When Novices Teach Better: Skill-Level Effects in Behavioral Cloning

Saumik Narayanan
Washington University in St. Louis
saumik@wustl.edu

Kassa Korley
Microsoft Research
kassakorley@gmail.com

Siddhartha Sen
Microsoft Research
sidsen@microsoft.com

Chien-Ju Ho
Washington University in St. Louis
chienju.ho@wustl.edu

Abstract

Behavioral cloning (BC), which trains models from offline demonstrations, is a common approach in reinforcement learning settings. Prior work argues that BC requires expert demonstrations and performs poorly when trained on low-skill data. We challenge this assumption by showing that, in certain regimes, training on low-skill data can yield models that outperform those trained on high-skill data. Because low-skill data is often cheaper and more easily acquirable, this finding has important practical implications. To explain this result, we introduce the notion of fragility, characterizing how a policy's reward degrades under errors, and provide theoretical insights on how fragility can predict low-skill outperformance. We test our approach in a synthetic environment and MuJoCo and validate it using human data from chess and iRacing. Motivated by these findings, we connect our results to curriculum learning by structuring training according to demonstrator skill, rather than task difficulty as in standard curriculum design, and show that such skill-based curricula can improve performance relative to standard BC approaches.

1 Introduction

Behavioral cloning (BC) is a form of imitation learning that trains models to replicate behaviors from offline demonstrations [3]. It is widely used in reinforcement learning, particularly when explicit reward signals are unavailable or when online interaction is impractical due to safety or cost concerns [44]. In some settings, BC can rival or even surpass offline RL methods [20, 31]. When trained on human data, it produces models that exhibit human-like behavior—a desirable property for AI systems that interact with people. It is commonly believed, however, that BC performs best with expert, high-skill demonstrations [23] and performs poorly with low-skill demonstrations [21]. As a result, practitioners often adopt Filtered BC [26], discarding all but the highest-skill demonstrations. While effective when expert data is plentiful, this approach is limiting in practice, where datasets often span a wide range of skill levels and expert demonstrations are costly or scarce.

This paper revisits the common belief that BC works best with only expert data. We demonstrate that in certain types of low-data regimes, training on low-skill demonstrations can actually outperform training on expert data. Since low-skill data is often cheaper and more easily acquirable, this finding has important practical implications. To understand this counterintuitive result, we introduce a central concept—***fragility***, which quantifies how a policy (and the demonstrations it generates) degrade in performance under errors. We show empirically that the surprising advantage of low-skill demonstrations tends to arise when high-skill policies are more fragile than low-skill ones, and provide theoretical insights that help explain this relationship.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Insights from psychology and education help motivate this perspective. Experts often rely on tacit knowledge and omit intermediate steps, making their behavior harder to model accurately [30]. High-skill strategies may operate closer to failure margins, amplifying the consequences of small modeling errors [35] and compounding errors in BC can cause learned policies to drift into unstable states [32]. A racing game illustrates the point: expert drivers maximize speed and hug the track's edges, achieving the fastest lap times but leaving little margin for error, while novice drivers stay near the center and avoid fragile states. With unlimited training data, BC can reproduce expert strategies, but with limited data, errors in expert demonstrations often occur in fragile states, leading to catastrophic outcomes, whereas errors in novice data are more easily absorbed. These perspectives suggest that the usefulness of a demonstration depends not only on skill but the tendency of the policy to enter fragile states, motivating our study of whether and when training low-skill demonstrations can yield more reliable BC models and whether this leads to useful insights in overall BC training.

More concretely, we investigate the role and impact of low-skill data in BC across four domains. We use expected reward as our measure of skill: policies with higher rewards are considered high-skill, and those with lower rewards are low-skill. We begin with controlled synthetic environments, where we manipulate rewards in a tree-based domain and simulate policies across different skill levels. This setup allows us to systematically vary environment characteristics and directly test how skill interacts with fragility. We then verify our results across 12 tasks in MuJoCo, a physics engine used for simulating robotics and biomechanics. After, we turn to chess, where public datasets provide billions of human games across skill levels. Finally, we study racing, using human demonstrations obtained through collaboration with the racing simulator iRacing, allowing us to evaluate whether our findings generalize to a complex real-world setting. Taken together, these domains provide complementary perspectives that reveal when and why low-skill data can be more useful than expert data.

Our experiments show that when low-skill demonstrations are less fragile, BC models trained on the low-skill data achieve higher performance at small budgets, and this effect disappears when high-skill demonstrations are less fragile or when data budgets increase sufficiently. These observations hold consistently across all four domains.

Motivated by these findings, we connect our results to curriculum learning in the setting of mixed-skill datasets. Traditional curriculum learning typically structures training according to task difficulty. Our findings highlight that demonstrator skill plays a central role in learning from such data, thereby enabling curricula that organize training based on demonstrator skills. Across both synthetic and real-world domains, we show that adopting this skill-based curriculum perspective leads to consistent performance improvements in behavioral cloning. These results provide new insights in leveraging demonstrations spanning a range of skill levels.

Contributions. Overall, this work revisits the common belief that BC works best when trained only on expert data and makes three contributions:

- We propose *fragility* as a measure of BC training datasets. We provide methods for measuring the fragility of a demonstrator and show under certain assumptions, fragility can predict the relative performance of BC models.
- We provide the counterintuitive finding across four domains that BC models trained on low-skill data can outperform models trained on high-skill data. We also show that this effect is strongly predicted by the relative fragility of low-skill and high-skill agents.
- We connect our findings to curriculum learning by applying simple, off-the-shelf curriculum schedules over demonstrator skill, rather than task difficulty. These results show that skill-based curricula can improve BC performance across domains, opening up applications for adaptive skill-based curriculum design.

2 Related Work

RL with Human Demonstrations. A substantial body of work addresses reinforcement learning from human demonstrations. Imitation learning trains models to replicate human policies directly from demonstrations [18]. The simplest form, Behavioral Cloning (BC), frames the task as supervised learning: given a state, the model predicts the action [3]. BC is widely applied in domains such as autonomous driving [40], robotics [11], and gaming [33], since it avoids explicit reward modeling or transition simulation. It can even outperform reward-based offline RL in some settings [22, 20, 31,

26]. Alternatives to BC have also been developed. Offline RL incorporates reward signals during training [24], which can improve task performance [21]. Inverse Reinforcement Learning aims to infer the underlying reward function of human behavior [2]. Finally, online RL approaches can leverage demonstrations, when access to a human oracle is available [39].

Leveraging Low-Skill Human Demonstrations. We study whether low-skill human data can improve the training of machine learning models. In supervised learning, this idea has been explored in crowdsourcing [17, 38]. By recruiting laypeople to provide relatively low-skill annotations, large-scale datasets can be built without relying on costly expert input. This strategy has been highly successful, with prominent examples such as ImageNet [10] driving progress in supervised learning.

In contrast, in the context of BC, prior work has shown that training on noisy or suboptimal demonstrations often degrades performance in BC [12, 26, 19], with follow-up theory reinforcing the view that BC typically requires expert data [21]. To overcome this, extensions to BC have been proposed: filtered BC discards low-skill demonstrations using post-processing classifiers, improving performance in robotics [9, 41, 6, 16], gaming [23, 33], and language modeling [42]; weighted BC assigns importance weights to demonstrations [14]; and conditional BC augments states with contextual variables such as modified rewards [8] or goals [13]. Different from this line of work, we focus on investigating whether and when low-skill data might improve BC training.

More broadly, reinforcement learning research has examined training with low-skill or mixed-skill human data, often using BC as a benchmark for evaluating alternative methods [12, 26, 19, 15]. Other studies have explored training with low-skill non-human data, typically generated by perturbing expert demonstrations with noise or using partially trained models [5, 7, 46].

Critical States. The fragility metric in our work is related to the notion of critical states in BC and RL, the idea that certain regions of the state space are riskier or more important than others. For instance, Kumar et al. [20] define critical states as a subset of states from which taking a wrong action is especially costly to recover. Similarly, Zhang et al. [45] propose an RL approach that involves first deploying a model in a less critical source environment (e.g., a traffic simulator) to learn representations of especially dangerous states, which can then be avoided once deployed in real-world environments. While related, our fragility concept differs in focus: it characterizes the overall *policy*, whereas the notion of critical states characterizes the *states*.

Curriculum Learning. Our approach builds on the idea of structuring training around demonstrations of varying skill levels. This is inspired by curriculum learning [4], a technique modeled after human pedagogy that organizes the training process to improve learning outcomes. Most research in curriculum learning structures training by gradually increasing difficulty from easy to hard examples. Curricula have also been designed around tasks, model architectures, or evaluation metrics [36]. Although first applied to supervised learning, curriculum learning has since gained traction in RL as well [29]. In our method, we structure the data by demonstrator skill level, beginning training with low-skill demonstrations and progressively incorporating higher-skill demonstrations.

3 Setting and Methodology

3.1 Problem Setting

Markov decision process (MDP). BC is formulated in the context of an MDP defined by the tuple $\{\mathcal{S}, \mathcal{A}, P, R, \gamma, S_0\}$. $\mathcal{S}$ is the state space, $\mathcal{A}(s)$ is the action space in state s, $P(s' \mid s, a)$ is the probability of transitioning to state s' after taking action a in state s, $R(s, a)$ is the reward for taking action a in state s, γ is the discount factor, and S_0 is the distribution of initial states. In this paper, our focus is on finite MDPs with $\gamma = 1$ and episode length T, though our discussion generalizes to infinite-horizon MDPs with $\gamma < 1$.

