Fighting Causal Confusion in Behavior Cloning with Multiple Environments

Han Lin
hl3199@columbia.edu
MS in Computer Science, Columbia University

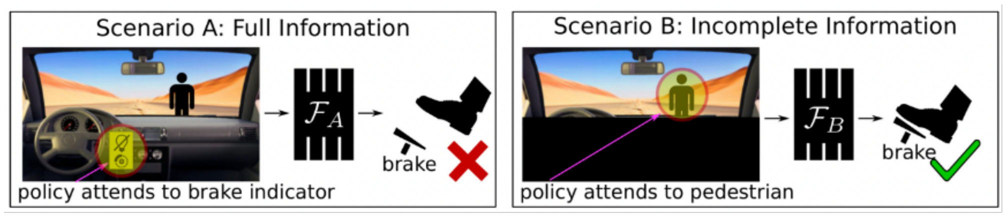
Abstract

Offline imitation learning aims at training learning agents with collected offline expert demonstrations without further interactions with the environments. Traditional Behavioral Cloning (BC) algorithm in imitation learning uses supervised learning approach to directly map input observations to expert actions, which is prone to causal confusions (spurious correlations) due to the difference among the observed variables from each training environment, and may not be able to generalize well [13]. Therefore, it is tempting to ask could we design a method that explicitly learns the underlying causal structure to tackle this issue. However, recent literature [21] shows a sober look that learning disentangled representations is fundamentally impossible without additional information on the models or data. So we need at least some additional information (e.g. multiple environments) to make the causal graph identifiable. Greatly inspired by the Nonlinear IRM model proposed by Lu et al. [22], we consider the setting of learning from expert's demonstrations under multiple environments, with the aim of generalizing well to new unseen environments. We make several adjustments of the original iCaRL three-phase procedure to adapt it to our imitation learning tasks, and proposed our new algorithm, Invariant Behavioral Cloning (IBC). We compare our method against several benchmark methods on three OpenAI Gym control tasks and show its effectiveness in learning imitation policies capable of generalizing to new environments. Finally, to boost our understanding, we also conduct extensive ablation tests over different part of our algorithm, which we believe could inspire future research in the direction of causal imitation learning.

1 INTRODUCTION

Offline Imitation Learning (OIL) aims at learning policies directly from existing expert demonstrations, which effectively avoids the need for costly environment interactions [28, 1]. For example, it would be dangerous and impractical for an agent to learn how to self-drive by making trails on real roads. The simplest method in OIL is Behavioral Cloning (BC), which solves a supervised learning problem over state-action pairs from expert demonstrations. However, recent findings suggest that BC suffers from spurious correlations (also called as causal confusion), where the learned policy depends on some nuisance variables strongly correlated with expert actions or training environments, instead of the true causes. [13, 33, 12]. Fig. 1 in [13] provides a good illustration example for spurious correlations in self-driving where more information yields worse imitation learning performance.

Illustrative Example: Consider using imitation learning to train a self-driving car (see Fig. 1). In scenario A, the input to the model is an image of the dashboard and windshield, where the dashboard has a *yellow indicator light* that comes on immediately when the brake is applied. While in scenario B, the input to the model (with identical architecture) is the same image but with the dashboard masked out. Both cloned policies achieve low training loss, but when tested on the real road, model B drives well, while model A does not. This is because model A wrongly learns to apply the brake only when the brake light is on. Even though the brake light is the **effect** of braking, model A could achieve low training error by misidentifying it as the **cause** instead.


Scenario A: Full Information
$\mathcal{F}_A$
brake
policy attends to brake indicator
Scenario B: Incomplete Information
$\mathcal{F}_B$
brake
policy attends to pedestrian
Figure 1: Example of spurious correlation (causal confusion). Directly applying Behavioral Cloning (Scenario A) will make the agents incorrectly learn the indicator light as cause of brake (ref: Fig. 1 in [13]).

If we rethink about this example, we would notice that the *indicator light* is perfectly correlated with the brakes. Therefore, it would be impossible to distinguish whether such *indicator light* is the cause or effect of taking brakes. Haan [13] provides a smart and simple idea to eliminate such nuisance variables: rather than learning the underlying causal graph directly, it randomly mask out a certain percentage of latent variables (e.g. features learned from VAEs) during the training process. If the training performance does not deteriorate for a certain mask, then the variables masked out are unlikely to be direct parents of expert actions, and thus could be removed safely. Spurious correlations are also eliminated together since the expert's policy is independent on these variables. Such method works well in several OIL tasks including OpenAI Gym and Atari Games [13, 25].

However, we are still curious about whether there exist better methods that could identify these nuisance variables by learning the causal structure directly. Sadly, recent progress shows a sober look that unsupervised learning of disentangled representations is fundamentally impossible without inductive biases on both the models and the data [21]. On the other hand, if we have expert demonstrations from multiple environments, it would be possible to identify the causal structure under certain assumptions [22]. Methods including Invariant Risk Minimization (IRM) is widely used when learn from multiple environments [2].

Our Contributions: With motivations stated above, we study offline imitation learning with expert demonstrations from multiple environments in this paper. Our contribution is three folds:

- To the best of our knowledge, we are one of the first to incorporate causal structure into imitation learning tasks, and the first open source implementation[1] of the recent model proposed by Lu et al. [22]. We also adapted several parts of their model to make it work for imitation learning tasks.
- Previous works either use non-causal approaches (without using causal inference to find direct causes of actions) [7, 13, 25], or are too theoretical towards causal inference [34, 19] (unclear how to apply to real imitation learning tasks). Our approach keeps a balance between the two.
- We make extensive ablation tests to show the effectiveness of each part of our proposed algorithm.

[1]Github repo for this project: `https://github.com/HL-hanlin/STAT8101_AppliedCausality`

2 RELATED WORK

Offline Imitation Learning with Behavioral Cloning: Imitation Learning aims at learning useful skills or behaviors from expert demonstrations [3, 16, 20, 27]. There are two main approaches for IL: inverse reinforcement learning methods that find a cost function under which the expert is uniquely optimal [9, 17, 24], and behavioral cloning methods that formulate the IL problem as a supervised learning problem that predicts expert actions from states [4, 5, 8]. Our work employs behavioral cloning as it exhibits the benefit of avoiding costly and dangerous environment interactions, which is crucial for applying imitation learning to real-world scenarios.

Causal Confusion and Nuisance Correlates: Behavioral Cloning is known to suffer from distributional shift, where the state distribution induced by a policy gets different from the training distribution on which the policy was trained. Recently, it has been evidenced that distributional shift leads to the causal confusion problem [5, 13, 33], where policies exploit the nuisance correlates in states for predicting expert actions. The starting point of our paper is de Haan et al. [13], which learns a policy with randomly masked disentangled representations and infers the best mask through during environment interaction.

Invariant Risk Minimization: A general idea to solve the causal confusion problem is to identify which features describe true correlations of interest that are stable across environments. [2] proposes Invariant Risk Minimization (IRM), which seeks to find data representations for which the optimal predictor is invariant across all environments. However, the general formulation of IRM is a challenging bi-leveled optimization problem, and the practical version of IRM, which is called IRM-v1, does not perform well on non-linear classifiers. Therefore, it is natural to seek IRM methods that works well for non-linear classifier, since the policy network in IL is usually represented by nonlinear neural networks. Lu et al. [22] proposes a framework for non-linear IRM which uses non-factorized prior that is able to capture complex structures in the latent states. Our paper is largely inspired and built on their approach, with proper adjustments to fit into our tasks.

