WHEN DOES PREDICTIVE INVERSE DYNAMICS OUTPERFORM BEHAVIOR CLONING? EXPLORING THE ROLE OF ACTION AND STATE UNCERTAINTY

Anonymous authors
Paper under double-blind review

ABSTRACT

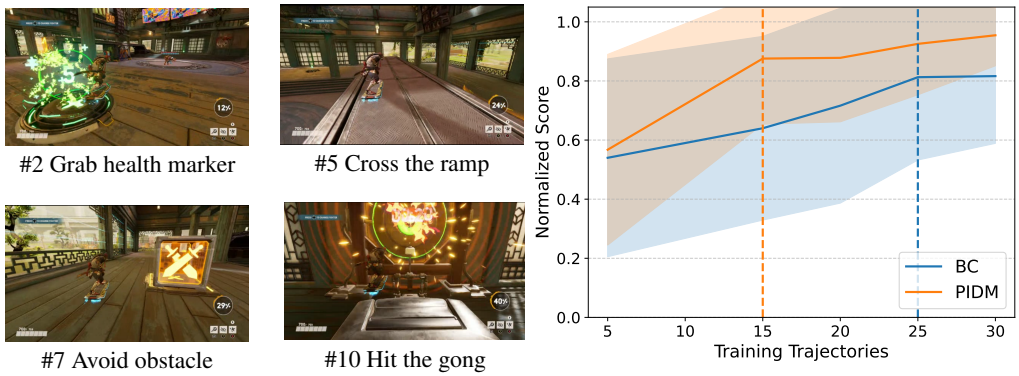
Offline imitation learning aims to train agents from demonstrations without interacting with the environment, but standard approaches like behavior cloning (BC) often fail when expert demonstrations are limited. Recent work has introduced a class of architectures we call predictive inverse dynamics models (PIDM), which combine a future state predictor with an inverse dynamics model to infer actions to reach the predicted future states. Although PIDM can be considered a form of behavioral cloning (in the sense of Bayes-optimality), it often outperforms conventional BC in practice. Although PIDM has shown promise, its benefits remain poorly understood. In this work, we analyze PIDM in the offline imitation learning setting and provide a theoretical explanation: under a perfect state predictor, the prediction error of PIDM can be lower than that of conventional BC, even in low-data regimes, and this gap increases when additional data sources can be leveraged. This efficiency gain is characterized by the variance of actions conditioned on future states, highlighting PIDM's ability to reduce uncertainty in states where future context is informative. We further demonstrate how this uncertainty reduction translates into sample efficiency improvements. We validate these insights empirically under more general conditions in 2D navigation tasks using human demonstrations, where BC requires on average 2.8 more samples than PIDM to reach comparable performance. Finally, we extend our evaluation to a complex 3D environment in a modern video game with high-dimensional visual inputs, and stochastic transitions, where BC requires over 66% more samples than PIDM in a realistic setting.

1 INTRODUCTION

Offline imitation learning aims to learn closed-loop control policies that replicate expert behavior using only pre-collected data, without access to a reward function or further interaction with the environment. This paradigm has broad applicability across domains such as robotics (Schaal, 1999; Fang et al., 2019), autonomous driving (Pan et al., 2020), and gaming (Pearce & Zhu, 2022; Pearce et al., 2023; Schäfer et al., 2023). A prominent line of research in imitation learning focuses on one- or few-shot generalization, where models are pretrained on large-scale datasets spanning diverse tasks (Duan et al., 2017), with the goal of adapting to new tasks from only a handful of demonstrations. However, collecting such large-scale expert demonstrations is often costly, time-consuming, or infeasible—particularly in specialized domains like robotics, where data acquisition is expensive and task-specific. As a result, many real-world applications lack the scale of data required to train or adapt large models using standard imitation learning techniques.

In contrast to approaches that rely on extensive pretraining, we focus on the low-data regime, where only a few demonstrations are available for the target task, and no additional data can be assumed. This setting is increasingly relevant in the current AI landscape, where large foundation models are trained on massive datasets, yet aligning them to new domains with limited supervision remains a significant challenge.

The most common offline imitation learning approach is behavior cloning (BC) (see Figure 2a), which can exhibit complex behavior (Osa et al., 2018; Pearce & Zhu, 2022; Florence et al., 2022) but



(a) Selected milestones of "Tour" task in 3D world

(b) Sample efficiency

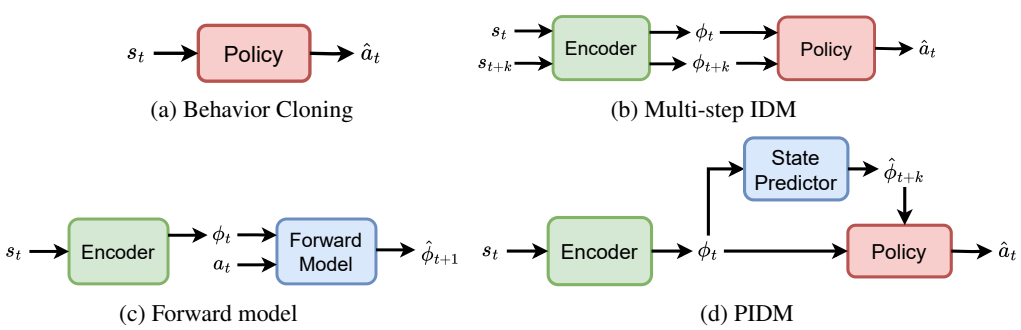
Figure 1: (a) Visualization of selected milestones from the complex "Tour" task in a 3D world with video input, stochastic transitions, and real-time inference. (b) Sample efficiency curves (mean $\pm$ std) for PIDM and BC. BC requires 66% more samples to achieve 80% success rate on average.

typically relies on the availability of many demonstrations per task. Recent work has introduced a promising alternative to BC, which we refer to as predictive inverse dynamics models (PIDMs) (Du et al., 2023; Xie et al., 2025; Tian et al., 2025). PIDM integrates two components: a state predictor, which forecasts plausible future states, and an inverse dynamics model (IDM), which infers the actions needed to reach those states (see Figure 2d). This modular design offers a key advantage—it allows leveraging diverse data sources, including action-free demonstrations and non-expert data. By augmenting a small set of expert demonstrations with such additional data, PIDM has demonstrated strong empirical performance (Xie et al., 2025). Interestingly, Xie et al. (2025) also reported that PIDM can significantly improve upon BC even when no additional data sources were available, suggesting the promise of PIDM for the low-data regime. However, the underlying reasons for their sample efficiency remain unclear. Is there something intrinsic to the PIDM architecture that enables this advantage? Under what conditions can we expect such gains to consistently emerge?

In this work, we analyze PIDM and provide theoretical insights into why decomposing the decision-making problem into a state predictor and an IDM can lead to significant sample efficiency improvements over BC. Specifically, PIDM can achieve comparable or superior performance using fewer expert demonstrations. First, we show that the prediction error of an optimal estimator for PIDM is always less than or equal to that of BC, resulting in a non-negative performance gap in favor of PIDM even in the small data regime. This gap is characterized by the expected conditional variance of actions given all possible future states, averaged over the current state distribution. Under additional assumptions, we show that the uncertainty reduction can provide sample efficiency gains.

