline RL, where the generated trajectories are used to connect distinct sets of states in the offline dataset.

3 Method

3.1 Preliminaries

We model the setting as a Markov Decision Process (MDP) with a continuous state space $\mathcal{S}$ and a continuous action space $\mathcal{A}$. $T : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the transition function that defines the distributions of the next state $s_{t+1} \in \mathcal{S}$, given the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ taken at timestep t.

The objective of imitation learning is to learn a policy π_θ parameterized by θ, that matches the expert's policy π_E. We assume that the expert generates demonstrations $\mathcal{D}_{demo}$ by rolling out its policy π_E in the environment. Note that we consider the more restrictive case of where demonstrations only consist of transitions (s, a, s') and not rewards, where s' is the next state and r is the reward.

Behavior cloning (BC) is a form of imitation learning where the policy π_θ learns to imitate expert actions via supervised learning. The policy learned by behavior cloning is found by minimizing the negative log likelihood over the demonstration data

$$\mathcal{L}_{BC} = \mathbb{E}_{(s,a)\in\mathcal{D}_{demo}} \left[-\log \pi_\theta(a \mid s) \right].$$

Note that BC does not require more interactions with the environment and can be considered to be offline. Furthermore, the expert behavior is inferred only from demonstrations, without access to the expert policy itself.

Compounding errors in behavior cloning As pointed out in [29, 40], behavior cloning suffers from covariate shift, where errors in the policy can compound and lead the agent to states where it cannot recover. Intuitively, this occurs as states in the training data $\mathcal{D}_{demo}$ is a small subset of the entire state space $\mathcal{S}$ and it is difficult for the policy π_θ to learn the optimal action for states outside of the training data. If during a rollout, once the policy π_θ makes an error and leaves the $\mathcal{D}_{demo}$, it may encounter completely new states, leading to the compounding of errors. Furthermore, as the agent moves farther away from $\mathcal{D}_{demo}$, there is very little hope for it to make the correct action and move back onto the distribution of the training data.

3.2 Problem Setting

We consider the same setting as BC, where we do not assume access to the environment or the expert policy during training and that only expert demonstrations with no rewards are given. Furthermore, we consider the setting where the expert demonstrations are given in the MDP where the set of initial states $s_0 \in S_0$ and goals $g \in \mathcal{G}$ is restricted to a very small subset of the state space. In other words, the expert only provides demonstrations for a limited number of initial states and goals. An example of this scenario is where the environment is a maze and the agent starts from the same initial state and the goal is fixed. We assume that the expert policy is optimal or nearly optimal, and that all demonstrations reach a goal. Formally, we define the demonstration trajectories τ_{demo} as coming

from a probability density

$$p(\tau_{demo} \mid \pi_E) = p(s_0) \prod_{t=0}^{T-1} \pi_E(s_t \mid a_t) p(s_{t+1} \mid s_t, a_t),$$

where $s_0 \in S_0$, and $s_T \in \mathcal{G}$. We assume that the set of initial states and goals is very low-dimensional manifold of the entire state space: in all of our experiments, S_0 and $\mathcal{G}$ consist of a single start or goal state and the ε-ball of its neighborhood. As such, only a few demonstrations are required to learn a stable optimal policy.

Our objective is to learn a policy that is robust to policy errors when imitating expert behavior and can learn to reach the goal from a variety of initial states. This is different than the multi-goal or multi-task setting, where the agent learns to solve multiple goals or tasks, usually from a small number of initial states. More formally, robustness of the policy $R(\pi_\theta)$ is defined as

$$R(\pi_\theta) = \mathbb{E}_{s_0 \in S_R} \left[\mathbb{1}\{\exists t \leq T, s_t \in \mathcal{G}\} \right],$$

where the expectation is taken over S_R that is strictly greater than the initial states given in $\mathcal{D}_{demo}$, $|\mathcal{S}_R| > |S_0|$. Note that our measure of robustness is somewhat coarse, in that we do not consider the shortest path to reach the goal from every start state (which would probably require more information such as diverse trajectories or environment interations). Instead, we seek to extend the region of attraction around $\mathcal{D}_{demo}$ such that the learned policy can still reach the goal.