Policy and rewards. A policy is denoted by $\pi(a|s)$, indicating the probability for taking action a in state s. We define $\zeta^\pi(s_0) = \{(s_t, a_t, r_t)\}|_{t=1,\dots,T}$ as the rollout of the policy π starting at state s_0 and (s_t, a_t, r_t) denote the (state, action, reward) at step t in the rollout. The value function of a policy at a state s is defined as $V^\pi(s) = \mathbb{E}_{\zeta^\pi(s)}\big[\sum_{t=1}^{T} r_t\big]$. The cumulative reward of a policy is defined as $J(\pi) = \mathbb{E}_{s_0 \sim \mathcal{S}_0}[V^\pi(s_0)]$. We also define $Q^\pi(s, a) = \sum_{s'} P(s'|s, a)(R(s, a) + V^\pi(s'))$.

Skill levels. The skill level of a policy π is defined by the expected reward of the policy $J(\pi)$. We focus on situations when there exist policy functions at a range of skill levels. We denote these policies as $\{\pi_1, \pi_2, ..., \pi_K\}$. Without loss of generality, we assume $J(\pi_1) \leq ... \leq J(\pi_K)$. For low-skill (high-skill) data or demonstrations, we refer to the state-action pairs (s, a) generated by the policy with relatively low (high) rewards among the set of all policies. The total set of state-action pairs generated by the policy function π_k comprises a dataset $\mathcal{D}_{\pi_k}$.

Behavioral cloning. BC learns a policy function using supervised learning: Given a dataset $\mathcal{D}$ of rollouts sampled from ζ^π, we learn a policy $\hat{\pi}$ which maps each state to its corresponding action.

3.2 Fragility

Behavioral cloning trained on finite data inevitably incurs errors at test time, particularly in low-budget regimes. Although higher-skill policies achieve higher rewards, they may differ substantially in how costly deviations from the demonstrated action are when errors occur. This motivates a metric that captures not the nominal quality of a policy, but the expected degradation in performance under action-level errors, which we refer to as *fragility*.

Formally, we define the fragility measure $\mathcal{F}$ as the expected loss in reward when $\hat{\pi}$ makes a single error (a decision according to some policy $\hat{\pi}_{\text{err}}$ instead of π). This is computed with the formula :

$$\mathcal{F}(\pi) = \mathbb{E}_{s\sim\zeta^\pi}\left[V^\pi(s) - \mathbb{E}_{a\sim\hat{\pi}_{\text{err}}(\cdot|s)}\Big[Q^\pi(s,a)\Big]\right]$$

This measure requires knowledge of the error policy $\hat{\pi}_{\text{err}}$, which is usually unknown before we have run BC. Below, we define an empirical approximation which approximates $\hat{\pi}_{\text{err}}(a|s) = \frac{1}{|\mathcal{A}|}$ as a uniform random policy, either sampled directly on a discrete action space, or from a discretized version of a continuous action space.

In this paper, we focus on this simple, model-agnostic error model as a pre-training proxy for the unknown errors that a learned BC policy may make. The fragility values and rankings may depend on the choice of error model, which we discuss further in Appendix A. We also note the need for an approximate value function V^π. This could be obtained by training V^π on D_π or leveraging known domain value functions (e.g., Stockfish in chess).

$$\overline{\mathcal{F}}(\pi) = \frac{1}{|\mathcal{D}_\pi|}\sum_{\zeta\in\mathcal{D}_\pi}\frac{1}{T}\sum_{k=1}^{T}\left(V^\pi(s_k) - \frac{1}{|\mathcal{A}|}\sum_a Q^\pi(s_k,a)\right) \tag{1}$$

3.2.1 Connection between Fragility and BC

Intuitively, fragility is useful because BC models trained on finite data inevitably make errors, especially in low-budget regimes, and the cost of these errors depends on where the demonstrated policy tends to operate. We first state an exact condition for when low-skill demonstrations can outperform high-skill demonstrations after BC training. Let $\hat{\pi}_{\pi,B}$ denote the policy learned by BC from a budget B of demonstrations by policy π, and define the imitation-induced degradation $\Delta_B(\pi) = J(\pi) - \mathbb{E}[J(\hat{\pi}_{\pi,B})]$, where the expectation is over the sampled training data and training randomness. Consider a low-skill policy π_L and a high-skill policy π_H, with $J(\pi_H) > J(\pi_L)$. Then low-skill BC outperforms high-skill BC exactly when

$$\Delta_B(\pi_H) - \Delta_B(\pi_L) > J(\pi_H) - J(\pi_L). \tag{2}$$

Fragility as a diagnostic. Equation 2 depends on $\Delta_B(\pi)$, which is not known before training. In this work, we use fragility as an empirical diagnostic for this degradation term: we measure whether $\mathcal{F}(\pi_H) > \mathcal{F}(\pi_L)$ and use this ordering to identify domains where high-skill BC may suffer larger finite-data degradation. Appendix A discusses when and why this diagnostic is meaningful under more general conditions. To provide intuition, we next consider a simplified setting that yields an exact threshold. Importantly, the empirical diagnostic itself is meaningful beyond this special case.

Special case: an additive error model. To make this diagnostic concrete, we consider a stylized additive error model. Consider two demonstrator policies, a low-skill policy π_L and a high-skill

policy π_H, with $J(\pi_H) > J(\pi_L)$. For a fixed training budget B, suppose that the learned BC policy trained on either demonstrator makes α_B action-level errors in expectation during rollout. We assume that these errors have additive effects on return, and that each consequential error made around demonstrator π decreases expected return by $\mathcal{F}(\pi)$ in expectation. Under these assumptions, $\Delta_B(\pi) = \alpha_B \mathcal{F}(\pi)$, or equivalently $J(\hat{\pi}_B) = J(\pi) - \alpha_B \mathcal{F}(\pi)$. Substituting into Equation 2, low-skill BC outperforms high-skill BC if and only if $\alpha_B(\mathcal{F}(\pi_H) - \mathcal{F}(\pi_L)) > J(\pi_H) - J(\pi_L)$. Since $J(\pi_H) > J(\pi_L)$ and $\alpha_B > 0$, this condition can hold only when $\mathcal{F}(\pi_H) > \mathcal{F}(\pi_L)$. Thus, greater fragility of the high-skill demonstrator is necessary in this special case. Moreover, when this fragility ordering holds, the exact threshold is $\alpha_B > (J(\pi_H) - J(\pi_L))/(\mathcal{F}(\pi_H) - \mathcal{F}(\pi_L))$. This captures why the effect is most likely in small-budget regimes: when data are limited, the effective error level α_B is larger, so errors around a more fragile high-skill policy are more likely to erase its original skill advantage. Our experiments test this diagnostic by comparing low- and high-skill BC across domains with different fragility orderings.

3.3 Methodology

In this work, we aim to better understand the effect of skill and fragility on behavioral cloning, and how these insights can be leveraged to improve BC. We consider domains with policies $\pi_1, \ldots, \pi_K$ at varying skill levels, each generating a corresponding demonstration dataset $\mathcal{D}_1, \ldots, \mathcal{D}_K$. We assume the datasets are given and labeled with the policy skill level. Our goal is to examine whether, under different data sizes, training on a low-skill dataset can yield better performance than training on a higher-skill dataset, and if this phenomenon is predicted by fragility.

We use the empirical fragility measure in Equation 1 to compute the fragility of demonstrations at different skill levels. In particular, we are interested in ordering fragility scores across skills to identify positive or negative trends between fragility and skill. We denote environments where high-skill agents are more fragile than low-skill agents as **Fragile High-Skill** and those where high-skill agents are less fragile than low-skill agents as **Resilient High-Skill**.

Given this classification, we run BC on each dataset $\mathcal{D}_k$, consisting of B demonstrations generated by π_k. Each dataset is split into training and validation sets. We initialize a model M and train it to minimize loss on the training data, repeating over multiple epochs until validation loss converges. Finally, we evaluate the task performance of the trained models at each budget and skill level.

3.4 Domains

Next, we describe the domains explored in this paper. We first start with a custom tree environment, allowing us full control to manipulate and isolate agent fragility. Next, we look at MuJoCo, consisting of 12 more complex tasks, with multidimensional states and continuous actions. Then, we move to chess, with huge human datasets available as well as a highly tuned metric of skill. Finally, we end with iRacing, providing additional human experiments in a complex, real-time setting. Overall, this gives us a set of comprehensive and broad domains to analyze the effect of low-skill BC.

Tree. We use a depth-5 binary tree environment to isolate the relationship between skill, fragility, and BC performance. Agents choose left or right at each node and receive rewards along the trajectory. We define skill by lookahead depth, so higher-skill agents can exploit delayed rewards that lower-skill agents cannot. To manipulate fragility, we vary the proportion of nodes that impose an immediate penalty but lead to a larger downstream reward. This construction creates environments where high-skill agents either enter fragile states more often or remain comparatively resilient, allowing us to directly test whether fragility predicts low-skill BC outperformance. Full construction details and an example tree are provided in Appendix B.