Second, we provide empirical evidence that the predicted sample efficiency gains apply to more general conditions, including the small data regime, with no additional data sources, and general modeling approaches, like neural networks. We perform experiments on a benchmark of four 2D navigation tasks in a state-based environment, using a dataset of human demonstrations, and observe that BC requires between $1.3\times$ and $4\times$ more demonstrations than PIDM. The simplicity of the environment allows us to understand how the theory works in practice by looking at the prediction error gap per state. It also allows us to isolate the efficiency gains due to the predicted error gap from the representational benefits of IDM shown in previous work (Lamb et al., 2023; Koul et al., 2023; Levine et al., 2024; Islam et al., 2022).

Finally, once we have built intuition as to *why* the PIDM decomposition is effective, we extend our investigation to complex tasks that require imitating complex navigation tasks, from image inputs, in a 3D world with stochastic transitions, in real-time using human demonstrations. In this real-world setting, sample efficiency is critical since obtaining human demonstrations is costly, and real-time requirements introduce additional constraints on the solution. In this setting, we continue to observe substantial efficiency gains — BC requires 66% more samples than PIDM —, demonstrating that the predicted performance gap is relevant for real-world applications.



(a) Behavior Cloning (b) Multi-step IDM

(c) Forward model (d) PIDM

Figure 2: (a) BC learns a policy conditioned on the current state. (b) IDM learns a policy conditioned on the current and future state k steps ahead. (c) Forward models predict a future state (representation) given a state and action. Both (b) IDM and (c) forward models can serve as auxiliary objectives to learn effective state representations. (d) PIDM represents an alternative to BC consisting of a state predictor, akin to an action-free forward model, that predicts future state representations, and an IDM policy. The state encoder alleviates the dependence on ground-truth future states at evaluation time.

2 Related Work

Inverse dynamics models. Inverse dynamics models (IDMs) predict the initial action that initiates a sequence leading to a transition from the current state to a future state k steps ahead. As a direct *imitation learning* mechanism, multi-step inverse objectives are commonly used to train policies or transferable encoders on high-dimensional observations; the inverse loss filters out exogenous factors (Mhammedi et al., 2023; Efroni et al., 2022; Lamb et al., 2023) and learns rich state representations which can later support policy learning or be reused across tasks. To enable efficient training, we focus on architectures, in which states are encoded into a latent space with a state encoder (see Figure 2b). Both the state predictor and the IDM policy operate in this latent space.

Forward models. Forward models (see Figure 2c) can be added as auxiliary objectives to improve learned representations (Levine et al., 2024). Alternatively, a forward model can be learned for planning purposes, e.g. using reinforcement learning (Thrun et al., 1990; Hafner et al., 2025) or model-predictive control (Zhou et al., 2024; Bar et al., 2025). They are different from the state predictor of PIDM in two ways. First, forward models require action input, while the state predictor is conditioned on the current state only. Second, forward models usually generate the next state, while the state predictor generates future states $k \geq 1$ steps ahead.

Predictive inverse dynamics models. Recent works have combined a state predictor with an IDM to learn generalizable policies. Inspired by diffusion models for video generation, Du et al. (2023) trained a diffusion model to predict future images conditioned on task descriptions, operating directly in image space. To simplify learning, Xie et al. (2025) proposed using compact image representations, enabling state predictors to train action-free demonstrations and IDMs on diverse, action-labeled trajectories. Tian et al. (2025) proposed end-to-end training, using the IDM objective to guide the state predictor. These approaches showed PIDM outperforms BC and other baselines. Our work provides theoretical and empirical insight into why decomposing future state and action prediction leads to this performance gain.

Behavior cloning and trajectory modeling. Recent analysis (Foster et al., 2024) argues that many practical implementations of BC, which rely on a log-loss, are implicitly modeling the whole state-action sequence. Our work complements such analysis by providing evidence that explicitly modeling (part of) the trajectory, the future state in the PIDM case, can improve sample efficiency.

3 Preliminaries

Problem setting. We consider the problem setting of an MDP defined by $(\mathbb{S}, \mathbb{A}, \mathcal{T}, \mathcal{R}, d_0)$ of state space $\mathbb{S}$; action space $\mathbb{A}$; transition function $\mathcal{T} : \mathbb{S} \times \mathbb{A} \mapsto \mathcal{P}(\mathbb{S})$, where $\mathcal{P}(\cdot)$ is a probability measure; reward function $\mathcal{R} : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{R}$; and initial state distribution d_0. To interact with the MDP, we first sample an initial state $s_0 \sim d_0$. Then, at each time step, we sample an action $a_t \sim \pi(\cdot \mid s_t)$

from the policy $\pi : \mathbb{S} \mapsto \mathcal{P}(\mathbb{A})$. Given this action, the environment transitions to a new state $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$ and provides a reward $r = \mathcal{R}(s_t, a_t)$. For this work, we assume no access to the reward signal and consider the offline imitation learning setting in which we are given a dataset of N trajectories of states and actions $\mathbb{D} = \{(s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T\}_{i=1}^{N}$ generated by following some unknown expert policy π^*. Our goal is to learn a policy π that is as close as possible to π^*. We consider two architectures, BC and PIDM.

BC treats offline imitation learning as a supervised learning problem and trains a policy to imitate the actions in the dataset given the most recent state. It consists of a single block, the policy (see Figure 2a), which can be trained to minimize the following loss:

$$\mathcal{L}_{\text{BC}}(\pi_\mu) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \mathbb{D}, \hat{\boldsymbol{a}}_t \sim \pi_\mu(\cdot \mid \boldsymbol{s}_t)} \left[d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t) \right] \tag{1}$$

for some measure of dissimilarity, d, between the action distribution induced by the learned policy π_μ and the ground truth actions under the dataset distribution. There are multiple design choices on how BC approximates the policy distribution that offer different fidelity and complexity tradeoffs, ranging from simple but effective point estimates to rich but complex generative models that can capture distributions with multiple modes.

PIDM consists of two submodels (see Figure 2d): a state predictor, p, that predicts future states for some horizon k, and an inverse dynamics model (IDM) policy, π_ξ, that predicts the next action needed to get from the current observation to the future observation in k steps. Similar to BC, there are multiple design choices. In addition to how to approximate the distributions, the two submodels can be trained from the same or different datasets to leverage additional data sources. We focus on the case where they are trained with the same dataset, using the following losses:

$$\mathcal{L}_{\text{SP}}(p) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{s}_{t+k}) \sim \mathbb{D}, \hat{\boldsymbol{s}}_{t+k} \sim p(\cdot \mid \boldsymbol{s}_t)} \left[d(\hat{\boldsymbol{s}}_{t+k}, \boldsymbol{s}_{t+k}) \right], \tag{2}$$

$$\mathcal{L}_{\text{IDM}}(\pi_\xi) = \mathbb{E}_{(s_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+k}) \sim \mathbb{D}, \hat{\boldsymbol{a}}_t \sim \pi_\xi(\cdot \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})} \left[d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t) \right]. \tag{3}$$

4 Theoretical Analysis

The PIDM approach can be seen as a decomposition of BC with explicit modeling of future states:

$$\pi_\mu(a_t \mid s_t) = \int_{\mathbb{S}} p(s_{t+k} \mid s_t) \pi_\xi(a_t \mid s_t, s_{t+k}) ds_{t+k}. \tag{4}$$

Intuitively, this decomposition can simplify the learning of a policy whenever the conditioning on the future state in the IDM policy provides useful information to identify which action to take. In this section, we study some potential gains of PIDM over BC when we assume access to a state predictor. All proofs are in Appendix A.