As we consider continuous states in our work, we measure the robustness R using samples from S_R, either by sampling randomly or by using stratified random sampling over a discretized grid of the state space. Furthermore, we are concerned with being able to learning robustness in terms of spatial coordinates such that the agent can learn to go back onto the optimal path, rather than other state dimensions such as joint velocities. Thus we restrict the expectation to only the spatial dimensions.

If the policy π_θ can learn to reach the goal from different initial positions not contained in the demonstrations, then it can also perform the correct action when an error is made during a rollout, making the policy robust to deviations. Throughout, we assume that there exists an action $a \in \mathcal{A}$ that allows the policy to go towards $\mathcal{D}_{demo}$ when in a state $s \notin \mathcal{D}_{demo}$. This scenario is true for many navigation and physics-based domains if we ignore the rare circumstances such as irrecoverable unsafe states or the breakdown of the agent.

3.3 Backwards Model-based Imitation Learning

In our work, we use a backwards dynamics model to provide more synthetic training data to the policy and therefore increase the policy's robustness. We call our method backwards model-based imitation learning or BMIL.

Backwards model The backwards model is a probabilistic generative model defined as $B = p(s_t, a_t \mid s_{t+1})$. This model estimates the conditional distribution of the reverse time dynamics. It takes in the next state and outputs the previous state and previous action. As we consider only continuous state and action spaces, we implement B as a conditional Gaussian, parameterized by ϕ.

The backwards model is decomposed into two functions $B = B_A \cdot B_S = p(a_t \mid s_{t+1}) \cdot p(s_t \mid a_t, s_{t+1})$, an action generator and previous state generator. The action generator M_A predicts which action was taken in order to land to the next state. There may be several such actions from different states that can lead to the next state. Thus the action generator implicitly encodes a backwards policy. It is important for this backwards policy to closely match the learned forward policy π but be different enough to generate diverse new traces such that the policy can become robust by imitating these traces.

The previous state generator B_S predicts the previous state given the next state and previous action taken. The goal of this generator is to accurately predict the backwards dynamics, which are learned using rollouts from the policy π. Unlike the action generator, this generator should be as accurate as possible in order to produce accurate traces.

As we consider the setting with no access to the expert or the environment, the reverse time model B is trained only on the demonstrations. The action generator and previous state generator is jointly

trained by maximum likelihood

$$\mathcal{L}_B = \sum_{t=0}^{T} \log p(a_t \mid s_{t+1}) + \log p(s_t \mid a_t, s_{t+1}), \tag{1}$$

where s_t, a_t, s_{t+1} is the state, action, and next state, respectively, at timestep t.

Model rollouts Given expert demonstrations τ_E, we use the backwards model to generate possible several short reverse rollouts or traces τ_B, starting from every state in τ_E. Note that all of these traces end on states within the demonstration data $s \in \mathcal{D}_{demo}$ and as the demonstration reaches the goal, following these traces will eventually lead to the goal. For all $s_1, \ldots, s_T$ in τ_E, we generate K traces τ_B by repeatedly rolling out previous state-action pairs from B until the desired horizon length H is reached. These traces are collected into a buffer $\mathcal{D}_M$. If the reverse time model B is accurate and the previous action generator B_A gives sufficiently diverse actions, the traces τ_B are then samples from the region of attraction or "funnels" around every state along the demonstration. As we assume all demonstrations reach the goal, these samples from the funnels can be used to learn a robust policy π_θ, as it can follow the traces onto the optimal path.

Action selection strategy for B_A As the backwards model B is trained only on expert data with a limited number of demonstrations, it is very likely that the backwards model can only learn accurate reverse-time dynamics for states contained within or close to the demonstration data $\mathcal{D}_{demo}$. Thus repeatedly rolling out B_A would only generate traces whose (s, a) pairs are contained within $\mathcal{D}_{demo}$ and would not help with learning robust policies. We thus need to choose an action selection strategy for B_A when generating these traces in order to provide synthetic transitions to robustify π_θ. A good action selection strategy would be similar to the reversed version of the expert policy π_E, but would choose put more weight into less probable actions. In other words, the action selection strategy for B_A should "cover" the probability distribution $p(a_E|s)$, but have enough of a wide tail to provide diverse rollouts. As our backwards model B is probabilistic (implemented as a conditional Gaussian with diagonal covariance), we can easily perturb $p(a_E|s)$ by increasing the distribution's variance. Let $a \sim \mathcal{N}(\mu, \sigma^2) = B^A_{\phi_A}(a|s')$. We consider two possible ways to generate action a: 1) simple scaling of the σ and 2) resampling a noisy version of a, $a' \sim \mathcal{N}(\mu, \sigma^2), a = \mathcal{U}[a' - k, a + k]$, where k is a fixed hyperparameter. For the scaling strategy, we further scale σ by entropy of the probability density as we wish to make the distribution "wider" for more peaky distributions. As we do not wish to deviate too much from the demonstration data, we perturb only the first action when $s' \in \mathcal{D}_{demo}$.