Causal Discovery from Observational Data: Phase 2 of iCaRL [22] uses independence testing [15] and conditional independence testing [35] to discover the direct parents of $\mathbf{Y}$ (which is $\mathbf{A}$ in our case). However, HSIC and KCIT works better for continuous variables, and the running time are not trivial for large sample size. [23] discretizes continuous variables space, but it does not scales well to the large data size. [26] assumes additive noise models which test either $\mathbf{Z} \to \mathbf{A}$ or $\mathbf{A} \to \mathbf{Z}$. However, we are only aiming for finding the direct parents of $\mathbf{A}$, and a latent variable $\mathbf{Z}$ does not necessarily to be directly connected on the causal graph. Here, we adopt the Fast (Conditional) Independence Test (FIT and FCIT) [11], which is a quite flexible method that works for both continuous and discrete/categorical variables. Since we need to run (conditional) independence tests for all pairs of latent variables, we applied Sidak-Bonferroni [36] to correct for multiplicity [30].

3 PRELIMINARIES

3.1 Notations

The notations commonly used in imitation learning is a little bit different from causal inference, so let's first define some definitions we'll use later in this paper:

$\mathbf{X}$: observations (usually denoted as $\mathbf{O}$ or $\mathbf{X}$ in causal inference, and $\mathbf{S}$ in imitation learning).
$\mathbf{A}$: actions (equivalent to labels $\mathbf{Y}$ in causal inference).
π: imitation policy, which is a function (e.g. neural network) that maps from $\mathbf{X}$ to $\mathbf{A}$.
$\mathbf{Z}$: latent states. $\mathbf{E}$: environments. PA($\mathbf{A}$): direct causes (parents) of $\mathbf{A}$ on the causal graph.

3.2 Problem Formulation

We introduce the standard imitation learning framework with multiple environments in this section. Consider $\mathcal{M} = \{(\mathcal{X}^e, \mathcal{A}, \mathcal{P}^e, r^e, \gamma)|e \in \mathcal{E}\}$ as a set of environments with observations $x^e \in \mathcal{X}^e$, actions $a \in A$, transition probabilities $\mathcal{P}^e : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$, rewards $r^e \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, and discount factor γ. $\mathcal{X}, \mathcal{P}$ and r^e are environment-dependent, while $\mathcal{A}$ and γ are shared across all environments. We have access to an offline training dataset consists of expert demonstrations from multiple environments $\mathcal{D} = \{\{(x_i, a_i)\}_{i=1}^{N^e}|e \in \mathcal{E}_{\text{train}}\}$, with each data point contains an observation x_i together with the expert's corresponding action a_i after observing x_i. Such offline setting, which restricts the learning agent from interacting with the environments, is very common in practice when interacting with the environment is very expensive or impossible as mentioned in Sec. 1.

Our goal is to learn a policy $\pi : \mathcal{X} \to \mathcal{A}$ that maps observations to the expert's actions and matches expert's policy $\pi_{\mathcal{D}}$ across all environments, and even generalizes well to some new environments. If we define $\mathcal{L}$ as the imitation loss (e.g. cross-entropy loss), then we could formalize our goal as finding a policy π that minimizes the maximum imitation loss across all environments:

$$\min_{\pi} \max_{e\in\mathcal{E}} \mathcal{L}^e(\pi, \pi_{\mathcal{D}}) \tag{1}$$

3.3 Comparison Benchmarks

We introduce the algorithm for Behavioral Cloning as well as its combination with IRM-v1 in this section. Since we mentioned CCIL[13] several times in the above sections and the following experimental section, we also include it here for completeness.

Behavioral Cloning (BC): The most commonly seen method in imitation learning is behavioral cloning, which uses supervised learning techniques. To be more specific, let's denote f as an encoder that maps observations $\mathbf{x}_t$ to latent states $\mathbf{z}_t$ in a low dimensional space. Behavioral Cloning learns a policy π that maps the feature map $f(\mathbf{x}_t)$ to the expert action a_t. The policy π and mapping function f are learned by minimizing the cross entropy loss as follows:

$$\mathcal{L}_{\text{BC}}(\mathbf{x}_t, a_t) = \mathcal{L}_{\text{CE}}(\pi(.|f(\mathbf{x}_t)), a_t) \tag{2}$$

Behavioral Cloning + Invariant Risk Minimization (BC+IRMv1): We borrow the idea of Invariant Risk Minimization (IRM) [2] into our imitation learning settings. Since our classifier (policy π) is non-linear, we use the IRM-v1 objective which has a penalty term instead of hard constraints. In our case, $R^e(\bar{\omega} \circ \Phi) = R^e_{\text{BC}} = \mathcal{L}_{\text{CE}}(\pi(.|f(x_t)), a_t)$, and we are aiming to find a policy π that is invariant across different environments.

$$\mathcal{L}_{\text{IRM}}(\Phi, \omega) = \sum_{e\in\mathcal{E}_{\text{train}}} R^e(\bar{\omega} \circ \Phi) + \lambda\|\nabla_{\omega|\omega=1.0} R^e(\bar{\omega} \circ \Phi)\| \tag{3}$$

CCIL (Haan et al. [13]): This method first learns an disentangled representation $f(\mathbf{X}) \in \mathbb{R}^d$ from images using β-VAE. Since each dimension of this disentangled representation could either be a direct cause/parent of expert action or not, there are 2^n possible graphs. Then the author parameterize the structure of the causal graph $\mathbf{G}$ as a vector of n binary variables, each indicating the presence of an arrow from $f(\mathbf{X}_i)$ to $\mathbf{A}$. In order to learn the causal graph, $\mathbf{G}$ is drawn uniformly at random over all 2^n possible graphs, and the paper minimizes the following loss in (4), where $\odot$ denotes elementwise product, and [,] represents concatenation. π is the policy network to be trained. The graph $\mathbf{G}^*$ that minimizes the loss $\mathcal{L}_{\text{CE}}$ inside expectation is the best approximation of the underlying causal graph.

$$\mathbb{E}_G[\mathcal{L}_{\text{CE}}(\pi([f(\mathbf{X}_i) \odot \mathbf{G}, \mathbf{G}]), \mathbf{A})] \tag{4}$$

4 METHOD: Invariant Behavioral Cloning (IBC)

Our proposed algorithm, Invariant Behavioral Cloning, is adapted from iCaRL (Lu et al. [22]). Analogy to their paper, we divide our procedure into three phases as shown in Algorithm 1.

Algorithm 1: Invariant Behavioral Cloning (IBC)

Phase1: First learn a NF-iVAE model by optimizing the objective function in Eq. (5) with $\mathcal{L}^{\text{SM}}_{\text{phase1}}$ replaced by SSM-VR loss in Eq. (21) on the offline dataset $\mathcal{D} = \{\{(x_i, a_i)\}_{i=1}^{N^e} | e \in \mathcal{E}_{\text{train}}\}$. Then we use the mean of the NF-iVAE encoder to infer the latent variables $\mathbf{Z}$ from observations.

Phase2: After inferring $\mathbf{Z}$, we can discover direct causes (parents) of $\mathbf{A}$ by testing all pairs of latent variables with Fast Conditional Independence Test (FCIT), i.e. finding a set of latent variables in which each pair of Z_i and Z_j satisfies that the dependency between them increases after conditioning on A. We apply Šidák-Bonferroni to make correction for multiplicity.

Phase3: Having obtained PA$(\mathbf{A})$, we can solve the optimization problems in Eq. (8) with BC+IRMv1 in Eq. (3). Then we learn another encoder that maps from observations $\mathbf{X}$ to PA$(\mathbf{A})$. When in a new environment, we use this encoder to infer PA$(\mathbf{A})$ from observations, and then leverage the learned policy π for action prediction.