MuJoCo. MuJoCo is a physics engine used for simulating biomechanics and contains 11 different environments for training agents [37]. We use an existing set of pretrained models which span several combinations of environment, architecture, and skill [43]. In our results, we evaluate on every environment/architecture combination with at least two models (except for SAC-HumanoidStandup, which failed to run). We denote the highest skill model as "high-skill" and the lowest-skill model as "low-skill". Overall, this gives us 12 additional synthetic tasks to validate our results on.

Chess. We conduct our experiments in chess using two datasets. First, we use the Lichess Database [25], which contains 7.4 billion human chess games at skill levels ranging from beginners

to grandmasters. Second, we construct a dataset of Stockfish (the strongest chess engine) games, representing skill level as nodes searched. For all experiments in this domain, we use the BC training framework provided by McIlroy-Young et al. [28] [1].

iRacing. We also extend our exploration of low-skill BC to the domain of Racing. Through an academic partnership with iRacing, a high-fidelity commercial racing simulator (Figure 14), we obtained access to human driving trajectories from the qualifying laps of a Formula 4 racing series[2]. The dataset contains 10,000 complete laps on each of four tracks: Summit Point, Circuito de Navarra, Autodromo Nazionale Monza, and Road America (map diagrams are available in Appendix E). We approximate skill by lap time, with the fastest 5,000 laps marked as high-skill data and the slowest 5,000 laps marked as low-skill data.

4 Results

4.1 Tree

Following the methods from Section 3.3, we train BC models on datasets generated by agents at skill levels from $k = 1$ to 4, using data budgets from 2^8 to 2^{16}. More details on the BC training process is available in Appendix B.2.

After training, we evaluate each model by simulating 16,384 randomly generated episodes and report the average task performance. We repeat this process with proportion of fragile states ranging from 0 to 0.1. For each environment configuration, we train models with 200 random seeds. We highlight two state configurations - with 10% of nodes as fragile (Fragile High-Skill) and with 0% (Resilient High-Skill) and plot the task performance and standard error of low and high-skill BC at a variety of budgets in Figure 1.

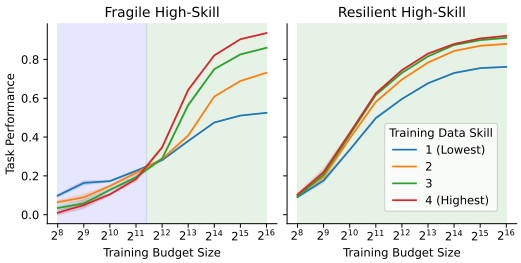


Figure 1: Performance and standard error after training BC models across budgets and skill levels in the tree domain without and with fragile nodes. Regions with low-skill training outperformance are highlighted in blue.

We see in the fragile environment, models trained on low-skill data outperform models trained on high-skill data at small budgets, while at larger budgets, training on high-skill data leads to improved task performance. However in the environment without fragile states, training on high-skill data improves performance at all budgets. Differences are significant with $p < .001$.

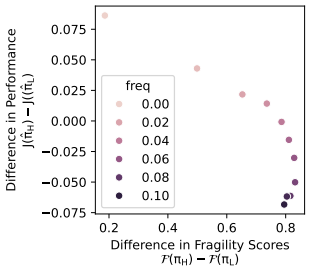


Figure 2: Comparing fragility difference and small-budget task performance by fragile state proportion in Tree.

To illustrate the potential connection between *fragility* and the benefit of low-skill training, we plot the differences in fragility and the difference in model performance at a low budget (2^{10}) with environment fragility proportions ranging from 0.0 to 0.1, and present the results in Figure 2.

We see a clear trend between the difference in fragility scores between the low and high-skill agents and the difference in task performance between models. As the fragility proportion increases, the difference in fragility score increases, and low-skill model performance begins to perform relatively better. The curve at the bottom of the plot is due to the character of the tree environments – as fragile states grow in proportion, the low and high-skill agents begin to behave identically, outpacing fragility differences.

To more directly test this relationship, we conduct additional controlled Tree experiments, varying skill and fragility separately. When demonstrator skill is held fixed, increasing fragility consistently

[1] Datasets and code for Tree, MuJoCo, and Chess are available at `https://github.com/saumikn/curricula-teaching`

[2] We are unable to release the dataset and code for iRacing due to our agreement with the provider.

increases imitation-induced degradation across all tested training budgets. Conversely, when fragility is held fixed while demonstrator skill varies, the higher-skill demonstrator produces better BC models at every tested budget. Full results are provided in Appendix B.3.

We also consider several alternative explanations for low-skill outperformance, including imitation error, state coverage, action entropy, training compute, and demonstration noise. In Appendix B.4, we show that these factors do not individually explain the low-skill advantage observed in Tree. We do not claim that fragility is uniquely predictive, but these results show that it captures information about the consequences of imitation errors which is not captured by these simpler explanations.

4.2 MuJoCo

In each MuJoCo environment, we use Equation 1 to estimate the fragility of the low-skill and high-skill agents. In this domain, we compute $V^{\pi}(s)$ directly by rolling out π from perturbed states, since we have direct access to π_L and π_H.

Then, we follow the methodology from Section 3.3 to compute the task performance of BC models trained on low-skill and high-skill data at budgets ranging from 2^2 to 2^{10} trajectories. We then evaluate each model on 10 rollouts and report the cumulative reward attained, noting whether or not the low-skill model outperforms at small budgets. We repeat this procedure for 10 seeds, and present a summary of the fragility and model performance results in Table 1. The full set of training plots, measuring model performance at multiple training budgets is available in Appendix C.

Overall, we see that fragility predicts that low-skill models will outperform high-skill models in 11/12 MuJoCo domains. All differences are significant with $p < 0.001$ except SAC-Pusher, which is significant with $p < 0.05$. [3] Breaking this down, in these eleven domains where high-skill agents are more fragile, we see low-skill models outperform in 10/11 of these domains.

Finally, we see that in SAC-InversePendulum, the one task where high-skill agents are less fragile, we indeed see that high-skill models consistently outperform. Overall, these results strongly corroborate our previous results, indicating that our notion of fragility is correlated with low-skill outperformance.

Domain	Low-Skill fragility	High-Skill fragility	Low-Skill better?	Aligns?
PPO-Swimmer	**0.08**	0.5	×	×
SAC-Ant	**19.5**	41.98	✓	✓
SAC-HalfCheetah	**3.59**	49.29	✓	✓
SAC-Hopper	**-0.95**	7.22	✓	✓
SAC-Humanoid	**1.23**	14.26	✓	✓
SAC-InvDblPend	**63.68**	160.08	✓	✓
SAC-InvPendulum	1.03	**0.0**	×	✓
SAC-Pusher	**1.22**	1.26	✓	✓
SAC-Reacher	**0.88**	0.96	✓	✓
SAC-Walker2d	**0.29**	11.48	✓	✓
TQC-HalfCheetah	**25.47**	178.9	✓	✓
TQC-Humanoid	**4.35**	58.71	✓	✓

Table 1: Fragility scores and model performance summary across 12 architectures and environments in MuJoCo.

4.3 Chess

We train two sets of models: human-like and engine-like. For human-like models, we select data at an Elo=800 (low) and Elo=2400 (high). For engine-like models, we select data at nodes=100 (low) and nodes=10k (high). Based on experiments conducted by Marco [27], these levels roughly correspond to human ratings of 1200 and 2500. In Appendix D.1, we analyze a wider set of skill levels and find qualitatively similar results.

Skill	Human	Engine
Low	0.233	0.283
High	0.288	0.222

Table 2: Fragility scores for humans and engines for the lowest and highest skill levels.

First, we compute the fragility of the human and engine agents at each skill level. We sample 4096 actions from the training datasets and use Stockfish as an approximation of $V^{\pi}(s)$ to compute Equation 1. We plot these scores in Table 2. In Appendix D.2, we show that these fragility scores are robust to weaker approximations of V and at multiple skill levels.

We see in the human dataset, fragility increases with skill, as the highest skill level has the most fragility, but in the engine dataset, fragility decreases as skill increases, and the highest skill level has

[3] Because the MuJoCo environments have substantially different reward scales, the absolute magnitude of fragility scores should only be compared within each environment.

the most resilience. [4] Thus, we denote the environment with human data as Fragile High-Skill, and the environment with engine data as Resilient High-Skill.

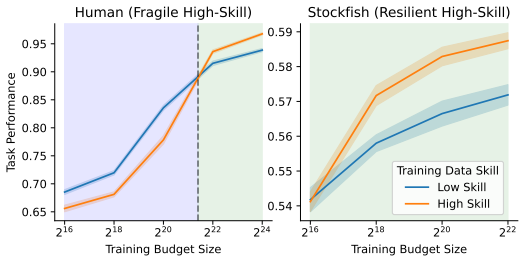


Figure 3: Performance and standard error after training BC models on human and engine data in chess by skill level. Regions with low-skill training outperformance are highlighted in blue.

To validate our synthetic results, we again follow the Section 3.3 methodology and train BC models on data across skill levels and budgets ranging from 2^{16} to 2^{24}. We evaluate the resulting models by playing 10240 chess games against a randomly-playing opponent. [5] We repeat this process with 20 random seeds and present the mean and standard error for each experiment configuration in Figure 3.