As the backwards model B is only trained on $\mathcal{D}_{demo}$, it is very likely that B becomes increasingly inaccurate the farther away state s' is from $\mathcal{D}_{demo}$ (though some generalization may occur states sufficiently close to $\mathcal{D}_{demo}$). However, we show that even in this constrained scenario, we find that having short reverse rollouts from the backwards model still helps to increase robustness in our experiments. In addition to using relatively short rollouts that limits compounding errors, we hypothesize that this is these traces do not necessarily need to be accurate in order to benefit the policy. It may be that having the general correct direction contained in the traces is sufficient for the policy to eventually reach states in the demonstration data.

Algorithm Our overall method BMIL is outlined in Algorithm 1. Given expert demonstrations of the form (s_t, a_t), we train the backwards model using Eqn. 1 to estimate the reverse-time dynamics $p(s_t, a_t \mid s_{t+1})$. We train our policy π_θ on both the demonstration data $\mathcal{D}_{demo}$ and the model traces τ_B by sampling from both at a fixed ratio and using maximum likelihood,

$$\mathcal{L} = p_d \mathcal{L}_{BC} + (1 - p_d)\mathbb{E}_{(s,a)\sim\tau_B}\left[-\log \pi_\theta(a \mid s)\right], \tag{2}$$

where p_d is the probability of sampling from $\mathcal{D}_{demo}$. As our aim is to learn a robust policy while still succeeding at the original start states and goals, we sample from the demonstrations a higher ratio than the model traces.

As we consider the restrictive case of an offline expert and demonstrations with no rewards, we pair behavior cloning with the backwards model. Note that BMIL does not depend on the type of imitation learning policy; any algorithm can potentially be used as long as the policy can train with samples of the form (s, a).

Algorithm 1 Backwards Model-based Imitation Learning (BMIL)

for N epochs **do**
  Train backwards model parameters ϕ using Eqn. (1).
  Generate K model traces τ_B from every state $s \in \mathcal{D}_{demo}$ and store them in $\mathcal{D}_M$.
  for M steps **do**
    Sample mini-batch of (s, a) from $\mathcal{D}_{demo}$ and $\mathcal{D}_M$ at a fixed ratio.
    Update policy parameters θ using Eqn. (2).
  end for
end for

4 Experiment Design

4.1 Environments

We validate our approach on several continuous control domains: 1) the Fetch robotics environment [23] and 2) maze navigation with two different agents (see Figure 6 and Figure 7 in Appendix A). For the Fetch robotics environments, we consider the "Push" and "PickAndPlace" tasks. For the maze environments, we consider 3 mazes of increasing difficulty and use two agents. All domains use the Mujoco simulator [38] for a total of 8 distinct domains. All environments have sparse reward structures, where either only every step has a constant negative reward until the goal is reached (Fetch) or only the goal has a non-zero reward (Maze). More details are provided in Appendix A.

4.2 Demonstrations

For all environments, we obtain demonstrations by first training an expert policy. The expert policy is trained by concatenating the goal position to the state, as in goal-oriented learning. For Fetch, we use pre-defined settings in [25] to train the expert policy. For the maze environments, we predefine a series of subgoals, and the position of the next subgoal is concatenated to the state. The reward is dense, using the negative Euclidean distance to the next subgoal and subsequent subgoals, and we use soft actor-critic [6] as the expert policy.

We rollout the expert policy to obtain 5 and 10 expert demonstrations for the Push and PickAndPlace environments, respectively. As each episode is of length 50, this amounts to a total of a couple hundred training samples. We thus make 10 identical copies of each demonstration in order to stably train the policy and backwards model (we do the same for all other methods). This has a similar effect of performing more gradient updates, but we found it is computationally cheaper. For Maze, we use 20 demonstrations for all 6 environments and do not repeat the demonstrations as the episodes are sufficiently long.