In the following text, we first explain the original methods used by Lu et al. [22], then present the obstacles we found when implement their methods as well as our corresponding improvements. More interesting details can be found in the Appendix (Sec. 8) due to space limit in the main text. We call $q_\phi(\mathbf{Z}|\mathbf{X}, \mathbf{A}, \mathbf{E})$ as encoder, $p_f(\mathbf{X}|\mathbf{Z})$ as decoder, and $p_{T,\lambda}(\mathbf{Z}|\mathbf{A}, \mathbf{E})$ as non-factorized prior.

4.1 Phase 1: Identifying Latent Variables

Original Method: Lu et al. [22] use the NF-iVAE model in phase 1, which is an extended iVAE with non-factorized prior that is able to capture complex structures in the latent states $\mathbf{Z}$. To be more specific, we are aiming at jointly learn ($\mathbf{f}$, $\mathbf{T}$, λ, ϕ) from the following objective:

$$\mathcal{L}_{\text{phase1}}(\theta, \phi) = \mathcal{L}^{\text{VAE}}_{\text{phase1}}(f, \hat{T}, \hat{\lambda}, \phi) - \mathcal{L}^{\text{SM}}_{\text{phase1}}(\hat{f}, T, \lambda, \hat{\phi}) \tag{5}$$

where we use $\hat{f}, \hat{T}, \hat{\lambda}, \hat{\phi}$ to represent copies of f, t, λ, ϕ that are treated as constants and whose gradient is not calculated during training.

The first term $\mathcal{L}^{\text{VAE}}_{\text{phase1}}$ is the Evidence Lower Bound (ELBO). The second term $\mathcal{L}^{\text{SM}}_{\text{phase1}}$ represents score matching loss, which is a well-known method for training unnormalized prior from the general multivariate exponential family distribution. Specifically, we estimate $\mathcal{L}^{\text{SM}}_{\text{phase1}}$ as:

$$\mathcal{L}^{\text{SM}}_{\text{phase1}} = \mathbb{E}_{p_D}[\mathbb{E}_{q_{\hat{\phi}}(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})}[\|\nabla_{\mathbf{z}} \log q_{\hat{\phi}}(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E}) - \nabla_{\mathbf{z}} \log p_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E})\|^2]] \tag{6}$$

$$= \mathbb{E}_{p_D}[\mathbb{E}_{q_{\hat{\phi}}(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})}[\sum_{j=1}^{n}[\frac{\partial^2_{p_{T,\lambda}}(\mathbf{Z}|\mathbf{A},\mathbf{E})}{\partial \mathbf{Z}_j^2} + \frac{1}{2}(\frac{\partial^2_{p_{T,\lambda}}(\mathbf{Z}|\mathbf{A},\mathbf{E})}{\partial \mathbf{Z}_j})^2]] \tag{7}$$

Drawbacks: During our implementation, we found the second derivatives in Eq. (6) have very large variance caused by the oscillation of the norm of latent states Z_j's, which makes the training process not that stable (see Fig. 7 in the Appendix Sec. 8.1.2).

Our Solution: We adopt a recent method called Sliced Score Matching (SSM) [29] to tackle this problem. The intuition of SSM comes from the observation that one dimensional problems are usually much easier to solve than high dimensional ones. Therefore, SSM projects the gradient of $\nabla_{\mathbf{z}} \log q_{\hat{\phi}}(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})$ and $\nabla_{\mathbf{z}} \log p_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E})$ onto some random directions $\mathbf{v}$'s and compares their average difference along these random directions. We use its variance reduction variant (SSM-VR) in our experiments, which optimizes the loss function in Eq. (21). Empirical performance of SSM as well as technical details are in the Appendix Sec. 8.1.2 and Sec. 8.1.3 respectively.

4.2 Phase 2: Discovering Direct Causes

With the encoder which maps from observations $\mathbf{X}$ to latent states $\mathbf{Z}$ we estimated in phase 1, we could now discover direct causes (parents) of actions $\mathbf{A}$. Technical details about how FCIT is implemented with decision trees are in the Appendix Sec. 8.2.2.

Original Method and their drawbacks: [22] uses independence testing [15] and conditional independence testing [35] in their paper. However, during our implementation, we found that these two methods need to construct some $n \times n$ matrices explicitly (n represents sample size), and the running time is too slow to make pairwise comparisons between all latent variables.

Our Solution: We choose to use Fast Conditional Independence Test (FCIT)[2] [11], which is a quite flexible method that (1) works well on continuous and discrete/categorical data, (2) fast process speed for large sample size compared with HSIC and KCIT, and (3) achieves lower Type 1 and Type 2 errors compared with alternative methods.

With such method, we could test all pairs of latent variables $\mathbf{Z}$ by comparing p-value from FCIT with some significance level α. As suggested during the office hour, the probability that at least one of the tests will be significant by change arises as we do FCIT across all pairs of latent variables. so we also correct the significance level to remove multiplicity. In Sec. 6.2, we also mentioned an alternative method to find correct threshold if we do not correct for multiplicity.

4.3 Phase 3: Learning an Invariant Policy

With direct parents of actions $\mathbf{A}$, we can learn an invariant policy π by solving the following optimization problem:

$$\min_{\pi} \sum_{e \in \mathcal{E}_{\text{train}}} R^e(\pi) = \min_{\pi} \sum_{e \in \mathcal{E}_{\text{train}}} \mathbb{E}_{\text{PA}(\mathbf{A}^e),\mathbf{A}^e}[\mathcal{L}_{\text{CE}}(\pi(\text{PA}(\mathbf{A}^e)), \mathbf{A}^e)] \tag{8}$$

[2]`https://github.com/kjchalup/fcit`

We use BC+IRMv1 (Eq. (3) in Sec. 3.3) in replace of this equation to find an environment-invariant policy. We have completed the whole training steps till now.

Then in a new testing environment, we need to infer $\text{PA}(\mathbf{A})$ from observations $\mathbf{X}$.

Original Method: Lu et al. optimize the following objective [22, 31] to infer $\text{PA}(\mathbf{A})$ from $\mathbf{X}$.

$$\max_{\mathbf{Z}} \log p_f(X|\text{PA}(\mathbf{A}), \neg\text{PA}(\mathbf{A})) + \lambda_1\|\text{PA}(\mathbf{A})\|_2^2 + \lambda_2\|\neg\text{PA}(\mathbf{A})\|_2^2 \tag{9}$$

Drawbacks: However, such method suffers two shortcomings we observed during our experiments:

(1) λ_1, λ_2 are hard to tune in practice. Sub-optimal values of these two parameters might change the scale of $\text{PA}(\mathbf{A})$ and $\neg\text{PA}(\mathbf{A})$, which might drift from the scale of actual latent states $\mathbf{Z}$.

(2) we need to run such optimization procedure for each evaluated observation in $\mathbf{X}$. This is not very realistic in our imitation learning tasks which usually contains 100 evaluation episodes, with each episode contains 500-1000 observations.

Our Solution: Given a new observation $\mathbf{X}_{\text{new}}$, another way we have considered is to choose the latent states in our training data whose corresponding observation $\mathbf{X}$ is closest to $\mathbf{X}_{\text{new}}$. However, this method also can't work well in practice. Therefore, we use a more brute force approach to learn a new encoder (neural networks) that maps observations $\mathbf{X}$ to latent states $\mathbf{Z}$.

5 EXPERIMENTS

In this section, we perform experiments on three commonly used imitation learning control tasks from OpenAi Gym [10]: Acrobot [14], Cartpole [6] and LunarLander [10] to test the effectiveness of our algorithm. All of them have continuous state space and discrete action space. Dataset descriptions could be seen in Table 1.