In the human-trained models, we see that with small budgets, training on low-skill data achieves the highest performance. However, with the engine-trained models, training on the highest-skill data leads to the best task performance at all data budgets. These differences are significant with $p < .001$. These results correspond exactly with agent fragility. In the fragile high-skill environment, low-skill models outperform high-skill models at small budgets. In the resilient high-skill environment, high-skill models outperform.

4.4 iRacing

Again, we follow the methodology from Section 3.3 and train BC models on low and high-skill data on each of the four tracks, at budgets ranging from 500 to 2000 laps. We evaluate each model on iRacing using a virtual joystick [1] and measure full lap completion over 20 attempts. Table 3 shows the results (budgets >1000 are omitted as all models succeed on all attempts).

We see at the smallest budget, low-skill models complete 62/80 laps across all tracks, while high-skill models complete 14/80 laps. This difference is significant with $p < 0.001$. We see that this trend holds for all tracks individually as well. For example in Summit Point, the low-skill trained model needs 500 laps to learn to complete the lap reliably, whereas the high-skill model needs 1000 laps to learn this.

		Budget		
Track Name	**Skill**	**500**	**750**	**1000**
Summit Point	Low	1.0	1.0	1.0
	High	0.0	0.9	1.0
Circuito de Navarra	Low	0.5	0.6	1.0
	High	0.0	0.35	1.0
Autodromo Nazionale Monza	Low	1.0	1.0	1.0
	High	0.6	1.0	1.0
Road America	Low	0.6	1.0	1.0
	High	0.2	1.0	1.0

Table 3: Effect of training BC models in a racing simulator by track, budget and skill. Regions with low-skill outperformance are highlighted in blue.

Our limited access to the simulator platform prevents us from directly computing fragility scores. However, we qualitatively analyzed the human demonstration data and our model trajectories and found strong evidence that low-skill agents are able to avoid fragile states more effectively than their high-skill counterparts. This is largely due to the fact that high-skill drivers tend to enter corners more aggressively and drive closer to the track edge. Appendix E.2 provides more detailed examples. Given this, we conclude that our proposed connection between agent fragility and small-budget performance also aligns in this domain.

Overall, our results across four domains provide evidence that learning BC models from low-skill data can lead to higher performance. This behavior is strongly predicted by measured fragility in the domains where it can be computed, while remaining consistent with qualitative evidence in racing.

[4] In Appendix D.2, we show an expanded plot showing this trend is monotonic across more skill levels.

[5] In Appendix D.1, we show other evaluation methods (different opponents, puzzle solving) have qualitatively the same result.

5 Skill-Based Curricula

In the previous section, we showed across domains that models trained on low-skill data can outperform high-skill models. However, those conclusions are most directly useful when practitioners must decide which data to collect for training. In this section, we consider a second common setting: train a high-performing model on an already-collected mixed-skill dataset. We show that ideas from standard curriculum learning can be applied in this setting through a simple schedule organized by demonstrator skill. Rather than proposing an optimized curriculum algorithm, our goal is to establish that demonstrator skill is a useful axis for curriculum design, complementing the more common focus on task difficulty and opening a path for more adaptive methods in future work.

5.1 Methodology

We assume access to mixed-skill dataset $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ which has already been collected. Given a budget B, each dataset $\mathcal{D}_k$ contains $\frac{B}{K}$ demonstrations from π_k. Our task is to choose a training schedule that maximizes the final model's performance, especially under the common scenario where marginal compute costs are much cheaper than data acquisition costs. We consider three possible schedules: two baselines from prior literature and one proposed curriculum that orders training by skill level.

The most common approach, **Standard BC**, is to train on the entire dataset until validation convergence. In another common approach, **Filtered BC**, the dataset is first preprocessed to include only the data at the highest skill, and then train the model until convergence. We propose a new approach, **Filtered BC-LSC** (i.e. Low-Skill Curriculum), which combines these two baselines using a standard curriculum schedule framework. We first train a model on the full dataset until convergence (Standard BC). We then iteratively remove the lowest-skill data and retrain on the remaining datasets, ending by training only on highest subset of data (Filtered BC). To assess the effect of different training curricula, we train a model under each schedule to obtain three models and evaluate them to see if applying a standard curriculum schedule over demonstrator skill can improve performance.

5.2 Results

In the main paper, we present the main set of results in tree. Our full set of results in Tree (Appendix B), MuJoCo (Appendix C.2) and chess (Appendix D.3) are qualitatively similar. We construct a tree dataset with a mixture of demonstrations from agents at four skill levels. We implement both the baselines, Standard BC and Filtered BC, as well as our proposed Filtered BC-LSC approach. We then train models with data budgets ranging from 2^8 to 2^{16} data points. After training, we evaluate each model by simulating 16,384 random episodes and report the task performance. We repeat this process with 200 random seeds and present the mean and standard error for each configuration in Figure 4.

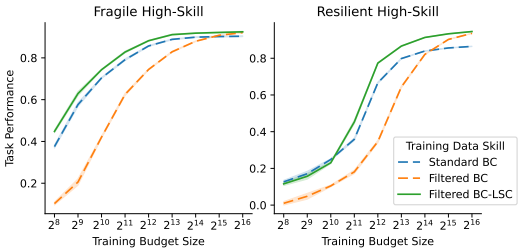


Figure 4: Performance and standard error after training BC models with low-skill curricula in the tree domain.

We see that Standard BC outperforms Filtered BC at small budgets, while Filtered BC performs better with larger budgets. Second, we see that our proposed approach minimizes the downsides of both approaches at all budgets – Filtered BC-LSC matches or exceeds Standard BC at low budgets, while performing competitively at large budgets against Filtered BC. Overall, we see a simple skill-based curriculum schedule allows us to perform well in both small-budget and large-budget regions.

6 Conclusion and Discussion

We demonstrate that low-skill demonstrations can be leveraged to improve the performance of BC models. Our findings show that BC trained on low-skill data can outperform BC trained on high-skill data in small-data regimes. This effect is strongly correlated with the *fragility* of the training datasets: it arises in domains where low-skill trajectories are more resilient to mistakes, but not in domains

where this is not the case. We also show that a simple curriculum learning–like strategy that organizes data by skill level can improve BC performance compared to standard approaches.

These results lead to several useful takeaways. When practitioners have reason to believe that high-skill policies are more fragile, they should consider training on low-skill data. While this behavior is not guaranteed, our findings suggest that practitioners can benefit from testing whether it holds in their specific domain. In scenarios where only small, mixed-skill datasets are available, emphasizing low-skill data during training can be particularly valuable, especially when compute costs are much lower than data acquisition costs. By contrast, in settings with large data budgets, low-skill data is less helpful, and practitioners should instead focus on collecting the highest-quality demonstrations.

There are a few important limitations in this work. Most importantly, fragility is a diagnostic method for understanding when models may have difficulty training on high-skill data, not a sufficient condition for low-skill outperformance.

We also do not claim that fragility is uniquely predictive of BC performance; other properties of the demonstrator may affect both imitation-induced degradation and final model performance. Our empirical fragility estimates use a uniform-random error policy as a simple pre-training proxy, and the resulting fragility values or rankings may be sensitive to this choice.

Second, estimating fragility requires either a value function, a simulator, or some other way to approximate the cost of errors. In domains where these are unavailable, the metric may be difficult to compute. This is the case in our racing domain, where our evidence for fragility is qualitative due to limited simulator access. Finally, our curriculum results should be viewed as a simple proof of concept rather than an optimized training strategy. We show that ordering data by demonstrator skill can improve BC performance, but more adaptive schedules may be needed to determine when and how much low-skill data should be used in a given domain.

There are also several directions for future work. We showed that a simple curriculum schedule can lead to increased performance, but more complex schedules may show additional benefits, especially developed jointly with online algorithms that dynamically determine which data to acquire. Second, methods for automatically estimating human strategy complexity in each domain would better guide practitioners in deciding which skill data to collect. Finally, it would be valuable to explore whether the benefits of training on low-skill data extend beyond BC to offline RL or other learning paradigms.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported in part by a TRIADS Graduate Fellowship at Washington University in St. Louis and an OpenAI Superalignment Fast Grant. The authors declare no competing interests.

References

[1] pyvjoy: Python bindings for vjoy, 2025. URL `https://github.com/tidzo/pyvjoy`.

[2] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[3] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[5] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[6] Annie S. Chen, Alec M. Lessing, Yuejiang Liu, and Chelsea Finn. Curating demonstrations using online experience, 2025. URL `https://arxiv.org/abs/2503.03707`.

[7] Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conference on robot learning*, pages 1262–1277. PMLR, 2021.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[9] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18353–18363, 2020.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[11] 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*, pages 158–168. PMLR, 2022.

[12] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[13] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

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

[15] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

[16] Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators, 2025. URL `https://arxiv.org/abs/2502.08623`.

[17] Jeff Howe et al. The rise of crowdsourcing. *Wired magazine*, 14(6):176–183, 2006.

[18] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[19] Anssi Kanervisto, Joonas Pussinen, and Ville Hautamäki. Benchmarking end-to-end behavioural cloning on video games. In *2020 IEEE conference on games (CoG)*, pages 558–565. IEEE, 2020.

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

[21] Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Should i use offline rl or imitation learning? 2022. URL `https://bair.berkeley.edu/blog/2022/04/25/rl-or-bc/`.