4.3 Implementation Details

For the policy, we use neural networks with 3 fully connected hidden layers with 256 neurons and ReLU activations. For the backwards model, we use 4-layer MLPs with 256 hidden units for both the action predictor B_A and previous state predictor B_S and use diagonal Gaussian distributions. To train the policy, we use $p_d = 0.5$ for the Fetch environments and at higher ratios $p_d = 0.8 \sim 0.95$ for the Maze environments. We find that higher demo ratios are necessary for longer-horizon and more complex domains. For the model rollouts, we use the variance scaling action selection strategy for the first action only and use an increasing rollout length of 1 to 10 for all domains, similar to [10]. All other hyperparameters and details are provided in the appendix.

4.4 Evaluation

We evaluate BMIL against behavior cloning (BC) and VINS [15]. VINS specifically aims to learn value functions robust to perturbations using negative sampling and the induced policy learns self-corrective behavior. VINS was chosen as it is most relevant to our setting; other methods such as DART [13] or SQIL [28] require either an online expert or environment interactions.

We use the same number of demonstrations for all methods and also keep the same policy network architecture and the total number of policy gradient steps equal across all methods. We train both

			Robustness (%)			Relative to BC		
			BC	VINS	BMIL	BC	VINS	BMIL
FETCH	Push (5 demos)		12.1±0.3	12.8±0.4	**14.6**±0.6	1	1.06	**1.21**
	PickAndPlace (10 demos)		4.1±0.1	3.4±0.1	**17.5**±0.9	1	0.84	**4.31**
MAZE	Point (20 demos)	UMaze	**48.8**±2.0	39.7±2.1	**47.9**±3.6	**1**	0.81	**0.98**
		Room5x11	36.3±3.2	22.0±3.4	**43.8**±3.9	1	0.61	**1.21**
		Corridor7x7	36.8±1.6	**37.5**±1.4	**39.5**±1.9	1	**1.02**	**1.07**
	Ant (20 demos)	UMaze	**62.1**±1.2	44.2±1.9	**64.9**±1.4	**1**	0.71	**1.04**
		Room5x11	**32.9**±0.9	30.2±0.7	28.7±0.7	1	0.92	0.87
		Corridor7x7	**21.6**±0.5	19.4±0.5	17.7±0.5	**1**	0.90	0.82

Table 1: Robustness evaluation for Fetch and Maze environments over 400 and 100 trials, respectively. The bounds indicate 95% confidence intervals. BMIL improves robustness considerably over BC in Fetch and Point maze environments.

the policy and backwards model until the backwards model loss converges. Note that our goal is not to solve the training task faster but rather to robustify the policy using the backwards model. Additionally, we wish to do solve the task at a various starting conditions while still being able to succeed at the original initial start-goal states.

To evaluate the robustness of the learned policy, we change the initial positions to random positions in the state space. For Fetch, we fix the initial gripper, object, and goal position during training and vary the gripper's x and y position within the table boundaries during evaluation. We use 10000 samples during evaluation. For Maze, we initialize the agent to a random start position within a discretized grid of the maze. We sample 100 for each discrete grid cell and compute the success rate. By sampling points for each grid cell, this gives us an idea of which positions are easy for the agent to reach the goal. Intuitively, such positions would be those near the goal and the demonstrated path.

5 Results

Our experiments aim to answer the following questions: 1) how robust of a policy does BMIL learn? and 2) what components of BMIL are important to improve robustness?

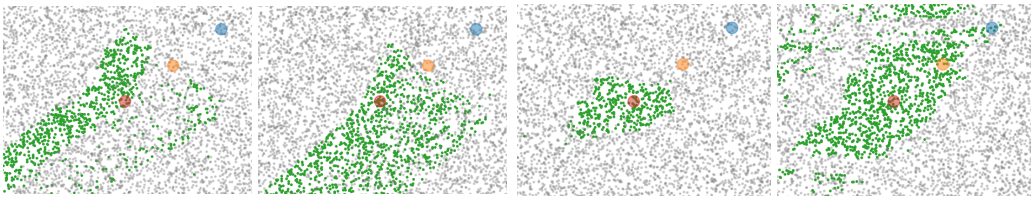
5.1 Robustness evaluation

The robustness results are shown in Table 1. We note that the absolute robustness percentages are generally low for all methods because it is very difficult to extrapolate from limited demonstrations $(5 - 20)$ from a single pair of initial start and goal states. We therefore also include the relative improvement over BC.