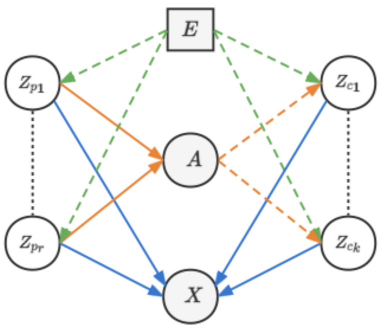
Dataset Description: We use Deep-Q Network (DQN) to train the expert policies for each control task. Expert scores together with scores from random actions are listed in Table 1. After training the expert agent, we generate 2000 trajectories of expert demonstrations, with each trajectory contains up to 500 state-action pairs.

Table 1: OpenAI Gym Datasets Descriptions.

	Observation Space	**Action Space**	**Random Score**	**Expert Score**
Acrobot-v1	6 (Continuous)	3 (Discrete)	-439.92 ± 13.14	-87.32 ± 12.02
CartPole-v1	4 (Continuous)	2 (Discrete)	19.12 ± 1.76	500.00 ± 0.00
LunarLander-v2	8 (Continuous)	4 (Discrete)	-452.22 ± 61.24	271.71 ± 17.88

Incorporate Spurious Correlates: We concatenate four additional spurious correlated states to the original states. The first three spurious states are generated from linear combinations of the original states with different multiplicative factors ($1\times$ and $2\times$ for the first and second environment respectively), and we use an environment identifier as the last spurious state. The right subplot in Fig. 2 plots the causal graph for the CartPole task. If we denote the four original observation states could be represented by $Z_1, ..., Z_4$, then we generate three additional states Z_5, Z_6, Z_7 by linear transformations of Z_2 to Z_4. Specifically, in environment #0, $Z_5 = Z_2, Z_6 = Z_3, Z_7 = Z_4$, and we set Z_4 as 0 to represent the environment indicator; and in environment #1, we set $Z_5 = 2Z_2+Z_3+Z_4$,

$Z_6 = Z_2 + 2Z_3 + Z_4$, $Z_7 = Z_2 + Z_3 + 2Z_4$, and $Z_8 = 1$. In other words, the values of Z_5 to Z_8 are environment-dependent, and spuriously correlated with the true states.



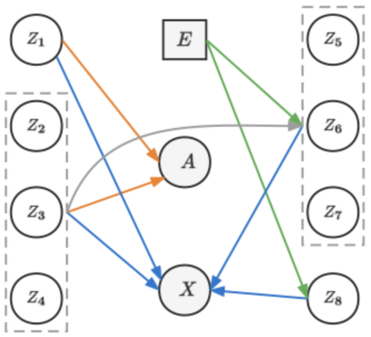


Figure 2: **Left:** The causal graph in Lu et al.[22]. Grey dashed lines represent arbitrary connections between the latent variables, Dashed green and orange arrow lines denote the edges which might vary across environments and might be abasent. **Right:** Causal graph for our CartPole task. To keep clean of the causal graph, the latent variables in the dashed rectangles have the same in/out edges to other variables. And the grey solid line between two rectangles means that Z_5 to Z_7 (spurious correlated latent states) are generated from linear combinations of the original latent states Z_2 to Z_4. The environment also has influence on Z_5 to Z_7.

Comparison of performance with & without spurious correlates: To show that Behavioral Cloning suffers from nuisance correlates, we run BC on both the original tasks as well as the tasks after our manipulation as mentioned above. Figure 3 shows the rewards of Behavioral Cloning strategy under (1) original state space and (2) augmented state space with spurious correlates. The conclusion here is clear: across these tasks, learning a policy from environment-dependent states leads to inferior performance.

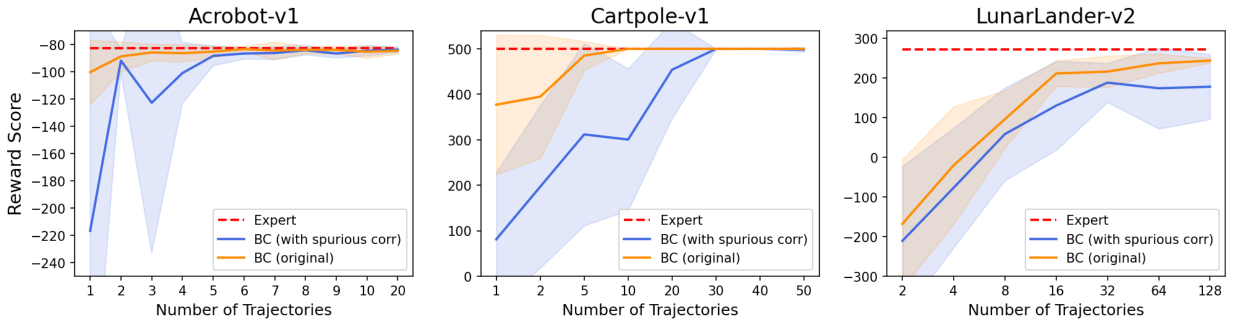


Figure 3: Reward (y-axis) vs number of training trajectories from expert demonstrations (x-axis) for dataset before (orange line) and after (blue line) adding spurious correlations. Expert scores are marked as red dashed lines. The shaded areas represent 0.5*standard deviation. Results are averaged over 15 runs.

Main Results: Our main results are presented in Fig. 4, where we compare several methods including BC, BC+IRMv1, CCIL and our IBC algorithm. The x-axis has increasing number of expert trajectories (amount of observed data). It is more difficult for the imitator to learn a high score when the amount of observed data. We conclude our findings in the following points:

- Our IBC algorithm is the most stable algorithm across all three tasks. It achieves the highest rewards for Cartpole, the second highest for Acrobot, and is within top two for LunarLander.
- CCIL exibits some interesting behavior. It is very effective for Acrobot, with rewards approximately equal to the expert score. However, its performance decreases pretty fast as the difficulty level of our tasks increases. We guess it is most useful for easy tasks.

- We could have gains by adding additional IRMv1 loss on BC when the tasks are not very difficult.
- Finally, BC is actually not that bad compared with other algorithm. It is still very effective for difficult tasks (e.g. LunarLander).

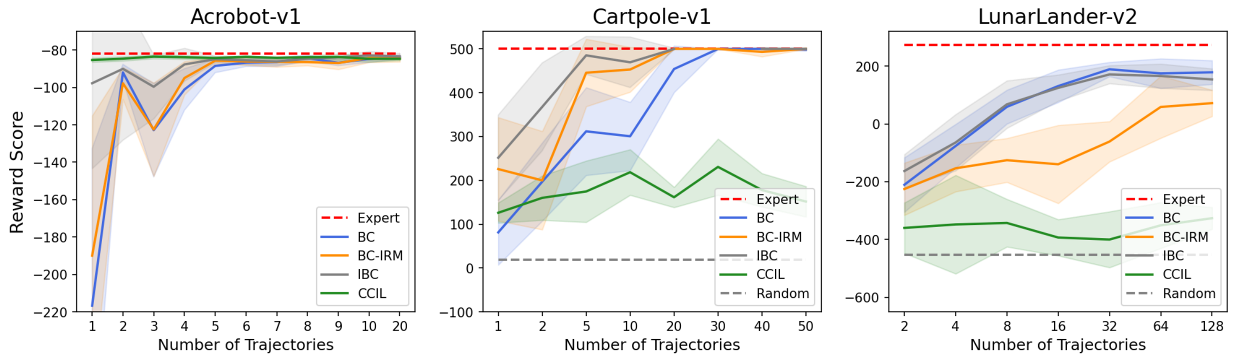


Figure 4: Reward (y-axis) vs number of training trajectories from expert demonstrations (x-axis) for different methods. Expert scores and random scores are marked as red and grey dashed lines respectively. The shaded areas represent 0.25*standard deviation. Results are averaged over 15 runs.