[22] Vladislav Kurenkov and Sergey Kolesnikov. Showing your offline reinforcement learning work: Online evaluation budget matters. In *International Conference on Machine Learning*, pages 11729–11752. PMLR, 2022.

[23] Vitaly Kurin, Sebastian Nowozin, Katja Hofmann, Lucas Beyer, and Bastian Leibe. The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*, 2017.

[24] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[25] Lichess. Lichess database, 2025. URL `database.lichess.org`.

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

[27] Meloni Marco. Stockfish and lc0, test at different number of nodes, 2021. URL `https://www.melonimarco.it/en/2021/03/08/stockfish-and-lc0-test-at-different-number-of-nodes/`.

[28] Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1677–1687, 2020.

[29] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[30] Mitchell J Nathan, Kenneth R Koedinger, Martha W Alibali, et al. Expert blind spot: When content knowledge eclipses pedagogical content knowledge. In *Proceedings of the third international conference on cognitive science*, volume 644648, pages 644–648, 2001.

[31] Rong-Jun Qin, Xingyuan Zhang, Songyi Gao, Xiong-Hui Chen, Zewen Li, Weinan Zhang, and Yang Yu. Neorl: A near real-world benchmark for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:24753–24765, 2022.

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

[33] Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, and Tim Genewein. Grandmaster-level chess without search. *arXiv preprint arXiv:2402.04494*, 2024.

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

[35] Herbert Alexander Simon. *Models of bounded rationality: Empirically grounded economic reason*, volume 3. MIT press, 1997.

[36] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

[37] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ internatxional conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[38] Jennifer Wortman Vaughan. Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18(193):1–46, 2018.

[39] Andrew Wagenmaker and Aldo Pacchiano. Leveraging offline data in online reinforcement learning. In *International Conference on Machine Learning*, pages 35300–35338. PMLR, 2023.

[40] Lingguang Wang, Carlos Fernandez, and Christoph Stiller. High-level decision making for automated highway driving via behavior cloning. *IEEE Transactions on Intelligent Vehicles*, 8 (1):923–935, 2022.

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

[42] Ruiyi Wang, Haofei Yu, Wenxin Zhang, Zhengyang Qi, Maarten Sap, Graham Neubig, Yonatan Bisk, and Hao Zhu. Sotopia-π: Interactive learning of socially intelligent language agents. *arXiv preprint arXiv:2403.08715*, 2024.

[43] Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. Minari, September 2024. URL `https://doi.org/10.5281/zenodo.13767625`.

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

[45] Jesse Zhang, Brian Cheung, Chelsea Finn, Sergey Levine, and Dinesh Jayaraman. Cautious adaptation for reinforcement learning in safety-critical settings. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, 2020.

[46] Tianxiang Zhao, Wenchao Yu, Suhang Wang, Lu Wang, Xiang Zhang, Yuncong Chen, Yanchi Liu, Wei Cheng, and Haifeng Chen. Skill disentanglement for imitation learning from suboptimal demonstrations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3513–3524, 2023.

[47] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL `https://arxiv.org/abs/2304.13705`.

A Additional Discussion on the Connection between Fragility and BC

This appendix clarifies when fragility is expected to help predict whether BC trained on low-skill demonstrations can outperform BC trained on high-skill demonstrations. The main text gives the exact degradation condition and an additive special case. Here, we characterize broader settings in which fragility is informative as a pre-training proxy for imitation-induced degradation, and discuss cases where the definition may need to be refined to account for error rates, error locations, or rollout-level compounding.

Exact condition. Let $\hat{\pi}_{\pi,B}$ denote the policy learned by BC from a budget B of demonstrations generated by policy π, and define the imitation-induced degradation as $\Delta_B(\pi) = J(\pi) - \mathbb{E}[J(\hat{\pi}_{\pi,B})]$, where the expectation is over sampled demonstrations and training randomness. Consider a low-skill demonstrator π_L and a high-skill demonstrator π_H, with $J(\pi_H) > J(\pi_L)$. Low-skill BC outperforms high-skill BC exactly when

$$\Delta_B(\pi_H) - \Delta_B(\pi_L) > J(\pi_H) - J(\pi_L).$$

Thus, the effect occurs when the additional degradation from imitating the high-skill demonstrator exceeds its original skill advantage. Fragility is useful because it estimates the cost of action-level imitation errors around the states visited by a demonstrator. It is therefore informative when differences in error cost explain a substantial part of the degradation gap.

Case 1: Common effective error level. The cleanest case is the additive model discussed in the main text. Suppose that, at budget B, the learned policies have the same effective level of consequential imitation errors, denoted by α_B, and that each error around demonstrator π decreases expected return by $\mathcal{F}(\pi)$, with losses adding across errors. Then $\Delta_B(\pi) = \alpha_B \mathcal{F}(\pi)$ under this model. Substituting into the exact condition, low-skill BC outperforms high-skill BC if and only if

$$\alpha_B\big(\mathcal{F}(\pi_H) - \mathcal{F}(\pi_L)\big) > J(\pi_H) - J(\pi_L).$$

This yields a direct threshold. If $\mathcal{F}(\pi_H) \leq \mathcal{F}(\pi_L)$, low-skill BC cannot outperform high-skill BC under this model. If $\mathcal{F}(\pi_H) > \mathcal{F}(\pi_L)$, then low-skill BC outperforms whenever

$$\alpha_B > \frac{J(\pi_H) - J(\pi_L)}{\mathcal{F}(\pi_H) - \mathcal{F}(\pi_L)}.$$

This captures the small-budget intuition: when data are limited, the effective error level is larger, so errors around a fragile high-skill demonstrator are more likely to erase its initial skill advantage.

Case 2: Unequal but comparable error levels. Fragility can remain informative even when high- and low-skill BC have different error levels, provided that these differences do not dominate the

difference in error cost. Suppose the learned policies have effective error levels $\alpha_{H,B}$ and $\alpha_{L,B}$. Under the same additive loss model, $\Delta_B(\pi_H) = \alpha_{H,B}\mathcal{F}(\pi_H)$ and $\Delta_B(\pi_L) = \alpha_{L,B}\mathcal{F}(\pi_L)$. Low-skill BC outperforms high-skill BC exactly when

$$\alpha_{H,B}\mathcal{F}(\pi_H) - \alpha_{L,B}\mathcal{F}(\pi_L) > J(\pi_H) - J(\pi_L).$$

Equivalently, the condition can be written as

$$\alpha_{H,B}\big(\mathcal{F}(\pi_H) - \mathcal{F}(\pi_L)\big) > J(\pi_H) - J(\pi_L) + (\alpha_{L,B} - \alpha_{H,B})\mathcal{F}(\pi_L).$$

This form makes explicit how unequal error levels affect the role of fragility. When $\alpha_{H,B} < \alpha_{L,B}$, the right-hand side includes an additional positive term, so the fragility gap must compensate not only for the original skill gap $J(\pi_H) - J(\pi_L)$ but also for the lower error level of high-skill BC. Thus, high-skill BC can be imitated more accurately and still underperform, but only when $\mathcal{F}(\pi_H) - \mathcal{F}(\pi_L)$ is sufficiently large. When $\alpha_{H,B} > \alpha_{L,B}$, the right-hand side is smaller, reflecting the fact that high-skill BC suffers both from more frequent errors and from potentially more costly errors. In this case, a positive fragility gap is not necessary for low-skill BC to outperform, but it further strengthens the effect. More generally, the ordering $\mathcal{F}(\pi_H) > \mathcal{F}(\pi_L)$ alone is not sufficient unless the above inequality holds. Fragility is therefore most predictive when $\alpha_{H,B}$ and $\alpha_{L,B}$ are close, or when the fragility gap is large relative to both the skill gap and the error-level difference.

Case 3: State-dependent errors. In practice, imitation errors may not be uniformly distributed across states. Let $\epsilon_{\pi,B}(s)$ denote the probability that the BC model trained from demonstrator π with budget B makes an error at state s, and let $c_\pi(s) = V^\pi(s) - \mathbb{E}_{a\sim\hat{\pi}_{\mathrm{err}}(\cdot|s)}[Q^\pi(s,a)]$ denote the expected cost of such an error under the proxy error model used to compute fragility. Under a state-dependent additive model with normalized state visitation distribution ζ^π over horizon T, the degradation is $\Delta_B(\pi) = T\mathbb{E}_{s\sim\zeta^\pi}[\epsilon_{\pi,B}(s)c_\pi(s)]$. By contrast, the fragility measure used in the paper is the unweighted average error cost, $\mathcal{F}(\pi) = \mathbb{E}_{s\sim\zeta^\pi}[c_\pi(s)]$. Thus, fragility directly captures degradation when $\epsilon_{\pi,B}(s)$ is constant along the demonstrator trajectory. More generally, it remains informative when error probabilities are not strongly anti-correlated with error costs. For example, if high-cost states are at least as likely to be misimitated as low-cost states, then larger fragility indicates that the policy visits states where imitation errors are more damaging. This is a plausible regime for low-budget BC, where decision-sensitive states may be both harder to imitate and more costly to get wrong.