For simplicity, we consider the case where the BC and IDM policies are single-point estimators that approximate the expected action. Let $\overline{\mu}(\boldsymbol{s}_t) \triangleq \mathbb{E}[\boldsymbol{a}_t \mid \boldsymbol{s}_t]$ and $\overline{\xi}(\boldsymbol{s}_t, \boldsymbol{s}_{t+k}) \triangleq \mathbb{E}[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k}]$ be the optimal estimators for π_μ and π_ξ, respectively.

Introduce the predicted error gap between the estimators of the BC and IDM policies:

$$\Delta \triangleq \text{EPE}(\overline{\mu}) - \text{EPE}(\overline{\xi}), \tag{5}$$

where $\text{EPE}(\cdot)$ is the expected prediction error, which for a random variable $\boldsymbol{y}|\boldsymbol{x}$ and an estimator $\zeta(\boldsymbol{x})$ is given by: $\text{EPE}(\zeta) \triangleq \mathbb{E}_{\boldsymbol{x}} \left[(\boldsymbol{y} - \zeta(\boldsymbol{x}))^2 \right]$.

Our first result quantifies Δ in terms of the uncertainty in $\boldsymbol{a}_t$ due to uncertainty in $\boldsymbol{s}_{t+k}$.

Theorem 1. *For optimal estimators $\overline{\mu}$ and $\overline{\xi}$, The predicted error gap is given by:*

$$\Delta = \mathbb{E}_{\boldsymbol{s}_t} \left[\text{Var}_{\boldsymbol{s}_{t+k} \mid \boldsymbol{s}_t} \left(\mathbb{E} \left[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k} \right] \right) \right] \geq 0. \tag{6}$$

Theorem 1 shows that knowing $\boldsymbol{s}_{t+k}$ can increase the prediction accuracy of $\boldsymbol{a}_t$. However, this improvement assumes access to a state predictor model. Hence, it can be seen as the ideal case that upper bounds the potential gain that can be achieved when the state predictor model has to be learned.

We can also derive the predicted error gap under the assumption that the estimators are unbiased, though not necessarily optimal.

Corollary 1. *Let $\widehat{\mu}(\boldsymbol{s}_t)$ and $\widehat{\xi}(\boldsymbol{s}_t, \boldsymbol{s}_{t+k})$ be unbiased estimators for π_μ and π_ξ, respectively. Let $\widehat{\delta}$ denote the difference in the estimators' own variance:*

$$\widehat{\delta} \triangleq \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}\left(\widehat{\mu}\left(\boldsymbol{s}_t\right)\right)\right] - \mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{s}_{t+k}}\left[\mathrm{Var}\left(\widehat{\xi}\left(\boldsymbol{s}_t, \boldsymbol{s}_{t+k}\right)\right)\right]. \tag{7}$$

Then, we have:

$$\widehat{\Delta} \triangleq \mathrm{EPE}\left(\widehat{\mu}\right) - \mathrm{EPE}\left(\widehat{\xi}\right) = \Delta + \widehat{\delta}. \tag{8}$$

The first term in (8), Δ, is the reduction in variance due to knowing $\boldsymbol{s}_{t+k}$, as given by (6), while $\widehat{\delta}$ compares the model uncertainties.

Note that both Theorem 1 and Corollary 1 naturally motivates the use additional data sources. The former shows that action-free data can be used to train a more accurate state predictor model that closes the distance to an upper bound. The latter shows that demonstrations for different tasks in the same environment — or even non-optimal demonstrations when $k = 1$ — can be used to reduce the variance of $\widehat{\xi}$, increasing $\widehat{\delta}$, which leads to a larger gap $\widehat{\Delta}$.

The next result connects the prediction error gap with sample efficiency gains. We assume asymptotic efficiency for simplicity, which implies that the variance of the estimator decreases approximately linearly with the number of samples, and can be expressed for any unbiased estimator $\widehat{\zeta}$ of some parameter ζ as: $\mathrm{Var}(\widehat{\zeta}) \approx \frac{C_\zeta}{n}$ for large enough n, where $C_\zeta > 0$ is the inverse of the Fisher information for the ζ parameter being estimated.

Theorem 2. *Let $\widehat{\mu}_n(\boldsymbol{s}_t)$ and $\widehat{\xi}_m(\boldsymbol{s}_t, \boldsymbol{s}_{t+k})$ be unbiased, asymptotically efficient estimators for the BC and IDM policies, respectively, where n and m denote the minimum number of samples required to achieve error level ε. Then, for large enough n and m, we have:*

$$\eta \triangleq \frac{n}{m} \approx \frac{C_\mu}{C_\xi}\left(1 + \frac{\Delta}{\varepsilon - \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]}\right) \geq 0. \tag{9}$$

The theoretical results of this section provide insights on the potential gains that the PIDM architecture can provide. Although they have been derived under simplifying assumptions, Section 5 provides empirical evidence that efficiency gains hold in practice, under general conditions.

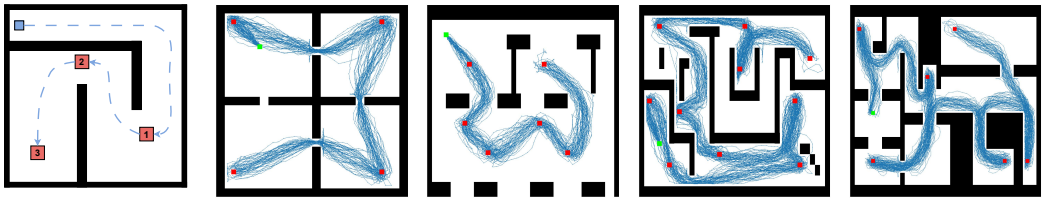
5 Experiments

To better understand how the efficiency gains predicted in Section 4 manifest in practice, we perform experiments in a 2D navigation environment, where we can easily analyze the properties of datasets and policies. We then conduct experiments in a 3D world that require precise execution of a complex task from images to validate our findings under real-world conditions.

5.1 Environments

2D navigation environment. We consider four tasks of varying complexity within a 2D navigation environment, visualized in Figure 3, in which the agent needs to reach a sequence of goals. The tasks are fully observable with low-dimensional states containing the x- and y-position of the agent as well as the positions of all goals, and whether they have already been reached. This simplified setting allow us to study the efficiency gains of PIDM over BC due to its action decomposition, isolated from other gains resulting from improved representations reported in prior work (Lamb et al., 2023; Koul et al., 2023; Levine et al., 2024). The agent chooses actions in $[-1, 1]^2$ for its movement, and the transitions are stochastic with Gaussian noise $\mathcal{N}(0, 0.2)$ added to the actions. For each task, a human player collected a dataset of 50 trajectories by navigating the agent to reach all goals using a controller. The datasets naturally contain some variability in actions in any given state as visualized by the human trajectories shown in Figure 3. We refer to Appendix B for more details on each dataset.