6 ABLATION TESTS

From the above experimental analysis, we see the effectiveness of our proposed IBC algorithm. Since our IBC algorithm contains three phases, it is natural to ask which phase is most crucial. Therefore, we make some ablation tests in this section by changing the priors in phase 1, and choose different parent selection methods in phase 2.

6.1 Phase 1 Ablation: Different Assumption on the Priors

To test the effectiveness of non-factorized prior, we first replace NF-iVAE with iVAE[3] in phase1. The assumptions for these two priors are listed in Table 2. iVAE assumes a stricter assumption. We train both variants on Cartpole with number of expert demonstration trajectories equal to 5. The training loss are the loss values we get from phase 3 on the training dataset, and the testing scores are evaluated on testing dataset. The results in Table 2 are averaged over 10 independent runs.

Table 2: Comparison of NF-iVAE and VAE used in Phase 1

	Assumption on the Prior $p_{\mathbf{T},\lambda}(\mathbf{Z}\vert\mathbf{A},\mathbf{E})$	**Training Loss**	**Testing Scores**
iVAE	$\prod_i \mathcal{Q}_i(Z_i)/\mathcal{Z}_i(\mathbf{A},\mathbf{E})\exp[\sum_{j=1}^{k} T_{i,j}(Z_i)\lambda_{i,j}(\mathbf{A},\mathbf{E})]$	0.0491 ± 0.0041	39.22 ± 47.75
NF-iVAE	$\mathcal{Q}(\mathbf{Z})/\mathcal{Z}(\mathbf{A},\mathbf{E})\exp[T(\mathbf{Z})^\top\lambda(\mathbf{A},\mathbf{E})]$	$\mathbf{0.0073 \pm 0.0091}$	$\mathbf{401.31 \pm 165.38}$

As we could observe, NF-iVAE achieves much less training loss as well as higher testing scores. Therefore, we could draw a conclusion that non-factorized prior is indeed helpful at least for the OpenAi Gym tasks we tested on.

6.2 Phase 2 Ablation: Parent Selection Threshold

In phase 2, we set the number of latent states equal to the size of observation space at the beginning, because we are not given the information about how many observed states contains spurious correlates. Then we use FCIT to detect a percentage of them as direct parents of $\mathbf{A}$. We regard a

[3]Code for iVAE is from `https://github.com/ilkhem/icebeem/tree/master/models/ivae`

latent state $\mathbf{Z}_i$ as $\text{PA}(\mathbf{A})$ if the multiplicity corrected p-value is still significant. However, the significance level α which decides whether or not we should reject $\mathcal{H}_0$ is a hyper-parameter that needs to be carefully tuned, since it influence the number of latent variables selected as $\text{PA}(\mathbf{A})$.

Alternative Approach: An alternative way is not make corrections for multiplicity. We could just compare the MSEs of the decision trees trained using both $\mathbf{Z}_i, \mathbf{Z}_j$ and using $\mathbf{Z}_i$ only. Central to this approach is the assumption that $\mathbf{Z}_i \perp\!\!\!\perp \mathbf{Z}_j|\mathbf{A}$ if and only if the MSE of the decision tree trained using both $\mathbf{Z}_i, \mathbf{Z}_j$ is not smaller than the MSE of the one trained using $\mathbf{Z}_i$ only (we denote this event as $indep \in \{0, 1\}$). For a certain $\mathbf{Z}_i$, we need to make pairwise comparisons with all other latent variables $\mathbf{Z}_j, (j \neq i)$ and get $indep_j, (j \neq i)$. Then we compute $\sum_{j\neq i} indep_j$, which is an indicator for the likelihood of $\mathbf{Z}_i$ as $\text{PA}(\mathbf{A})$. The hyper-parameter we need to tune is the threshold T, above which we regard $\mathbf{Z}_i$ as $\text{PA}(\mathbf{A})$ if $\sum_{j\neq i} indep_j \geq T$ holds. Since the threshold here is chosen manually, it is tempting to do ablation tests over different thresholds.

Fig. 5 contains our ablation results on the CartPole task. The original non-spurious observed variables in this task is 4 as we described in Sec. 5. In the left subplot of Fig. 5, we first tests the influence of the choice of number of latent states. From the left subplot, we could see that our reward is highest when we set the number of latent states $\mathbf{Z}$ equal to the number of original observed states $\mathbf{X}$, which is consistent with our intuition. The variance of rewards doubles as we choose more latent spaces, but the median rewards (the red horizontal lines) are still quite high (with the only exception when number of latent states equal to 6). This result indicates that we need to bear larger variance if we have no information about the number of nuisance variables in the observations.

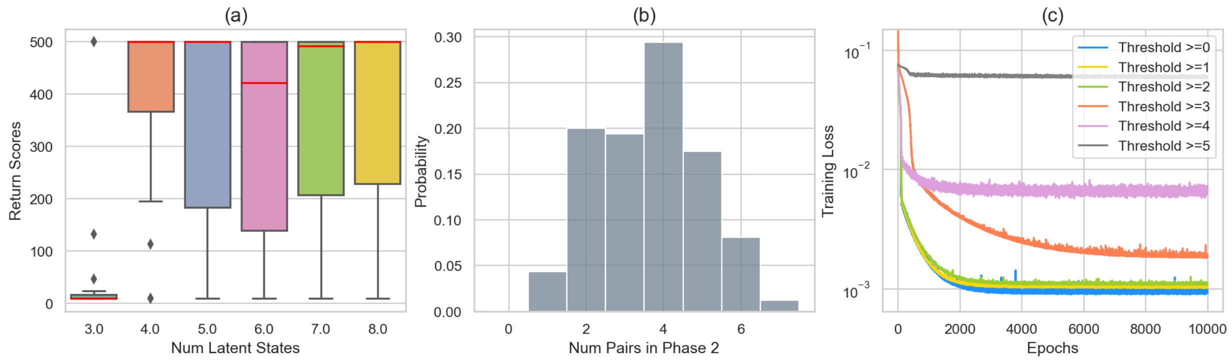


Figure 5: **Left:** Rewards with different number of latent states averaged over 10 runs (median rewards are marked red). **Middle:** Average number of pairs significant in the conditional independence test. **Right:** Training loss in phase3 with different threshold values.

How to Tune the Threshold? The next question is which threshold should we choose to decide whether a latent state is a $\text{PA}(\mathbf{A})$. One of the approach is to try several different thresholds, and observe their training loss in phase 3. If the training loss curve increase a lot when we increase the threshold by 1, then it probably means that there are not enough latent variables to predict the actions well. The right subplot in Fig. 5 shows that when we increase the threshold from 2 to 3, the training loss increases a lot, so a threshold of 2 is a good choice for this task.

7 CONCLUSIONS

In this report, we propose a new method, Invariant Behavioral Cloning, and show its effectiveness on three OpenAI Gym tasks. Ablation study on phase 1 indicates that NF-iVAE is indeed helpful. Ablation study on phase 2 discusses practical issues about how to detect direct parents of $\mathbf{A}$.

8 APPENDIX

To make the main body succinct and self-contained, we decide to separate the remaining 5 pages as the appendix. In this section, we illustrate in more details the drawbacks we found in each phase of the original iCaRL algorithm, as well as our adaptations to make it work better for our needs.

Personally speaking, I really enjoy the casual format of Dialogue and Discussion sections as shown in some papers we read in class (e.g. Arjovsky 2020 [2], Lu 2019 [22]), which helps me understand the problems better. Therefore, we decide to make some of the following subsections as Q&A format to better express our process of thinking. Hope you could enjoy!