Case 4: Monotone non-additive degradation. The above cases assume additive error costs, but the diagnostic role of fragility can also be formalized for non-additive degradation. Suppose the imitation-induced degradation can be represented as $\Delta_B(\pi) = g(e_B(\pi), \mathcal{F}(\pi))$, where $e_B(\pi)$ is an effective error level and g is a common function that is nondecreasing in both arguments and strictly increasing in $\mathcal{F}(\pi)$ when $e_B(\pi)$ is fixed. This captures settings where errors compound, interact across time, or induce nonlinear losses after the learned policy leaves the demonstrator distribution. If $e_B(\pi_H) = e_B(\pi_L)$, then $\mathcal{F}(\pi_H) > \mathcal{F}(\pi_L)$ implies $\Delta_B(\pi_H) > \Delta_B(\pi_L)$. If $e_B(\pi_H) \geq e_B(\pi_L)$, the same fragility ordering is sufficient to imply larger degradation for high-skill BC. However, low-skill BC outperforms high-skill BC only under the stronger condition $g(e_B(\pi_H), \mathcal{F}(\pi_H)) - g(e_B(\pi_L), \mathcal{F}(\pi_L)) > J(\pi_H) - J(\pi_L)$. When $e_B(\pi_H) < e_B(\pi_L)$, the fragility ordering alone is not sufficient: the larger fragility of π_H must be large enough to overcome the lower effective error level of high-skill BC. Thus, beyond the additive model, fragility remains informative through its monotone contribution to degradation, but it does not by itself determine outperformance unless error levels are equal or otherwise controlled.

When the current fragility measure may be insufficient. The current fragility definition is intentionally simple: it measures the average cost of proxy action-level errors along the demonstrator trajectory. This makes it computable before training a BC model, but it also means that fragility alone need not fully predict imitation-induced degradation. For example, if high-skill BC has a much lower error rate than low-skill BC, then high-skill BC may still outperform even when $\mathcal{F}(\pi_H) > \mathcal{F}(\pi_L)$. If errors concentrate in low-cost states for the high-skill policy but in high-cost states for the low-skill policy, then the unweighted average error cost may be misleading. Similarly, if the proxy error model differs substantially from the errors made by the learned policy, the measured fragility may not align with the true degradation.

Possible refinements. When additional information is available, fragility can be refined to better approximate imitation-induced degradation. If state-dependent error probabilities can be estimated, one can use an error-weighted degradation proxy,

$$\mathcal{D}_B(\pi) = T\mathbb{E}_{s\sim\zeta^\pi}\left[\epsilon_{\pi,B}(s)c_\pi(s)\right],$$

which matches the state-dependent additive model above. If the learned model's action-confusion distribution can be estimated, the proxy error policy can be replaced by a learner-aware error policy $q_{\pi,B}(a|s)$:

$$\mathcal{F}_{q,B}(\pi) = \mathbb{E}_{s\sim\zeta^\pi}\left[V^\pi(s) - \mathbb{E}_{a\sim q_{\pi,B}(\cdot|s)}[Q^\pi(s,a)]\right].$$

If the main concern is compounding errors, one could instead define a rollout-based fragility measure by injecting errors according to a specified noise model and measuring the resulting drop in return. These refinements require additional information, such as a trained model, validation errors, a simulator, or a more accurate model of likely imitation mistakes. In this paper, we use the simpler pre-training fragility measure to test whether the ordering $\mathcal{F}(\pi_H) > \mathcal{F}(\pi_L)$ is already a useful empirical diagnostic across domains.

Summary. The exact condition for low-skill BC to outperform high-skill BC is that the degradation gap $\Delta_B(\pi_H) - \Delta_B(\pi_L)$ exceeds the original skill gap $J(\pi_H) - J(\pi_L)$. Fragility helps anticipate when this condition may hold by measuring how costly imitation errors are around the states visited by a demonstrator. It is most directly predictive when error levels are comparable, when error probabilities are not strongly anti-correlated with error costs, or when degradation is monotone in both effective error level and fragility. In more complex settings, fragility remains useful as a pre-training diagnostic, but it can be refined using estimates of error rates, error locations, learner-specific action confusions, or rollout-level degradation.

B Tree Experiments

B.1 Domain Construction

As described in Section 4, each tree task is composed of a binary tree structure. Each node in the tree is randomly assigned a reward between 0 and 1. We also designate a subset of the nodes as *fragile*. These nodes have an extra penalty for reaching them (-8), but an even larger reward (10) after making the correct action in this state. However, if the wrong action is taken in this state, no extra reward is given, and the agent simply suffers the initial penalty. An example initialization of a tree structure is visualized in Figure 5, with an example of a fragile state and extra-reward state highlighted in orange and green, respectively.

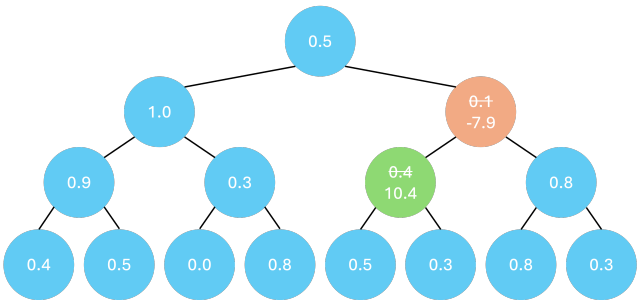


Figure 5: An example tree structure, with a fragile state highlighted in orange, and the extra reward state highlighted in green.

With this definition of state fragility, we can then characterize entire tree domains by the proportion of states which are randomly assigned as fragile, with this proportion ranging from 0 to 1.

B.2 Model Training

In Section 4.1 we conduct all experiments with a tree depth of 5. To train our BC models, we construct datasets which contain data points on the action an agent would take at a randomly generated state. We then create multiple datasets at budgets ranging from 2^8 to 2^{16}, at skill levels ranging from $k = 1$ to 4, with fragility proportions ranging from 0.0 to 0.1.

On each generated model, we train a BC model. This model consists of a simple fully-connected neural network with 1 layer, and 16 nodes. We allocated 75% of the dataset for training, and the rest for validation. We trained the neural network on the training dataset using the ADAM optimizer with

the initial learning rate of 0.1, and trained until validation loss converged (measured using a patience level of 5 epochs). The full training code is available at https://github.com/saumikn/low-skill-learning

The full set of results is relatively compute-inexpensive in this domain. We performed our experiments on our compute cluster with CPUs only. Using about 500 CPUs and 6TB of memory, the experiments in this section (repeating over all experimental seeds) took under one day in parallel.

For each generated model, we evaluate its task performance by generating an additional set of 2^{14} tree environments using the same configuration that the model was originally trained on, and evaluate the expected performance of trained agent in these environments, by computing the average sum of rewards encountered by the agent.

In Section 4.2, we largely followed the same training paradigm as in Section 4.1, with two differences.

(1) First, we constructed training datasets which contained a mixture of actions across skill levels. For instance, with Standard BC with a budget of 2^{10}, the dataset actually contained 2^8 data points at $k = 1$, 2^8 data points at $k = 2$, 2^8 data points at $k = 3$, and 2^8 data points at $k = 4$, split into 75% training and 25% validation as before.

(2) We construct a training curricula which consists of a series of training datasets. In each step, we train the model until validation convergence. In the next step, we take the best model from the previous step, and re-run training on the next dataset, after entirely resetting the learning rate and optimizer state.

B.3 Controlling Skill and Fragility

The experiments in Section 4.1 vary the proportion of fragile states, which can change both the skill and fragility of the resulting demonstrators. To separate these effects, we conduct additional controlled experiments where we parameterize a fragile state using three values: the net reward r obtained when the state is navigated correctly, the return gap g between the correct and incorrect actions, and the probability p that a node is designated as fragile. The original fragile Tree environment corresponds to $r = 2$, $g = 10$, and $p = 0.1$. Varying these parameters together with the demonstrator lookahead depth k allows us to construct demonstrators with similar skill but different fragility, or similar fragility but different skill.

Directly measuring imitation-induced degradation. We first directly measure the imitation-induced degradation term from Equation 2,

$$\Delta_B(\pi) = J(\pi) - \mathbb{E}\left[J(\hat{\pi}_{\pi,B})\right].$$

Unlike fragility, estimating this quantity requires training and evaluating the resulting BC policy. In the controlled Tree domain, however, we have access to both the demonstrator and learned policies, allowing us to directly measure this degradation.

We fix lookahead depth at $k = 4$, fragile-state probability at $p = 0.1$, and successful net reward at $r = 2$, while varying the error gap over $g \in \{4, 6, 8, 10, 12, 14\}$. This produces demonstrators with approximately the same skill, $J(\pi) \approx 4.726$, while fragility varies from 0.878 to 2.024. Table 4 reports imitation-induced degradation as a percentage of demonstrator reward, with each value averaged over 40 random seeds.

Fragility	$B = 64$	$B = 256$	$B = 1{,}024$	$B = 4{,}096$	$B = 16{,}384$
0.878	44.4%	34.6%	16.3%	5.9%	2.6%
1.106	55.2%	44.7%	27.5%	8.5%	3.3%
1.335	63.9%	55.4%	38.2%	10.4%	4.0%
1.565	74.3%	64.7%	47.1%	13.1%	4.9%
1.794	80.8%	73.0%	57.7%	16.6%	5.8%
2.024	85.4%	82.2%	65.3%	20.6%	6.6%

Table 4: Imitation-induced degradation as a percentage of demonstrator reward when demonstrator skill is held approximately fixed and fragility is varied. Each value is averaged over 40 random seeds.