On the Fetch environments, BMIL substantially increases robustness over BC and VINS. In particular, our method has an approximately 20% and 330% improvement over BC on Push and PickAndPlace, respectively. VINS on the other hand performs similarly to BC. For the Maze environments, BMIL generally outperforms BC for the Point agent, while the robustness is decreased for the Ant agent. Somewhat surprisingly, BC performs quite well on the long-horizon Maze domains. It may be that as BC has some built-in extrapolation capabilities or that the backwards model may need a better latent representation using an encoder.

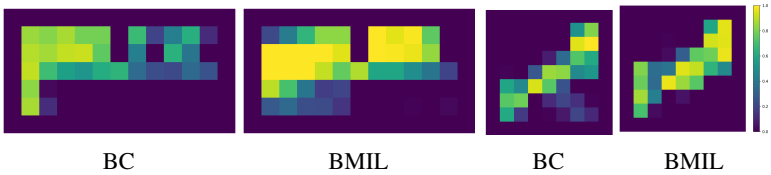
The success rates during training are shown in Table 3 in the Appendix. BMIL achieves success rates close to 1 for most domains, suggesting that increased robustness does not necessarily come at the cost of decreased performance on demonstrated start-goal states. On the other hand, VINS cannot consistently succeed during training, even though its robustness is similar to BC.

Visualization of robustness Figure 2 shows which starting positions succeed during the robustness evaluation for Fetch. The green points correspond to successful episodes. We can see that both BC and BMIL succeed more frequently when starting from nearby the demonstration data (approximately a straight line from the start (red) to goal (blue)). However, we can see that BMIL learns a much



(a) Push (b) PickAndPlace

Figure 2: Visualization of robustness on Fetch for BC and BMIL. We vary the gripper's x, y position to evaluate robustness. The green and gray points denote successful and unsuccessful initial positions. The red, orange, and blue points denote the initial start, object, and goal during training. BMIL learns a much larger region of attraction along the path from the initial start (red) to the goal position (blue) and even succeeds in some other areas much farther away.



(a) PointRoom5x11 (b) PointCorridor7x7

Figure 3: Visualization of robustness on PointRoom5x11 and PointCorridor7x7 for BC and BMIL. Positions in the domain is discretized into a grid and 100 random points are sampled as the initial state position. Bright yellow corresponds to 100% success rate while dark purple corresponds to 0%. The regions of attraction for both BC and BMIL are similar in size but BMIL succeeds more often within its region of attraction.

larger region of attraction over BC and even succeeds at points that are much farther away from $\mathcal{D}_{demo}$. We hypothesize that instead of perturbing a single state within $\mathcal{D}_{demo}$ as done in VINS, learning a short reverse rollout from this state allows BMIL to learn optimal paths from states much farther away from $\mathcal{D}_{demo}$, leading to higher robustness values. Figure 3 shows a similar visualization for some Point maze environments where the agent's position is discretized into a grid. BMIL learns a region of attraction similar in size to that of BC, but has a higher rate of success at each cell.

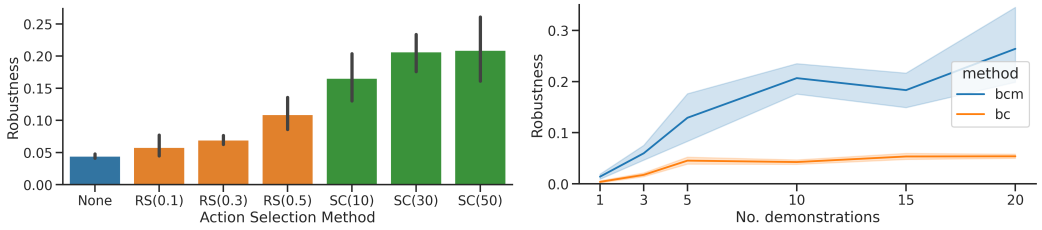
5.2 Additional Experiments

Forwards vs Backwards Dynamics We first analyze the utility of a backwards vs a forward dynamics model. On the Fetch environments, we train BMIL with a forward dynamics model $p(s' \mid s, a)$ and compare against the original backwards model $p(s, a \mid s')$. To generate model rollouts from demonstration states, we use the action from the policy $a = \pi_\theta(s)$. As shown in Table 2, the forwards model offers little to no benefit over BC for both Push and PickAndPlace, suggesting that the backwards model is needed to produce a robust policy.