3D world. For a complex environment under real-world conditions, we constructed a dataset comprising human gameplay demonstrations within a modern 3D video game titled "Bleeding Edge", developed by Ninja Theory. The environment corresponds to the "Dojo" practice level. It features a third-person perspective with a freely controllable camera, where the camera orientation



(a) 2D Navigation task (b) Four room (c) Zigzag (d) Maze (e) Multiroom

Figure 3: Visualization of 2D navigation environment. (a) Tasks require the agent (blue box) to navigate to reach the goals (red boxes) in a particular order. (b) - (e) Visualizations of all four tasks and the traces of the 50 trajectories within the datasets.

directly affects the agent's movement direction, introducing a non-trivial perception-action coupling. Observations are captured as raw video frames, which are processed through a pre-trained image encoder to obtain embeddings that facilitate efficient learning. These embeddings are subsequently passed to the networks of the algorithms. In our experiments, we use the pre-trained ViT-B/16 Theia vision encoder (Shang et al., 2024). The action space contains continuous actions $[-1, 1]^4$ to control the x- and y-movement of the controlled character and the camera. State transitions occur asynchronously at 30 FPS and require real-time inference. Due to the game's deployment on a remote server in a distant cloud region, transitions are stochastic, affected by variable latency and visual artifacts. Within the environment we consider a task we refer to as "Tour" that consists of $\sim$36 seconds of precise navigation with 11 milestones, testing the agents' capability to steer and stay on track while avoiding obstacles and reacting at objects of interests (see Figure 1a for visualization of some milestones and Appendix C.2 for the complete list).

5.2 Algorithms

Model architecture. In the 2D navigation environment with fully observable states, we train MLP networks for the encoder and policy networks of BC and PIDM, and use $k = 1$ for PIDM. In contrast, the complex 3D world task is partially observable with inputs being video frames that are first being processed by a pre-trained vision encoder. The policy then receives a stack of vision encoder embeddings for three frames spanning one second to approximate a single state. BC and PIDM policies are then conditioned on these stacked representations for the current state and, in the case of PIDM, for the future state. To leverage the representational benefits of multi-step IDM (Lamb et al., 2023; Koul et al., 2023), we train the PIDM policy using $k \in \{1, 6, 11, 16, 21, 26\}$ for this task and additionally condition the PIDM policy network on a one-hot encoding of k. During evaluation, we query the PIDM policy and state predictor with $k = 1$. For BC and PIDM, we use the $tanh$ activation function on the action logits to get actions in the desired $[-1, 1]$ range.

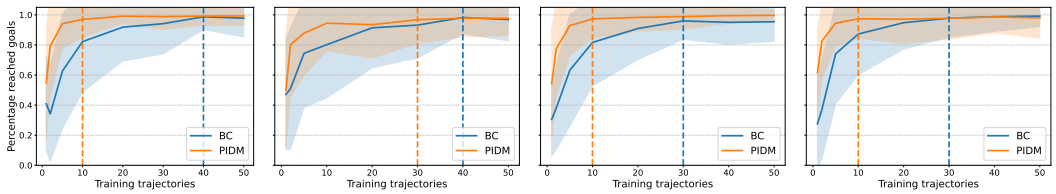
State predictor. We consider two state predictors. In the 2D navigation environment, we leverage an instance-based learning model (Keogh, 2010) for a deterministic state predictor:

$$p(s_t) = s^{i^\star}_{\tau^\star + k} \quad \text{with} \quad (\tau^\star, i^\star) \triangleq \arg\min_{\tau, i} ||s_t - s^i_\tau||^2, \tag{10}$$

with s^i_τ referring to the state at time step τ in training trajectory i. In short, the state predictor first queries for the nearest state within any training trajectory, as measured by the Euclidean distance, and then predicts the state k steps ahead of that state within the same training trajectory. In the 2D navigation environment, we further constrain the query for the nearest state to only match states in which the same goal needs to currently be reached. Computation of this lookup is efficient for the small data regime considered in our work.

In the complex 3D world task, we consider a simplified state predictor that is purely based on the time step and a single fixed training trajectory, denoted with superscript i. At time step t, the state predictor returns the state s^i_{t+k} within that training trajectory. Despite its simplicity, we find that even this simple state predictor can enable effective evaluation when paired with a trained IDM model.

Training details. To train the policies, we sample batches of 4096 (s_t, a_t) or (s_t, a_t, s_{t+k}) tuples for BC and IDM policies, respectively, using ground-truth states and actions from training trajectories.



(a) Four room (b) Zigzag (c) Maze (d) Multiroom

Figure 4: Performance per number of training trajectories for BC and PIDM in four tasks trained on human datasets. Lines and shading correspond to the average performance and standard deviation across 500 evaluations. We further visualize the number of samples required by PIDM and BC to reach 95% of the highest achievable performance with vertical dotted lines.

Table 1: Sample efficiency ratios of PIDM over BC for 2D navigation tasks and average across tasks

Task	Four room	Zigzag	Maze	Multiroom	average
$\eta_{\text{PIDM}}(95\%) \uparrow$	4.0	1.33	3.0	3.0	2.83

All networks are optimized end-to-end for $100\,000$ optimization steps from the BC and IDM losses defined in Equation (1) and Equation (3) using the Adam optimizer. For further details on hyperparameter tuning and architectures used in the 2D navigation and complex video game environments, please refer to Appendix B and Appendix C, respectively.

5.3 Sample Efficiency Gains for 2D Navigation

To study the sample efficiency gains of PIDM, we train a BC and PIDM on each dataset with varying numbers of trajectories, namely (1, 2, 5, 10, 20, 30, 40, 50). Our performance metric is the fraction of reached goals in the right order. For each task and number of training trajectories, we train BC and PIDM for ten random seeds, and evaluate four checkpoints throughout training of each seed using 50 rollouts, giving a total of 500 values of the performance metrics per checkpoint. We report aggregate results over these 500 values for the best checkpoint for each task and number of training trajectories.

To summarize efficiency gains, we compute efficiency ratios η_{PIDM} for each task, given by

$$\eta_{\text{PIDM}}(c) = \frac{n(\text{BC}, c\max)}{n(\text{PIDM}, c\max)}, \tag{11}$$

where $n(\text{A}, x)$ is the number of samples required by algorithm A to reach at least performance x and $\max$ denotes the maximum performance achievable in the task. In other words, we compute the ratio of the number of samples that BC and PIDM require to obtain the achievable performance up to a factor of $c < 1$. Figure 4 visualizes the percentage of reached goals for BC and PIDM across varying number of samples, and Table 1 summarizes 95% efficiency ratios, showing significant sample efficiencies for IDM over BC, as predicted by the analysis in Section 4. We find the efficiency gains to be most significant in the four room where BC requires $4\times$ more demonstrations than PIDM (40 vs 10) to achieve comparable performance.

5.4 Future Conditioning as a Variance Reduction Opportunity

Our theoretical insights of Theorem 1 and Theorem 2 indicate that states, in which there is high uncertainty on the action due to uncertainty in the future state, as given by $\Delta(s)$, are key to potential sample efficiency gains of PIDM over BC. What effect do these states have on the learned IDM policy, and how might they lead to improved efficiency?