Imitation-induced degradation increases monotonically with fragility at every tested budget. At the three smallest budgets, increasing fragility from 0.878 to 2.024 raises the fraction of demonstrator performance lost from 44.4% to 85.4%, from 34.6% to 82.2%, and from 16.3% to 65.3%, respectively.

At the largest budget, BC is close to the demonstrator in every condition, but degradation still increases from 2.6% to 6.6%.

At the smallest budget, the least and most fragile conditions also have similar validation action accuracy, 56.6% and 55.3%, respectively, despite degradation increasing from 44.4% to 85.4%. Thus, the substantially larger degradation is not explained by a comparable increase in imitation error. These results provide a direct connection between fragility and the degradation term used in Equation 2, rather than only between fragility and final BC performance.

Separating skill and fragility. The experiment above also provides a matched-skill comparison. Demonstrator reward remains approximately fixed while fragility varies, and greater fragility produces greater imitation-induced degradation and worse final BC performance. We next construct the opposite comparison, where fragility is approximately matched while demonstrator skill varies.

We compare a lower-skill demonstrator with $k = 2$, $r = 4$, and $g = 12$ against a higher-skill demonstrator with $k = 3$, $r = 4$, and $g = 4$, with $p = 0.1$ in both cases. As shown in Table 5, the two demonstrators have nearly identical fragility, differing by only 0.0011, while their expected rewards differ by 1.10.

Demonstrator	**Lookahead** k	**Net Reward** r	**Error Gap** g	**Skill** $J(\pi)$	**Fragility**
Lower-Skill	2	4	12	5.11	0.9975
Higher-Skill	3	4	4	6.21	0.9986

Table 5: Two Tree demonstrators with approximately matched fragility but different skill.

We train BC models on both demonstrators across the same set of budgets. Table 6 reports the resulting task performance, with each comparison averaged over more than 40 random seeds.

Budget	**Lower-Skill BC Return**	**Higher-Skill BC Return**	**Higher − Lower**
64	1.90 (37%)	3.53 (57%)	+1.64
256	2.37 (46%)	4.35 (70%)	+1.97
1,024	2.89 (57%)	5.55 (89%)	+2.66
4,096	4.11 (81%)	5.98 (96%)	+1.87
16,384	4.70 (92%)	6.15 (99%)	+1.45

Table 6: BC task performance when demonstrator fragility is approximately matched while skill varies. Percentages in parentheses give BC performance as a percentage of the corresponding demonstrator's reward.

BC trained on the higher-skill demonstrator outperforms BC trained on the lower-skill demonstrator at every tested budget. Thus, when the higher-skill demonstrator no longer has a fragility disadvantage, the low-skill advantage observed in the original Tree experiments disappears. Together with the matched-skill experiment above, these results separate the effects of skill and fragility and provide a more direct test of the relationship between fragility and imitation-induced degradation.

B.4 Alternative Explanations

In addition to fragility, there are several other differences between low-skill and high-skill demonstrations which could potentially explain low-skill outperformance. Here, we consider imitation error, state coverage, action entropy, training compute, and demonstration noise.

Imitation error. First, we investigate the action error rates of low-skill and high-skill models in the Tree environment with 10% fragile nodes. At a small budget of 2^8, the model trained on low-skill data has an error rate of 47%, while the model trained on high-skill data has an error rate of 43%. Thus, imitation error alone predicts the opposite ordering from task performance: the low-skill model makes more action errors despite achieving higher task performance.

State coverage and action entropy. The Tree construction also controls for several simple properties of the demonstration data. Training states are randomly generated from the same state distribution at every demonstrator skill level, so differences in state coverage cannot explain the low-skill advantage. In addition, all demonstrator policies are deterministic, giving zero conditional action entropy

at every skill level. By left-right symmetry of the randomly generated trees, marginal left and right action frequencies are also equal in expectation across skill levels. Thus, neither state coverage nor action entropy explains the performance ordering in this controlled domain.

B.5 Training Compute Budgets

While training until convergence is a reasonable assumption in domains with expensive data acquisition costs, one question may arise – are we spending too much in compute costs to get improved performance at the low-skill, low-budget region? In Table 7, we find that this is not true – in fact, low-skill models take fewer epochs to converge than high-skill models at the low-budget region.

Budget	Low-Skill Epochs	High-Skill Epochs
2^8	223.6	246.0
2^{10}	161.9	189.2
2^{12}	821.6	846.5
2^{14}	1375.6	635.6
2^{16}	1020.7	470.3

Table 7: Comparison of Low-Skill vs. High-Skill Epochs across budgets.

B.6 BC from Noisy Demonstrations

Instead of using a standard BC approach for modeling low-skill and high-skill, recent works have proposed using modified methods which can be used to learn models of low-skill or noisy demonstrations. In general, our work focuses on behavioral cloning, and we leave extensions to other policy classes to future work. That said, we expect the core phenomenon to generalize beyond BC.

To empirically validate this, we have performed one additional experiment, replicating our main result in the small-budget region using the BC from Noisy Data approach taken by Sasaki and Yamashina [34] instead of a standard BC approach. The link to the figure is available in Figure 6. Overall, our results are qualitatively very similar to our main results in the small-budget region - models trained on low-skill data with small budgets outperform in fragile environments, and models trained on high-skill data outperform in the resilient domains. This aligns with our intuition behind the mechanism of low-skill outperformance, because the low-skill (and high-skill) data is not noisy, but comprises different strategies, prior methods which are designed to learn from noisy methods wouldn't provide any benefits under our regime.

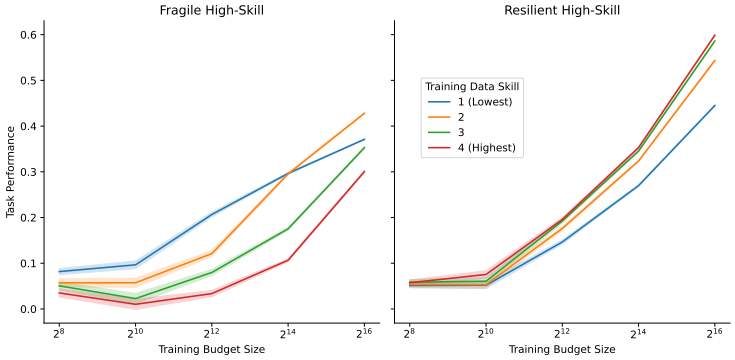


Figure 6: Effect of training BC models across budgets and skill levels in tree, using the BCND method from Sasaki and Yamashina [34]

C MuJoCo Experiments

C.1 Fragility and Model Performance

In Section 4.2, we replicated our experiments from Tree in a more complex set of environments. In Table 1, we printed a summary of the results of low-skill and high-skill training. In Figure 7,

we provide a more comprehensive set of results showing task performance for each environment, demonstrator skill, and training budget.

Overall, we see in nearly all domains, the model trained on low-skill data performs consistently better at low-skill budgets. The only exceptions are PPO-Swimmer and SAC-InvertedPendulum.

We performed our experiments on our compute cluster with CPUs only. Using our cluster with 500 CPUs and 6TB of memory, the experiments in this section (repeating over all experimental seeds) took approximately one week in parallel. In additional, additional experimental effort which did not go into the final paper took approximately two weeks.

C.2 Training with Low-Skill Curricula

In each MuJoCo environment, we construct a mixed-skill dataset with the demonstrations from the lowest and high-skill agents. We implement both the baselines, Standard BC and Filtered BC, as well as our proposed Filtered BC-LSC approach. We then train models with data budgets ranging from 2^2 to 2^{10} data points. After training, we evaluate each model by simulating 10 rollouts and report the task performance. We repeat this process with 10 random seeds and present the mean and standard error for each configuration in Figure 8.

We again see that Standard BC generally outperforms Filtered BC at small budgets and worse at higher budgets, though this is somewhat obscured by a few domains where Filtered BC outperforms at all budgets. More importantly, we see that our proposed approach of Filtered BC-LSC significantly outperforms both baselines in at least one budget value in 9/12 MuJoCo environments.

C.3 Equal Error Rates

As we did in Tree, we perform an additional analysis to investigate whether higher error rates explains why models trained on low-skill data tend to outperform models trained on high-skill data. We train a series of models using a small budget of 2^4 data points, and report the final converged MSE of each model in the validation set in Table 8

Environment	Low-Skill	High-Skill
Pusher-SAC	0.006	0.002
Humanoid-TQC	0.018	0.020
Reacher-SAC	0.001	0.002
HalfCheetah-SAC	0.098	0.043
Hopper-SAC	0.004	0.009
Ant-SAC	0.068	0.062
Walker2d-SAC	0.011	0.003
InvertedPendulum-SAC	0.000	0.005
InvertedDoublePendulum-SAC	0.000	0.000
Humanoid-SAC	0.023	0.015
HalfCheetah-TQC	0.095	0.028
Swimmer-PPO	0.001	0.006

Table 8: Comparison of low-skill and high-skill values across environments.

Overall, we find very mixed results, with 6/12 environments showing a higher MSE for the low-skill model and the other half showing a higher MSE for the high-skill model. Overall, these results are inconclusive, indicating that is unlikely for error rates to be the reason why high-skill models perform worse.