8.1 Phase 1: Replace Score Matching with Sliced Score Matching (SSM)

8.1.1 Why Vanilla Score Matching Does Not Work Well?

Phase 1 loss in the original paper from Lu et al. [22] is defined as $\mathcal{L}_{\text{phase1}} = \mathcal{L}^{\text{VAE}}_{\text{phase1}} - \mathcal{L}^{\text{SM}}_{\text{phase1}}$. Such loss is optimized by jointly learn (θ, ϕ) from the following objective:

$$\mathcal{L}_{\text{phase1}}(\theta, \phi) = \mathcal{L}^{\text{VAE}}_{\text{phase1}}(f, \hat{T}, \hat{\lambda}, \phi) - \mathcal{L}^{\text{SM}}_{\text{phase1}}(\hat{f}, T, \lambda, \hat{\phi}) \tag{10}$$

where we use $\hat{f}, \hat{T}, \hat{\lambda}, \hat{\phi}$ to represent copies of f, t, λ, ϕ that are treated as constants and whose gradient is not calculated during training. The first term $\mathcal{L}^{\text{VAE}}_{\text{phase1}}$ on the right hand side has the form:

$$\mathcal{L}^{\text{VAE}}_{\text{phase1}}(f, \hat{T}, \hat{\lambda}, \phi) = \mathbb{E}_{p_D}[\mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})}[\log p_f(\mathbf{X}|\mathbf{Z}) + \log p_{\hat{T},\hat{\lambda}}(\mathbf{Z}|\mathbf{A},\mathbf{E}) - \log q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})]] \tag{11}$$

which is called Evidence Lower Bound (ELBO) commonly used in VAEs.

Since our model has unnormalized priors, we need to incorporate score matching [18] during the training process. The score matching loss in the original paper has the form:

$$\begin{aligned}\mathcal{L}^{\text{SM}}_{\text{phase1}} &= \mathbb{E}_{p_D}[\mathbb{E}_{q_{\hat{\phi}}(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})}[\|\nabla_{\mathbf{z}} \log q_{\hat{\phi}}(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E}) - \nabla_{\mathbf{z}} \log p_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E})\|^2]] &(12)\\ &= \mathbb{E}_{p_D}[\mathbb{E}_{q_{\hat{\phi}}(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})}[\sum_{j=1}^{n}[\frac{\partial^2_{p_{T,\lambda}}(\mathbf{Z}|\mathbf{A},\mathbf{E})}{\partial \mathbf{Z}_j^2} + \frac{1}{2}(\frac{\partial^2_{p_{T,\lambda}}(\mathbf{Z}|\mathbf{A},\mathbf{E})}{\partial \mathbf{Z}_j})^2]] &(13)\end{aligned}$$

This procedure looks reasonable in theory. However, the second derivatives in Eq. (6) here has very large variance caused by the oscillation of the norm of latent states Z_j's, which makes the training process not that stable. The drawback mentioned above is also observed by other researchers, which is caused by the difficulty of computing the Hessian of log-density functions. Specifically, methods including Denoising Score Matching (DSM) [32] and Sliced Score Matching (SSM) [29] are proposed to tackle this problem.

8.1.2 Stabilize Phase 1 Training with Sliced Score Matching (SSM)

Here, we adopt Sliced Score Matching (SSM) in our project to replace the original Score Matching. We choose SSM since it is one of the latest methods, and its performance is better than others as we found both from the empirical study in their paper, as well as experiments from our OpenAI Gym tasks. Let's first describes its empirical performance before delving into too much technical details.

Fig. 6 (referenced from Table 1 in [29]) compares SSM, and its variance reduction variant SSM-VR with other methods including Denoising Score Matching (DSM), curvature propagation (CP), and approximate backpropagation (approx BP) for NICE models on MNIST. As we can see, SSM outperforms other methods in terms of the score matching loss and log-likelihoods.

	Test SM Loss	Test LL
MLE	-579	**-791**
SSM-VR	**-8054**	-3355
SSM	-2428	-2039
DSM ($\sigma = 0.10$)	-3035	-4363
DSM ($\sigma = 1.74$)	-97	-8082
CP	-1694	-1517
Approx BP	-48	-2288

Figure 6: Score matching losses and log-likelihoods for NICE models on MNIST. $\sigma = 0.1$ is by grid search and $\sigma = 1.74$ is from the heuristic of Saremi et al. (2018).

With evidence from SSM paper, we test the performance of SSM on our CartPole OpenAi gym task together with the original Score Matching method[4]. We use the variance reduction variant (SSM-VR) since it achieves the best performance in Fig. 6. The specific form for SSM-VR estimator is given in Eq. (21) which is shown in the next subsection. We set the number of random projections in SSM-VR estimator (Eq. (21) below) to 10 ($M = 10$) in our experiment.

As we can see from Fig. 7, phase 1 training loss with SM oscillate a lot even training for 1500 epochs, and there's no trend to decrease. On the other hand, training loss with SSM finally stabilizes after around 500 epochs, which confirms the effectiveness of SSM.

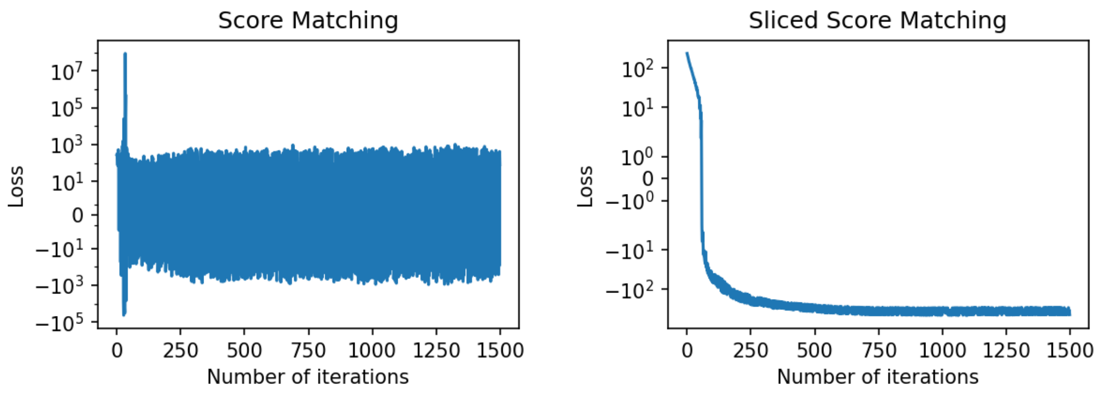


Figure 7: Phase 1 training loss for with Score Matching (SM) and Sliced Score Matching (SSM).

8.1.3 Technical Details of Sliced Score Matching (SSM)

In this section, we explain the formulation of Sliced Score Matching in detail. The formula for the SSM-VR estimator we used in our experiment is shown at the end of this subsection.

Let $p_{T,\lambda}(Z|A, E)$ be the normalized density determined by our model. We have:

$$p_{T,\lambda}(\mathbf{Z}|\mathbf{A}, \mathbf{E}) = \frac{\tilde{p}_{T,\lambda}(\mathbf{Z}|\mathbf{A}, \mathbf{E})}{N_{T,\lambda}} \tag{14}$$

[4]We use SSM code from this repo: `https://github.com/ermongroup/sliced_score_matching`

where $N_{T,\lambda}$ is the partition function.

For convenience, we denote the score functions of $p_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E})$ and $q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})$ as $s_m(\mathbf{Z};T,\lambda) = \nabla_\mathbf{z} \log p_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E})$ and $s_d(\mathbf{Z};\phi) = \nabla_\mathbf{z} \log q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})$ respectively. Note that since $\log p_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E}) = \log \tilde{p}_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E}) - \log N_{T,\lambda}$, we could conclude that $s_m(\mathbf{Z};T,\lambda)$ does not depend on the intractable partition function $N_{T,\lambda}$.