To answer this question, we qualitatively analyze the learned PIDM policies in each task, and compute the EPE gaps for any particular state $\boldsymbol{s}_t = s$ within our datasets:

$$\Delta(s) \triangleq \text{Var}_{\boldsymbol{s}_{t+k}|s}\left(\mathbb{E}\left[\boldsymbol{a}_t \mid s, \boldsymbol{s}_{t+k}\right]\right), \text{ such that: } \Delta = \mathbb{E}_{\boldsymbol{s}_t}[\Delta(s)]. \tag{12}$$

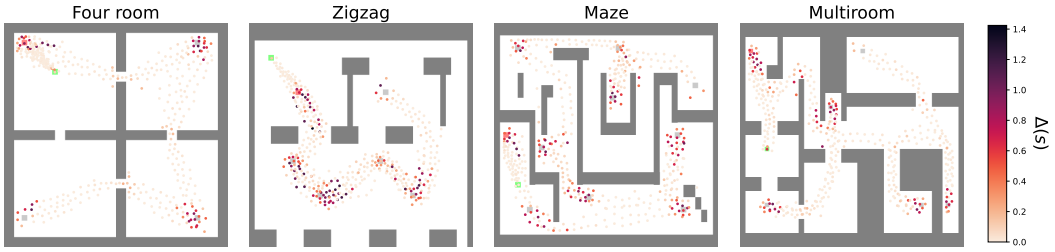
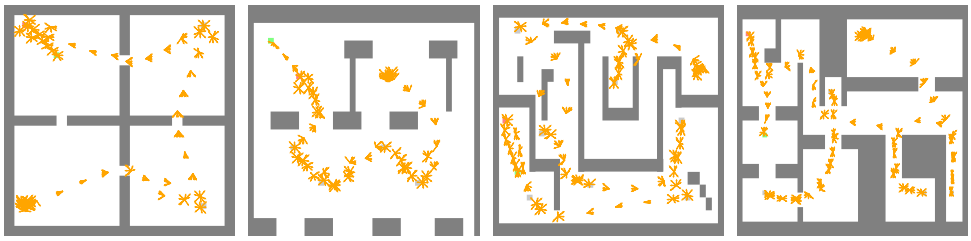


Figure 5: Visualized state-wise EPE gaps $\Delta(s)$ from Equation (12) computed for each dataset. We observe large gaps in states surrounding the goals where human actions are more diverse.



(a) Four room (b) Zigzag (c) Maze (d) Multiroom

Figure 6: Visualization of IDM policies when queried for representative states and possible future states in each cardinal and diagonal direction for all four tasks. Predicted actions spread out in states where the dataset exhibits large $\Delta(s)$.

To approximate $\Delta(s)$ for continuous states in our 2D environment, we discretize the map with k-means clustering over states and then compute the sample variance over actions grouped by centroid and future states within each dataset. Figure 5 visualizes the estimated values of $\Delta(s)$ for each dataset, with 500 clusters being computed to group states. Interestingly, we observe that the human movement exhibits significantly larger action variability in states surrounding the goals which the player has to navigate to.

To qualitatively analyze the PIDM policies, we train them with all available training trajectories. We obtain representative states by taking the centroids of k-means clustering (using $k = 75$ for maze and multiroom and $k = 50$ for four room and zigzag) and computing eight possible future states that are reachable within $k = 1$ step into each cardinal or diagonal direction. Then, we condition the PIDM policy with the 8 possible futures per centroid. Figure 6 visualizes the actions predicted by the PIDM policy for each centroid and future state pair. We can clearly see that the actions from the same centroid state are pointing in various directions only whenever the centroid state is close to a goal or other states with large $\Delta(s)$, as visualized in Figure 5. This result shows that the PIDM policy learns to attend to the future state in states where the future state helps to reduce uncertainty over the action prediction, which is precisely where Theorem 1 predicts a potential performance gain for PIDM. In contrast, in states where the action variability within the dataset is minimal, the PIDM policy exhibits significantly less diversity in predicted actions as seen by states in which most arrows point in a similar direction.

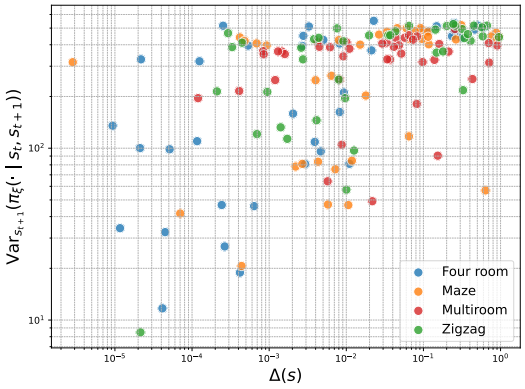


Figure 7: Correlation of state-wise action variance $\Delta(s)$ and the variance of PIDM policies.

Figure 7 further connects our newly gained understanding to Theorem 1 by showing the correlation between state-wise action variability $\Delta(s)$ and the variance of the PIDM policy for varying future states, indicating that the PIDM policy exhibits higher variance for states with higher $\Delta(s)$, meaning PIDM has learned to model as predicted by our theory. The variance of the IDM policy is computed for representative centroid states over eight future states.

5.5 Sample Efficiency Gains in a 3D World

After building an intuition for the efficiency gains of PIDM over BC both from a theoretical perspective, and under general conditions in a simplified environment, we now demonstrate similar benefits in a complex task that is representative of real-world applications. We consider the complex task that we name "Tour" as described in Section 5.1 in which the agent needs to navigate from images in the 3D world of a modern video game that requires real-time inference, with stochastic transitions, and where success is defined by achieving 11 milestones (see also Appendix C.2).

To compare agent performance on this task, BC and PIDM are trained using 5, 15, 20, 25 and 30 demonstrations. Our performance metric is the percentage of milestones that have been reached. We train BC and PIDM for 5 random seeds, and evaluate the latest checkpoint of each seed with 10 rollouts, giving a total of 50 values of the performance metric per number of demonstrations for each algorithm. Figure 1b shows PIDM achieves 95% success (on average) at the end of training, and a success rate of 87% with 15 demonstrations, while BC requires 25 demonstrations to reach a 81% success rate, so BC requires $\eta_{\text{PIDM}}(80\%) = 1.66$ times more samples than BC to reach a success rate of 80%. This confirms the potential of PIDM to improve sample efficiency over BC even in the small data regime. Moreover, when additional data sources are available, we expect these efficiency gains to increase.

6 Conclusion

This work analyzed the performance advantages of predictive inverse dynamics models (PIDM) as an alternative to behavior cloning (BC) for offline imitation learning, particularly in low-data regimes. Through theoretical analysis and empirical experiments, we shed light onto the advantages of PIDM observed in prior work, proving that PIDM can reduce action prediction error by conditioning on future states, especially in regions of high uncertainty. Empirical results across navigation tasks in 2D and 3D environments confirmed sample efficiency gains, with BC requiring up to $4\times$ more demonstrations than PIDM to achieve comparable performance. Interestingly, qualitative analysis showed that learned PIDM policies attend to future states only when they provide informative context for reducing prediction variance. Altogether, this work provides a principled explanation for PIDM's effectiveness and offers insights that pave the way for more efficient imitation learning methods that leverage state prediction and future conditioning.

On the other hand, although we proved that PIDM is solving a problem that is not harder than BC, more work is needed to understand under which conditions PIDM is superior to BC. In particular, the impact of learning the future state distribution remains unclear and it is not evident whether learning this distribution is easier or more robust than directly learning actions. Additionally, the quality of the state predictor can significantly affect overall performance. Moreover, we focused on point estimators of the policy distribution for both BC and PIDM, which is a fair comparison, but it is unclear whether the observed sample efficiency gains of PIDM over BC would remain for a richer policy class (e.g., using diffusion models). These limitations highlight important directions for future work, including better understanding the trade-offs in learning state predictors, modeling choices, and extending the theory to more general settings.