Action selection strategy We ablate different action selection strategies in trace generation in Figure 4a (and Figure 8 in the appendix). Compared to no perturbation (None), we see that either

	Robustness (%)			Relative to BC		
	BC	BMIL (Forwards)	BMIL (Backwards)	BC	BMIL (Forwards)	BMIL (Backwards)
Push (5 demos)	12.1$\pm$0.3	12.4$\pm$0.6	14.6$\pm$0.6	1	1.03	1.21
PickAndPlace (5 demos)	4.1$\pm$0.1	4.1$\pm$0.2	17.5$\pm$0.9	1	1.03	4.31

Table 2: Forwards ($p(s' \mid s, a)$) vs Backwards ($p(s, a \mid s')$) dynamics model: The forwards dynamics model performs similarly to BC and does not increase robustness.



(a) Action selection strategy

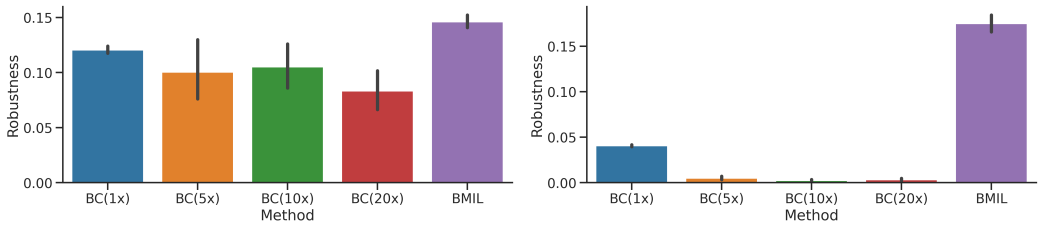
(b) No. of demonstrations

Figure 4: **(a)** Action selection strategy: effect of no perturbation (None), resampling (RS), and scaling (SC) on robustness on PickAndPlace. The numbers in the brackets indicate coefficients. We see that perturbing the first action is beneficial compared to the no perturbation (None) method and allows the backwards model to generate diverse traces. **(b)** Number of demonstrations on PickAndPlace: more demonstrations increase robustness for BC and BMIL, but BC plateaus at a much lower level.

action selection strategy improveness robustness as it can lead to more diverse trajectories unseen within the support of the demonstrations. We use the variance scaling strategy SC(30) for all Fetch experiments as it was more stable than SC(50).

Number of demonstrations We also studied our method with varying numbers of demonstrations. As shown in Figure 4b, both BC and BMIL improve in robustness with more demonstrations, but BC plateaus at a much lower level. On the other hand, BMIL requires slightly more (10) demonstrations than needed (3 demonstrations are sufficient for BC to succeed during training) in order to train the backwards model (Figure 9 in the appendix).

Computation budget As BMIL trains both the policy and the backwards model, it requires more total gradient updates than BC. On Fetch domains, BMIL uses approximately 5x more computation than BC. We train BC for more steps to match or exceed BMIL's computation budget, as shown in Figure 5 (BC given 1x–20x computation budget). However, more training for BC does not improve robustness and in fact seems to have a harmful effect on robustness.



Push

PickAndPlace

Figure 5: Computation budget: we increase the number of gradient steps for BC by $1 - 20$x for Fetch. BMIL has roughly 5 times more total gradient steps than BC(1x) due to the backwards model update. More BC gradient steps does not increase robustness.

6 Discussion

This work proposed a method to tackle the issue of covariate shift in imitation learning. We considered the restrictive setting where the expert is offline, where its behavior can only be inferred from demonstrations, and no access to additional environment interactions. Specifically, we show that pairing a generative backwards model with behavior cloning can allow a policy to learn a wider region of attraction around the demonstration data. By rolling out imagined traces from states within the demonstration and perturbing actions to generate diverse traces, BMIL learns a wider funnel than naive BC. Through experiments on several long-horizon, sparse-reward, continuous control domains, BMIL noticeably improves robustness when trained on a narrow set of initial start and goal states and evaluated at random starting positions. Future work will include applying BMIL to more difficult tasks and extending BMIL to allow for environment interactions and pairing with other imitation learning algorithms.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2008.10.024. URL https://www.sciencedirect.com/science/article/pii/S0921889008001772.