The original Score Matching method [18] minimizes the Fisher divergence between $p_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E})$ and $q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{A},\mathbf{E})$, which is defined as:

$$L(T,\lambda,\phi) = \frac{1}{2}\mathbb{E}_{p_{T,\lambda}}[\|s_m(\mathbf{Z};T,\lambda) - s_d(\mathbf{Z};\phi)\|_2^2] \tag{15}$$

By applying integration by parts, $L(T,\lambda,\phi)$ can be written as $L(T,\lambda,\phi) = J(T,\lambda,\phi) + C$, where C is a constant that does not depend on T, λ and ϕ.

$$J(T,\lambda,\phi) = \mathbb{E}_{p_{T,\lambda}}[tr(\nabla_\mathbf{z} s_m(\mathbf{Z};T,\lambda) + \frac{1}{2}\|s_m(\mathbf{Z};T,\lambda)\|_2^2)] \tag{16}$$

$tr(.)$ denotes the trace of a matrix, and

$$\nabla_\mathbf{z} s_m(\mathbf{Z};T,\lambda) = \nabla_\mathbf{z}^2 \log \tilde{p}_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E}) \tag{17}$$

is the Hessian of the log-density function. The constant C can be ignored and the following unbiased estimator of the remaining terms is used to train $\tilde{p}_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E})$:

$$\widehat{J}(T,\lambda;\mathbf{z}_1^N) = \frac{1}{N}\sum_{i=1}^{N}[tr(\nabla_\mathbf{z} s_m(\mathbf{z}_i;T,\lambda) + \frac{1}{2}\|s_m(\mathbf{z}_i;T,\lambda)\|_2^2)] \tag{18}$$

where $\mathbf{z}_1^N$ is the shorthand for the collection of latent variables $\{\mathbf{z}_1,\mathbf{z}_2,...,\mathbf{z}_N\}$ sampled i.i.d. from $p_{T,\lambda}$, and $\nabla_\mathbf{z} s_m(\mathbf{z}_i;T,\lambda)$ denotes the Hessian of $\log \tilde{p}_{T,\lambda}(\mathbf{Z}|\mathbf{A},\mathbf{E})$ evaluated at $\mathbf{z}_i$.

The intuition behind Sliced Score Matching comes from the observation that one dimensional problems are usually much easier to solve than high dimensional ones. Therefore, SSM projects $s_m(\mathbf{Z};T,\lambda)$ and $s_d(\mathbf{Z};\phi)$ onto some random direction $\mathbf{v}$ and propose to compare their average difference along that random direction. They consider the following objective as a replacement of the Fisher divergence $L(T,\lambda,\phi)$ in Eq. (15):

$$L(T,\lambda,\phi;p_\mathbf{v}) = \frac{1}{2}\mathbb{E}_{p_\mathbf{v}}\mathbb{E}_{p_\mathbf{d}}[(\mathbf{v}^\top s_m(\mathbf{Z};T,\lambda) - \mathbf{v}^\top s_d(\mathbf{Z};\phi))^2] \tag{19}$$

$$J(T,\lambda;p_\mathbf{v}) = \mathbb{E}_{p_\mathbf{v}}\mathbb{E}_{p_\mathbf{d}}[\mathbf{v}^\top \nabla_\mathbf{z} s_m(\mathbf{Z};T,\lambda)\mathbf{v} + \frac{1}{2}(\mathbf{v}^\top s_m(\mathbf{Z};T,\lambda))^2] \tag{20}$$

where $\mathbf{v} \sim p_\mathbf{v}$ and we require $\mathcal{E}_{p_\mathbf{v}}[\mathbf{v}^\top\mathbf{v}] > 0$ and $\mathcal{E}_{p_\mathbf{v}}[\|\mathbf{v}\|_2^2] < \infty$.

Specifically, if $p_\mathbf{v}$ is one of multivariate standard normal $\mathcal{N}(0,I_D)$ or the uniform distribution over $\{\pm 1\}^D$, we have $\mathbb{E}_{p_\mathbf{v}}[(\mathbf{v}^\top s_m(\mathbf{Z};T,\lambda))^2] = \|s_m(\mathbf{Z};T,\lambda)\|_2^2$, so the second term of Eq. (20) can be integrated analytically, which leads to the following estimator:

$$\widehat{J}_{\text{SSM-VR}}(T,\lambda;\mathbf{z}_1^N,\mathbf{v}_{11}^{\text{NM}}) = \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M}\mathbf{v}_{ij}^\top \nabla_\mathbf{z} s_m(\mathbf{z}_i;T,\lambda)\mathbf{v}_{ij} + \frac{1}{2}\|s_m(\mathbf{Z};T,\lambda)\|_2^2 \tag{21}$$

$\widehat{J}_{\text{SSM-VR}}$ is the estimator we used in phase 1 to replace the original Score Matching method.

8.2 Phase 2: Replace HSIC and KCIT with FCIT

8.2.1 Why Not Use HSIC and KCIT as in the Original Paper?

In phase 2 of Lu et al. [22], they use independence testing and conditional independence testing between all pairs of latent variables $(\mathbf{Z}_i, \mathbf{Z}_j), (i \neq j)$ and compare their p-values to determine if $\mathbf{Z}_i$ and $\mathbf{Z}_j$ can be potential candidates as direct parents of $\mathbf{A}$. In their paper, they use HSIC [15] and KCIT [35] for (conditional) independence tests. However, during our implementation, we found that these two methods needs to construct $n \times n$ matrix explicitly (n represents the sample size), and the running speed is too slow to make pairwise comparisons between all latent variables.

In general, there's no specific reason to use HSIC and KCIT, and the choice of (conditional) independence test methods could be problem specific. Ideally, we want methods with the following properties:

- In our OpenAI Gym tasks, the observational variables $\mathbf{X}$ are continuous, while the actions $\mathbf{A}$ are discrete (see Table. 1). But there also exist abundant imitation learning tasks with mixed continuous & discrete observations and actions. We would like to choose a method that is very flexible towards different data types.
- Imitation Learning tasks are usually data intensive. For example, each trajectory in our experiments (x-axis in Fig. 4) contains up to 500 state-action pairs $\{x_i, a_i\}$. So the sample size is around $10^4 \sim 10^5$ in our case. It could be even larger on more complicated control tasks. Since we need to compare each pair of latent variables, which is quadratic in the number of pairs, the running clocktime of the testing method is also an important concern in practice.
- Of course, it would be even better if we could find a method that satisfies the first two bullet points, as well as achieving lower estimation errors (Type 1 & 2 errors).

With the above three objectives, we found that FCIT is the best fit:

- It works well on continuous and discrete/categorical data.
- It's **fast**! It can process hundred-dimensional conditional independence queries with 10^5 samples in less than 60s, which is much faster than HSIC and KCIT.
- It achieves lowest Type 1 and Type 2 errors compared with alternative methods including CHSIC, KCIT, KCIPT, RCIT, and CCI on a wide range of datasets. (see Sec. 4 in [11]).

8.2.2 Technical Details of Fast Conditional Independence Test

The intuition behind FCIT is that if $\mathbf{Z_i} \not\perp\!\!\!\perp \mathbf{Z_j} | \mathbf{A}$, then prediction of $\mathbf{A}$ using both $\mathbf{Z_i}$ and $\mathbf{Z_j}$ as covariates should be more accurate than prediction of $\mathbf{A}$ when only one of $\mathbf{Z_i}, \mathbf{Z_j}$ is used.

It trains two decision trees f_1 and f_2, with the first maps $\mathbf{A} = f_1(\mathbf{Z}_i, \mathbf{Z}_j)$, and the second maps $\mathbf{A} = f_2(\mathbf{Z}_i)$. If we denote the MSEs of f_1 and f_2 as MSE1 and MSE2 respectively, then we could obtain the one-tailed p-value for the null hypothesis $\mathcal{H}_0$ that MSE1$\neq$MSE2 on average. We include its algorithm here for completeness.