REFERENCES

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15791–15801, 2025.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provably filtering exogenous distractors using multistep inverse dynamics. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RQLLzMCefQu.

Bin Fang, Shidong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3(4):362–369, 2019.

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

Dylan J. Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 120602–120666, 2024.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pp. 1–7, 2025.

Riashat Islam, Manan Tomar, Alex Lamb, Yonathan Efroni, Hongyu Zang, Aniket Didolkar, Dipendra Misra, Xin Li, Harm Van Seijen, Remi Tachet des Combes, et al. Agent-controller representations: Principled offline rl with rich exogenous information. *arXiv preprint arXiv:2211.00164*, 2022.

Eamonn Keogh. *Instance-Based Learning*, pp. 549–550. Springer US, 2010.

Anurag Koul, Shivakanth Sujit, Shaoru Chen, Ben Evans, Lili Wu, Byron Xu, Rajan Chari, Riashat Islam, Raihan Seraj, Yonathan Efroni, et al. Pclast: Discovering plannable continuous latent states. In *International Conference on Machine Learning*, 2023.

Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Rajiv Didolkar, Dipendra Misra, Dylan J Foster, Lekan P Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Alexander Levine, Peter Stone, and Amy Zhang. Multistep inverse is not all you need. *Reinforcement Learning Journal*, 2:884–925, 2024.

Zakaria Mhammedi, Dylan J Foster, and Alexander Rakhlin. Representation learning with multi-step inverse kinematics: An efficient and optimal approach to rich-observation RL. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24659–24700. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/mhammedi23a.html.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2): 1–179, 2018.

Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos A Theodorou, and Byron Boots. Imitation learning for agile autonomous driving. *The International Journal of Robotics Research*, 39(2-3):286–302, 2020.

Tim Pearce and Jun Zhu. Counter-strike deathmatch with large-scale behavioural cloning. In *IEEE Conference on Games*, pp. 104–111, 2022.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6): 233–242, 1999.

Lukas Schäfer, Logan Jones, Anssi Kanervisto, Yuhan Cao, Tabish Rashid, Raluca Georgescu, David Bignell, Siddhartha Sen, Andrea Treviño Gavito, and Sam Devlin. Visual encoders for imitation learning in modern video games. In *Workshop on Adaptive and Learning Agents*, 2023.

Jinghuan Shang, Karl Schmeckpeper, Brandon B. May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. In *8th Annual Conference on Robot Learning*, 2024.

Sebastian Thrun, Knut Möller, and Alexander Linden. Planning with an adaptive world model. *Advances in neural information processing systems*, 3, 1990.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, 2025.

Amber Xie, Oleh Rybkin, Dorsa Sadigh, and Chelsea Finn. Latent diffusion planning for imitation learning. In *International Conference on Machine Learning*, 2025.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Proofs

A.1 Proof of Theorem 1

Theorem 1. $\Delta = \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}_{\boldsymbol{s}_{t+k}|\boldsymbol{s}_t}\left(\mathbb{E}\left[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k}\right]\right)\right] \geq 0.$

Proof: The prediction error for these estimators is given by:

$$\mathrm{EPE}(\overline{\mu}) = \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\left[\left(\boldsymbol{a}_t - \overline{\mu}(\boldsymbol{s}_t)\right)^2\right], \tag{13}$$

$$\mathrm{EPE}(\overline{\xi}) = \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t,\boldsymbol{s}_{t+k}}\left[\left(\boldsymbol{a}_t - \overline{\xi}(\boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right)^2\right]. \tag{14}$$

We can rewrite the EPE by using iterated expectation and replacing the definitions of optimal estimators:

$$\begin{aligned}\mathrm{EPE}(\overline{\mu}) &= \mathbb{E}_{\boldsymbol{s}_t}\left[\mathbb{E}_{\boldsymbol{a}_t|\boldsymbol{s}_t}\left[\left(\boldsymbol{a}_t - \mathbb{E}\left[\boldsymbol{a}_t \mid \boldsymbol{s}_t\right]\right)^2\right]\right] \\ &= \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right] \qquad (15)\\ \mathrm{EPE}(\overline{\xi}) &= \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+k}}\left[\mathbb{E}_{\boldsymbol{a}_t|(\boldsymbol{s}_t,\boldsymbol{s}_{t+k})}\left[\left(\boldsymbol{a}_t - \mathbb{E}[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k}]\right)^2\right]\right], \\ &= \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+k}}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right]. \qquad (16)\end{aligned}$$

We can further simplify Equation (15). First, we apply the law of total variance to $\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$:

$$\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{s}_{t+k}|\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right] + \mathrm{Var}_{\boldsymbol{s}_{t+k}|\boldsymbol{s}_t}\left(\mathbb{E}[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k}]\right). \tag{17}$$

Second, we take the expectation over $\boldsymbol{s}_t$:

$$\mathbb{E}_{\boldsymbol{s}_t}[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)] = \mathbb{E}_{\boldsymbol{s}_t}\left[\mathbb{E}_{\boldsymbol{s}_{t+k}|\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right]\right] + \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}_{\boldsymbol{s}_{t+k}|\boldsymbol{s}_t}\left(\mathbb{E}[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k}]\right)\right]. \tag{18}$$

Third, we simplify the first term of the r.h.s.:

$$\mathbb{E}_{\boldsymbol{s}_t}\left[\mathbb{E}_{\boldsymbol{s}_{t+k}|\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right]\right] = \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+k}}[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})]. \tag{19}$$

Finally, we have:

$$\mathbb{E}_{\boldsymbol{s}_t}[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)] = \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+k}}[\mathrm{Var}(\boldsymbol{a} \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})] + \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}_{\boldsymbol{s}_{t+k}|\boldsymbol{s}_t}(\mathbb{E}[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k}])\right]. \tag{20}$$

Now, we can easily compute the performance gap between the MSEs of both estimators:

$$\begin{aligned}\mathrm{EPE}(\overline{\mu}) - \mathrm{EPE}(\overline{\xi}) &= \mathbb{E}_{\boldsymbol{s}_t}[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)] - \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+k}}[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})] \\ &= \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}_{\boldsymbol{s}_{t+k}|\boldsymbol{s}_t}\left(\mathbb{E}[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right]\right. . \end{aligned} \tag{21}$$

■

A.2 Proof of Corollary 1

Corollary 1. *Let $\widehat{\mu}(\boldsymbol{s}_t)$ and $\widehat{\xi}(\boldsymbol{s}_t, \boldsymbol{s}_{t+k})$ be unbiased estimators for the BC and IDM policies, respectively. Let $\widehat{\delta}$ denote the difference in the estimators' own variance:*

$$\widehat{\delta} \triangleq \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}\left(\widehat{\mu}\left(\boldsymbol{s}_t\right)\right)\right] - \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+k}}\left[\mathrm{Var}\left(\widehat{\xi}\left(\boldsymbol{s}_t, \boldsymbol{s}_{t+k}\right)\right)\right]. \tag{22}$$