[2] Robert R. Burridge, Alfred A. Rizzi, and Daniel E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6): 534–555, 1999.

[3] Ashley D. Edwards, Laura Downs, and James C. Davidson. Forward-backward reinforcement learning, 2018.

[4] Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HygsfnR9Ym.

[5] Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. There is no turning back: A self-supervised approach for reversibility-aware reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=3X65eaS4PtP.

[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2017.

[7] Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations. In *AAAI*, 2018.

[8] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf.

[9] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), apr 2017. ISSN 0360-0300.

[10] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

[11] Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/fd5c905bcd8c3348ad1b35d7231ee2b1-Paper.pdf.

[12] Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu. Bidirectional model-based policy optimization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5618–5627. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/lai20b.html.

[13] Michael Laskey, Jonathan Lee, Roy Fox, Anca D. Dragan, and Ken Goldberg. DART: noise injection for robust imitation learning. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, volume 78 of *Proceedings of Machine Learning Research*, pages 143–156. PMLR, 2017. URL http://proceedings.mlr.press/v78/laskey17a.html.

[14] Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5757–5766. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/lee20g.html.

[15] Yuping Luo, Huazhe Xu, and Tengyu Ma. Learning self-correctable policies and value functions from demonstrations with negative sampling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rke-f6NKvS.

[16] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[17] Matthew T. Mason. The mechanics of manipulation. In *IEEE International Conference on Robotics and Automation*, 1985.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[19] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[20] Suraj Nair, Mohammad Babaeizadeh, Chelsea Finn, Sergey Levine, and Vikash Kumar. Trass: Time reversal as self-supervision. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 115–121, 2020. doi: 10.1109/ICRA40945.2020.9196862.

[21] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *NeurIPS*, pages 14814–14825, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c8cc6e90ccbff44c9cee23611711cdc4-Abstract.html.

[22] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018. ISSN 1935-8253. doi: 10.1561/2300000053. URL http://dx.doi.org/10.1561/2300000053.

[23] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018.

[24] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

[25] Antonin Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

[26] Nasim Rahaman, Steffen Wolf, Anirudh Goyal, Roman Remme, and Yoshua Bengio. Learning the arrow of time for problems in reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rylJkpEtwS.

[27] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[28] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xKd24twB.

[29] Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/ross10a.html.

[30] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

[31] Harsh Satija, Philip Amortila, and Joelle Pineau. Constrained Markov decision processes via backward value functions. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8502–8511. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/satija20a.html.

[32] Stefan Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997.

[33] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[34] Yannick Schroecker, Mel Vecerik, and Jon Scholz. Generative predecessor models for sample-efficient imitation learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkeVsiAcYm.

[35] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[36] Wen Sun, Arun Venkatraman, Geoffrey J. Gordon, Byron Boots, and J. Andrew Bagnell. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3309–3318. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sun17d.html.

[37] Pierre Thodoroff, Audrey Durand, Joelle Pineau, and Doina Precup. Temporal regularization for markov decision process. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/4b0250793549726d5c1ea3906726ebfe-Paper.pdf.

[38] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[39] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[40] Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 3024–3030. AAAI Press, 2015. ISBN 0262511290.

[41] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination, 2021.

[42] Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/044a23cadb567653eb51d4eb40acaa88-Paper.pdf.

[43] Tao Yu, Cuiling Lan, Wenjun Zeng, Mingxiao Feng, Zhizheng Zhang, and Zhibo Chen. Playvirtual: Augmenting cycle-consistent virtual trajectories for reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=GSHFVNejxs7.

[44] Shangtong Zhang, Vivek Veeriah, and Shimon Whiteson. Learning retrospective knowledge with reverse reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19976–19987. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e6cbc650cd5798a05dfd0f51d14cde5c-Paper.pdf.

[45] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, JÃ¡nos KramÃ¡r, Raia Hadsell, Nando de Freitas, and Nicolas Heess. Reinforcement and imitation learning for diverse visuomotor skills. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.009.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code and data will be provided in the supplementary material, and also released prior to final publication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Briefly in Section 4, and in full in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]