8.3 Properties of Imitation Learning/Reinforcement Learning Dataset

In this section, we briefly explain the difference between dataset coming from imitation/reinforcement learning, and the dataset usually used in causal inference. As we mentioned in Sec. 3.2, our offline dataset $\mathcal{D} = \{\{(x_i, a_i)\}_{i=1}^{N^e} | e \in \mathcal{E}_{\text{train}}\}$ contains expert demonstrations from multiple environments. Each piece of expert demonstration is a trajectory that contains observation-action pairs $\{(x_t, a_t)\}|_{t=1}^{T}$ for a consecutive T steps. In each step t, the next state x_{t+1} a learning agent observes depends on the action a_t it takes. If the agent went to a failing state at step t (for example, the lunar lander hit the ground and crashed), then the trajectory for this learning agent would end. The testing score of a learning policy (as we used in Table 2) is usually represented as the average steps an agent could survive. Therefore, a major difference between imitation learning dataset and causal inference dataset is its time-dependency.

In our report, our proposed algorithm IBC does not utilize such time-dependency property. A future work direction would be to lean more toward imitation learning by using the information from its dynamics/trajectories. Incorporating the state coverage loss and prediction loss mentioned in [7] could be a good start.

8.4 Meaning of Multiple Environments

In our report, we create multiple environments by designing some latent states with spurious correlations to the original states. What does multiple environments mean in reality is still an important question. An example with more practical influence is self-driving. In this application, multiple environments could be the images gathered from sensors on the car under different streets, driving styles, and weathers, etc. A future work would be to test our algorithm on these self-driving dataset, which could benefit more to the society.

Here's the end of our report. We welcome any questions and advice. Thanks for your time!

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[2] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization, 2019.

[3] Y. Aytar, T. Pfaff, D. Budden, T. L. Paine, Z. Wang, and N. de Freitas. Playing hard exploration games by watching youtube. *CoRR*, abs/1805.11592, 2018.

[4] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*, page 103–129, GBR, 1999. Oxford University.

[5] M. Bansal, A. Krizhevsky, and A. S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018.

[6] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

[7] I. Bica, D. Jarrett, and M. van der Schaar. Invariant causal imitation learning for generalizable policies. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[8] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[9] K. Brantley, W. Sun, and M. Henaff. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*, 2020.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[11] K. Chalupka, P. Perona, and F. Eberhardt. Fast conditional independence test for vector variables with large sample sizes, 2018.

[12] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *CoRR*, abs/1904.08980, 2019.

[13] P. de Haan, D. Jayaraman, and S. Levine. Causal confusion in imitation learning. *CoRR*, abs/1905.11979, 2019.

[14] A. Geramifard, C. Dann, R. H. Klein, W. Dabney, and J. P. How. Rlpy: A value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16(46):1573–1578, 2015.

[15] A. Gretton, K. Fukumizu, C. Teo, L. Song, B. Schölkopf, and A. Smola. A kernel statistical test of independence. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[16] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. P. Agapiou, J. Z. Leibo, and A. Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

[17] J. Ho and S. Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016.

[18] A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.*, 6:695–709, 2005.

[19] D. Kumor, J. Zhang, and E. Bareinboim. Sequential causal imitation learning with unobserved confounders. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[20] H. M. Le, Y. Yue, and P. Carr. Coordinated multi-agent imitation learning. *CoRR*, abs/1703.03121, 2017.

[21] F. Locatello, S. Bauer, M. Lucic, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *CoRR*, abs/1811.12359, 2018.

[22] C. Lu, Y. Wu, J. M. Hernández-Lobato, and B. Schölkopf. Nonlinear invariant risk minimization: A causal approach. *CoRR*, abs/2102.12353, 2021.

[23] D. Margaritis. Distribution-free learning of bayesian network structure in continuous domains. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, page 825–830. AAAI Press, 2005.

[24] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[25] J. Park, Y. Seo, C. Liu, L. Zhao, T. Qin, J. Shin, and T. Liu. Object-aware regularization for addressing causal confusion in imitation learning. *CoRR*, abs/2110.14118, 2021.

[26] J. Peters, D. Janzing, and B. Scholkopf. Causal inference on discrete data using additive noise models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2436–2450, dec 2011.

[27] T. Pohlen, B. Piot, T. Hester, M. G. Azar, D. Horgan, D. Budden, G. Barth-Maron, H. van Hasselt, J. Quan, M. Vecerík, M. Hessel, R. Munos, and O. Pietquin. Observe and look further: Achieving consistent performance on atari. *CoRR*, abs/1805.11593, 2018.

[28] S. Schaal. Learning from demonstration. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

[29] Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced score matching: A scalable approach to density and score estimation. *CoRR*, abs/1905.07088, 2019.

[30] D. L. Streiner. Best (but oft-forgotten) practices: the multiple problems of multiplicity—whether and how to correct for many statistical tests. *The American Journal of Clinical Nutrition*, 102(4):721–728, 08 2015.

[31] X. Sun, B. Wu, C. Liu, X. Zheng, W. Chen, T. Qin, and T. Liu. Latent causal invariant model. *CoRR*, abs/2011.02203, 2020.

[32] P. Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[33] C. Wen, J. Lin, T. Darrell, D. Jayaraman, and Y. Gao. Fighting copycat agents in behavioral cloning from observation histories. In *NeurIPS*, 2020.

[34] J. Zhang, D. Kumor, and E. Bareinboim. Causal imitation learning with unobserved confounders. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12263–12274. Curran Associates, Inc., 2020.

[35] K. Zhang, J. Peters, D. Janzing, and B. Schölkopf. Kernel-based conditional independence test and application in causal discovery. *CoRR*, abs/1202.3775, 2012.

[36] Z. Šidák. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62(318):626–633, 1967.

Algorithm 1 Fast (conditional) Independence Test

```
def cross_validate(covarite, regressand):
    # Find the best decision tree hyperparameters for
    # regressing `regressand` on `covariate`.

    # Return a trainable decision tree with the best hyperparameters.


def fit_test(x, y, z, n_perm=8, frac_test=.1):
    # Store total sample size and test set size.
    n_samples = x.shape[0]
    n_test = floor(frac_test * n_samples)

    # Find best decision tree parameters for learning y = f(x, z).
    best_tree_x = cross_validate(concat(x, z), y)

    # Find the best decision tree parameters for learning y = f(z).
    best_tree_nox = cross_validate(z, y)

    # Obtain `n_perm` MSEs on random train-test splits.
    mses_x = list()
    mses_nox = list()

    # In our implementation, this loop is parallelized.
    for perm_id in range(n_perm):
        perm_ids = random.permutation(n_samples)
        x_test, x_train = x[perm_ids][:n_test], x[perm_ids][n_test:]
        y_test, y_train = y[perm_ids][:n_test], y[perm_ids][n_test:]
        z_test, z_train = z[perm_ids][:n_test], z[perm_ids][n_test:]

        # Train a decision tree to predict y using x and z.
        best_tree_x.train(concat(x_train, z_train), y_train)
        mses_x.append(
            mse(best_tree_x.predict(concat(x_test, z_test)), y_test))

        # Train a decision tree to predict y using only z.
        best_tree_nox.train(z_train, y_train)
        mses_nox.append(mse(best_tree_nox.predict(z_test), y_test))

    # Obtain the one-tailed p-value for the null hypothesis that
    # on average, mses_nox is the same as mses_x.
    t, pval = ttest_1samp(mses_nox - mses_x)
    if t < 0:
        return 1 - pval / 2.
    else:
        return pval / 2.
```

Figure 8: Algorithm for FCIT.