Then, we have:

$$\widehat{\Delta} \triangleq \mathrm{EPE}\left(\widehat{\mu}\right) - \mathrm{EPE}\left(\widehat{\xi}\right) = \Delta + \widehat{\delta}. \tag{23}$$

Proof: Since the estimators are unbiased, we know that:

$$\begin{aligned}\mathbb{E}\left[\widehat{\mu}(\boldsymbol{s}_t)\right] &= \mathbb{E}\left[\boldsymbol{a}_t \mid \boldsymbol{s}_t\right], \\ \mathbb{E}\left[\widehat{\xi}(\boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right] &= \mathbb{E}\left[\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k}\right].\end{aligned} \tag{24}$$

We can express $\boldsymbol{a}_t = \mathbb{E}[\boldsymbol{a}_t \mid \boldsymbol{s}_t] + \boldsymbol{\varepsilon}$ where $\boldsymbol{\varepsilon}$ is a zero-mean random variable, possibly dependent on $\boldsymbol{s}_t$. Using the definition of $\overline{\mu}$ from Theorem 1, the EPE can be expressed as the sum of the irreducible variance and the estimator's own variance:

$$\begin{aligned}\mathrm{EPE}(\widehat{\mu}) &= \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\left[\left(\boldsymbol{a}_t - \widehat{\mu}(\boldsymbol{s}_t)\right)^2\right] \\ &= \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\left[\left(\boldsymbol{a}_t - \widehat{\mu}(\boldsymbol{s}_t) + \overline{\mu}(\boldsymbol{s}_t) - \overline{\mu}(\boldsymbol{s}_t)\right)^2\right] \\ &= \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\left[\left(\left(\boldsymbol{a}_t - \overline{\mu}(\boldsymbol{s}_t)\right) + \left(\overline{\mu}(\boldsymbol{s}_t) - \widehat{\mu}(\boldsymbol{s}_t)\right)\right)^2\right] \\ &= \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\left[\left(\boldsymbol{a}_t - \overline{\mu}(\boldsymbol{s}_t)\right)^2\right] + \mathbb{E}_{\boldsymbol{s}_t}\left[\left(\overline{\mu}(\boldsymbol{s}_t) - \widehat{\mu}(\boldsymbol{s}_t)\right)^2\right] \\ &\quad + 2\mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\left[\left(\boldsymbol{a}_t - \overline{\mu}(\boldsymbol{s}_t)\right)\left(\overline{\mu}(\boldsymbol{s}_t) - \widehat{\mu}(\boldsymbol{s}_t)\right)\right] \\ &= \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right] + \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\widehat{\mu}(\boldsymbol{s}_t))\right],\end{aligned} \tag{25}$$

where the cross-term vanishes since $\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\boldsymbol{\varepsilon} \mid \boldsymbol{s}_t\right] = 0$:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\left[\left(\boldsymbol{a}_t - \overline{\mu}(\boldsymbol{s}_t)\right)\left(\overline{\mu}(\boldsymbol{s}_t) - \widehat{\mu}(\boldsymbol{s}_t)\right)\right] &= \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{\varepsilon}}\left[\left(\boldsymbol{\varepsilon} + \overline{\mu}(\boldsymbol{s}_t) - \overline{\mu}(\boldsymbol{s}_t)\right)\left(\overline{\mu}(\boldsymbol{s}_t) - \widehat{\mu}(\boldsymbol{s}_t)\right)\right] \\ &= \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{\varepsilon}}\left[\boldsymbol{\varepsilon}\left(\overline{\mu}(\boldsymbol{s}_t) - \widehat{\mu}(\boldsymbol{s}_t)\right)\right] \\ &= \mathbb{E}_{\boldsymbol{s}_t}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\boldsymbol{\varepsilon} \mid \boldsymbol{s}_t\right]\left(\overline{\mu}(\boldsymbol{s}_t) - \widehat{\mu}(\boldsymbol{s}_t)\right)\right] \\ &= 0.\end{aligned} \tag{26}$$

Following the same approach for $\overline{\xi}$, we have:

$$\mathrm{EPE}(\widehat{\xi}) = \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+k}}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right] + \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{s}_{t+k}}\left[\mathrm{Var}(\widehat{\xi}(\boldsymbol{s}_t, \boldsymbol{s}_{t+k}))\right]. \tag{27}$$

Subtracting (27) from (25) and grouping terms according to definitions (5) and (22) concludes the proof. ■

A.3 Proof of Theorem 2

Theorem 2. *Let $\widehat{\mu}_n(\boldsymbol{s}_t)$ and $\widehat{\xi}_m(\boldsymbol{s}_t, \boldsymbol{s}_{t+k})$ be unbiased, asymptotically efficient estimators for the BC and IDM policies, respectively, where n and m denote the minimum number of samples required to achieve error level ε. Then, for large enough n and m, we have:*

$$\eta \triangleq \frac{n}{m} \approx \frac{C_\mu}{C_\xi}\left(1 + \frac{\Delta}{\varepsilon - \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]}\right) \geq 0. \tag{28}$$

Proof: From (25) and using the asymptotic variance approximation:

$$\begin{aligned}\mathrm{EPE}(\widehat{\mu}_n) &= \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right] + \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\widehat{\mu}(\boldsymbol{s}_t))\right] \\ &\approx \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right] + \frac{C_\mu}{n}.\end{aligned} \tag{29}$$

Since $\mathrm{EPE}(\widehat{\mu}_n) = \varepsilon$, we can solve for n:

$$n \approx \frac{C_\mu}{\varepsilon - \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]}. \tag{30}$$

Following the same reasoning for $\mathrm{EPE}(\widehat{\xi})m)$, we get:

$$m \approx \frac{C_\xi}{\varepsilon - \mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{s}_{t+k}}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right]}. \tag{31}$$

Using the definition of the sample efficiency ratio, we have:

$$\begin{aligned}\eta &\triangleq \frac{n}{m} \\ &\approx \frac{C_\mu}{C_\xi}\left(\frac{\varepsilon - \mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{s}_{t+k}}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right]}{\varepsilon - \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]}\right).\end{aligned} \tag{32}$$

From the first line of (21), We can obtain this identity:

$$\mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{s}_{t+k}}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right] = \mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right] - \Delta. \tag{33}$$

By expanding it in (32) and simplifying terms, we get (28). Finally, we just need to ensure the fraction is non-negative. To do so, note that

$$\varepsilon \geq \max\{\mathbb{E}_{\boldsymbol{s}_t}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right], \mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{s}_{t+k}}\left[\mathrm{Var}(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_{t+k})\right]\} \geq 0, \tag{34}$$

since the prediction error cannot be smaller than irreducible error. ■

B Details for 2D Navigation Environment and Experiments

In this section, we describe further details about the 2D navigation environment and the experiments in this setting.

B.1 Additional Environment Details

Tasks within the 2D navigation environment specify a layout of the environment and differ in the number of goals. The general setting stays the same with each task specifying an order to its goals and the agent needs to reach a goal before being able to reach any subsequent goals. This setup makes these tasks punishing since missing any goal will mean that subsequent goals cannot be reached anymore unless the agent returns back to the currently required goal. An episode within any task finishes after all goals have been reached, or after a maximum number of time steps has been reached. The state dimensionality, number of goals, and maximum number of time steps for each task is listed in Table 2.

In all tasks, we introduce stochasticity in the transition function through Gaussian noise. Instead of displacing the agent based on its selected action $a \in [-1, 1]^2$ alone, we displace the agent based on clipped noise-added actions:

$$\mathrm{clip}(a + \epsilon, -1, 1) \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, 0.2 \cdot \mathbb{1}) \tag{35}$$

We emphasize that the sampled noise is *not* modifying the actions but rather modeled as part of the environment, meaning that, from the perspective of the agent, the environment transitions are stochastic given a state and action. The agent will bounce off any walls that it collides with with walls being visualized as black bars in all figures.

Table 2: Statistics of all four 2D navigation tasks and the human datasets. The first four columns correspond to properties of the tasks, given by the number of goals, maximum number of time steps to complete the task, and the state dimensionality, while the last four columns correspond to the total number of trajectories/ time steps within the collected dataset (across all 50 trajectories) and statistics over the trajectory length.

Task	Num goals	Max time steps	$\|s\|$	Total steps	Trajectory length		
					Min	Avg	Max
Four room	4	200	14	5821	103	116.42	154
Zigzag	6	150	20	4009	66	80.18	106
Maze	10	300	32	9785	176	195.70	227
Multiroom	6	500	20	12 961	241	259.22	314

B.2 Dataset Details

Table 2 shows statistics for each 2D navigation task and the collected human dataset. During data collection, the human player was instructed to collect high-quality trajectories that reach all goals as fast as possible. The player controlled the movement of the controllable agent using the joystick of a gamepad controller. We note that the player was unaware of the data analyses that we conducted to avoid any risk of introducing bias.

B.3 Hyperparameter Search

To ensure fair comparison, we conducted a comparable gridsearch for both BC and PIDM in the multiroom task using 50 training trajectories. First, we conducted a gridsearch over the model architecture considering sixteen different sizes of the MLP network architecture, the use of normalization in the network (either batch normalization, layer normalization, or no normalization), and learning rate with three constant candidate learning rates ($1e^{-6}, 1e^{-5}, 1e^{-4}$. The considered architectures consisted of any of five MLP blocks before any potential normalization layer and any of the five MLP blocks after the normalization. The considered network blocks were:

1. MLP(256)
2. MLP(256, 128)
3. MLP(512, 256)
4. MLP(512, 1024, 256)
5. MLP(1024, 2048, 512)

From this search, we identified a single network architecture that performed best for BC and among the best for PIDM to keep for consistent comparisons thereafter. The architecture consists of network block MLP(512, 1024, 256) followed by batch normalization before MLP(256, 2) with the last 2D layer outputting the action logits. We apply ReLU activation in between all layers and $tanh$ activation to the output logits.

After fixing the network architecture, we still found some training instability for BC and IDM so we decided to further tune the learning rate for BC and IDM by searching over 14 learning rate configurations defined by their initial learning rate, and potential learning rate scheduling, and considered each configuration with and without gradient norm clipping. We first tuned the learning rate configuration for BC and IDM in multiroom after which we found IDM training to be stable across tasks. For BC, we further tuned the learning rate for each individual task to obtain stable training results. The identified learning rates are shown in the table below.

Table 3: Learning rate configuration for each task and algorithm

Task	BC configuration	IDM configuration
Four room	Linear decay $1e^{-3} \rightarrow 1e^{-6}$ over 50 000 steps + grad norm clipping	constant $1e^{-5}$
Zigzag	Linear decay $1e^{-4} \rightarrow 1e^{-6}$ over 50 000 steps + grad norm clipping	constant $1e^{-5}$
Maze	Linear decay $1e^{-4} \rightarrow 1e^{-6}$ over 50 000 steps + grad norm clipping	constant $1e^{-5}$
Multiroom	Linear decay $1e^{-4} \rightarrow 1e^{-6}$ over 50 000 steps	constant $1e^{-5}$

C Details for Complex Task in 3D-World

C.1 Dataset Details

The dataset consists of 30 demonstrations collected by a human playing the game. Table 4 shows the number of steps and length (seconds) of the demonstrations in the dataset.

Table 4: Statistics of demonstrations of "Tour" task.

	Total steps			Trajectory length (in seconds)		
Task	Min	Avg	Max	Min	Avg	Max
Tour	1006	1067.2 ± 29.4	1139	33.83	35.91 ± 0.99	38.29

C.2 Additional Environment Details

Table 5 contains the 11 milestones required to complete the "Tour" task.

C.3 Evaluation Protocol

Two human experts that were familiar with the task evaluated all the rollouts. The evaluation was blind to avoid cognitive bias, since the evaluators did not know whether the rollout they were evaluating corresponded to BC or PIDM. For each rollout, they checked if the agent achieved every milestone of the task, scoring with value 1 if the milestone was achieved and 0 otherwise, so the maximum score per rollout is 11 (the number of milestones). However, we report performance in terms of % of this maximum score.

C.4 Additional Algorithmic Details

C.4.1 Vision encoder

We use "theia-base-patch16-224-cddsv" from Huggingface as pretrained vision encoder. The vision encoder remains frozen during training (and evaluation). Each video frame is passed to this encoder, which generates an embedding vector of length 768. This embedding vector of the current frame is the input to the BC policy. While the embedding of the current and future frames are the input to the state encoder of the PIDM.

C.4.2 Grid search

To ensure fair comparison and some degree of generalization, we conducted a comparable grid search for both BC and PIDM in a different more complex task, with more milestones, for which none of the algorithms could achieve 100% performance after being trained with a dataset of 30 demonstrations. We used the results from the grid search in the 2D environment as a basis, with ReLu activations in between all layers and batch normalization at the output of the state encoder. The output was a *tanh* activation. We evaluated two different sizes of the MLP network architecture, under two learning rates. The considered MLP network blocks were:

1. State encoder: MLP(1024, 512, 512), Policy: MLP(512, 256)

2. State encoder: MLP(1024, 2048, 1024, 512, 512), Policy: MLP(512, 512, 256)

We also tried two learning rates per algorithm, namely linear decay $1e$-3 $\rightarrow$ $1e$-6 and $5e$-5 for PIDM, and linear decay $1e$-4 $\rightarrow$ $1e$-6 and $1e$-4 for BC, with decay for 60,000 steps. Other hyperparameters that remained constant where: training lasted 60,000 steps, optimization algorithm was Adam with standard parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e$-8), and batch size was 4096. Moreover, We also use as suggested by the grid search in the 2D environment.

We observed the small network blocks with linear decay was the best combination, and BC (88%) achieved slightly higher average performance than PIDM (86%) for that task, but not statistically significant. For training in the "Tour" task, we used this configuration and used the rest of the parameters used for the grid search, with the only exception of the number of training steps, which we increased to 100,000 and we could see the loss had converged and remained stable after 60,000 (which is when the linear decay stops).

Table 5: Milestones of "Tour" task in Bleeding Edge with corresponding thumbnails

#	Milestone	Thumbnail
1	Start off with a sharp left 180° turn	
2	Navigate towards the first health marker and grab it	
3	Cross the main floor of the Dojo	
4	Take a left onto the ramp	
5	Turn while going up and stay on the ramp for 6-7 secs	
6	Right turn and navigate the corridor	
7	Circumvent the box by steering left	
8	Navigate towards the second health marker and grab it	
9	Pass through the final corridor	
10	Hit the Gong	
11	Stop and don’t move anymore