D Chess Experiments

In Section 4.3, we repeat our experiments using Chess instead of a synthetic domain. Broadly, we follow the experimental decisions taken by McIlroy [28], which was itself inspired by the training process of the Leela chess engine. We collect human trajectories from the online platform Lichess.org. Each action is labeled according to the skill level of the human who made it, where 400 is the lowest possible skill, and 3400 is roughly the highest skill (though technically the range is uncapped).

In addition to the human data, we construct a synthetic demonstration dataset using the leading chess engine: Stockfish. We have Stockfish play matches against itself, varying the node depth of Stockfish search to simulate skill levels.

D.1 Model Training

Stockfish Depth	Low-Skill Fragility	High-Skill Fragility
1	0.31235	0.27070
10	0.31217	0.27074
100	0.30161	0.25938
1000	0.29605	0.24837
10000	0.30086	0.25364
100000	0.30203	0.25337

Table 9: Fragility scores at varying Stockfish depths.

Each state in the training dataset is represented as a $112 \times 8 \times 8$ matrix, and each action is represented as a one-hot encoded vector of length 1858. Using the transformer-based architecture of the Leela T74 architecture, we randomly initialize the weights of the model before training the BC model on the human or Stockfish data until convergence using SGD with an initial learning rate of 0.1.

Once the model is trained, we evaluate its task performance by having the trained model play against a completely random (i.e untrained) chess model until the end of the game. Our task score is computed as the average result over a match of independent 10240 chess games.

The full set of results is available in Figure 9.

We performed our experiments on our compute cluster with CPUs only. Using our cluster with 500 CPUs and 6TB of memory, the experiments in this section (repeating over all experimental seeds) took approximately two week in parallel. In additional, additional experimental effort which did not go into the final paper took approximately four weeks.

In addition to evaluating the model against a completely random opponent, we also evaluate two other approaches, playing against a 800-rated opponent (a model trained to convergence on 256M chess positions at the 800 level), and evaluate the model on a set of puzzles. We present these results in Figure 10 and 11.

Skill	Human	Engine
Low	0.233	0.283
	0.252	
↓	0.264	0.243
	0.280	
High	0.288	0.222

Table 10: Fragility scores for humans and engines across skill levels

D.2 Fragility Computation

In Table 2, we showed fragility scores for low-skill and high-skill humans, evaluated using the highest depth stockfish. In Table 10, we show the fragility against other skill agents, and show that

these trends still hold, where humans have lower fragility at lower skills, while engines have higher fragility at higher skills. In Table 9, we also show fragility scores for low-skill and high-skill humans, evaluated using multiple depths of stockfish, and we find that the trends hold at all depths.

D.3 Training with Low-Skill Curricula

Using both the human and Stockfish dataset, we implement both baselines, Standard BC and Filtered BC, as well as our proposed Filtered BC-LSC approach, with data budgets ranging from 2^{16} to 2^{24}. We evaluate the resulting models by playing 10240 chess games against a randomly-playing opponent, and report the average win rate of the model. We repeat this process with 20 random seeds, and present the results in Figure 12.

Overall, we see in both configurations that Standard BC outperforms Filtered BC at small budgets, while Filtered BC performs better with larger budgets. This aligns with existing work which suggests Filtered BC is best when budgets are large enough to keep sufficient data after filtering. Second, we see that our proposed approach minimizes the downsides of both approaches at all budgets – Filtered BC-LSC matches or exceeds Standard BC at low budgets, while performing competitively at large budgets against Filtered BC. Overall, we see a simple curriculum strategy leveraging low and high-skill demonstrations allows us to perform well in both small and large-budget regions.

D.4 Equal Error Rates

As in tree and MuJoCo, we conduct an analysis to investigate the action error rates of low-skill and high-skill models to determine if imitation error rates helps explain low-skill overperformance in Chess. On our set of models trained on 800, 1600, and 2400 human data at a budget of 2^{16}, we found that the 800-trained model had a policy accuracy of 11.0%, the 1600-trained model had an accuracy of 12.6%, and the 2400-trained model had an accuracy of 10.0%. From these results, we can also conclude that the small changes in error rates are unlikely to lead to low-skill outperformance. In particular, we see that the 1600-trained model has a higher accuracy than the 800-trained model, but still performs worse empirically at this data budget.

E Racing Experiments

E.1 Model Training

Through an academic partnership with a commercial racing simulator platform, we obtained access to human driving trajectories from a Formula 4 racing series. Specifically, we use a dataset of 10000 complete laps on each of four different tracks: Summit Point Raceway, Road America, Circuito de Navarra, and Autodromo Nazionale Monza. A map of each track is shown in Figure 13, and a screenshot of the simulator is shown in Figure 14.

We approximate skill by ordering the laps by time, with the fastest 5000 laps representing high-skill data and the slowest 5000 laps representing low-skill data. We evaluate our models using a virtual joystick [1] that forwards the controls to the simulator.

We train a deep MLP model on state variables (e.g., car speed, heading, etc.) to predict four driving controls (steering, throttle, brake, and gear) at 60Hz. We use action chunking [47] to ensure stability and accommodate inference latency, predicting 16 future actions at a time.

E.2 Additional Analysis

We performed an extensive qualitative analysis of off-track and loss-of-control incidents experienced by both our high-skill and low-skill models. In nearly all examples we studied, the failure occurred because the model made decisions that caused it to enter a fragile state.

Figure 15 shows an example of a failure on Road America. As the trajectory shows, the model makes a tighter turn and eases off the throttle too late, pushing it to the outer boundary of the track and resulting in a more fragile state. In contrast, a successful trajectory takes a wider turn and eases the throttle early (dropping to 0.4), allowing it to complete the turn safely.

Figure 16 shows a typical example where the low-skill model makes a decision that leads to a less fragile state than the high-skill model, in this case at a challenging turn on Summit Point. Here, the high-skill model is too aggressive on the entry to the turn and does not brake enough (0.6) compared to the low-skill model (0.8). This causes the high-skill model to exceed the track boundary and eventually lose control on the grass.

Impact Statement

This work uses human-generated data from two primary sources: a public chess database and a private racing simulator dataset. The chess data is drawn from the Lichess Player Database, which consists of anonymized games that are publicly available for research. The racing data was obtained through an academic partnership with a commercial simulator platform. This dataset was handled in accordance with our data sharing agreement with the provider, and all data was anonymized to protect user privacy. The racing platform was not involved with the research direction or conclusions of this paper, and there are no conflicts of interest between the platform and this work.

Our research aims to create more robust and reliable models through behavioral cloning, and we do not foresee direct negative societal impacts. On the contrary, our finding that low-skill data can improve model resilience may contribute to the development of safer AI systems, as it encourages learning from a wider, more cautious range of behaviors rather than solely from "expert" demonstrations that may operate close to failure margins.

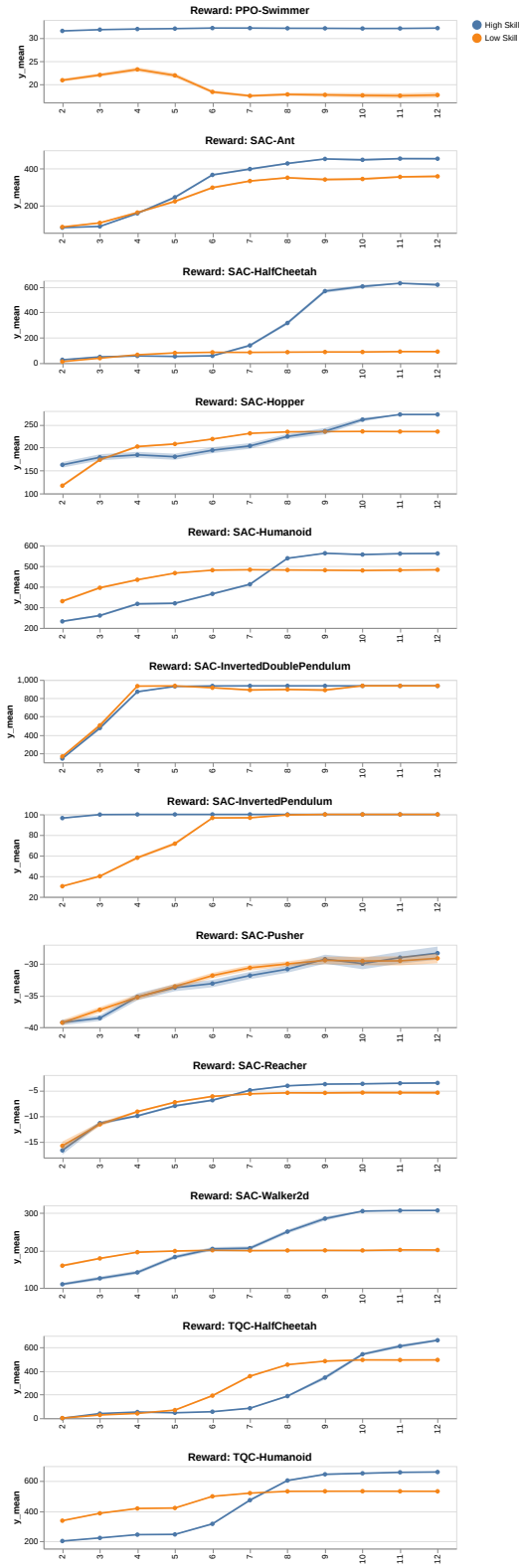


Figure 7: Effect of training BC models across budgets and skill levels in MuJoCo

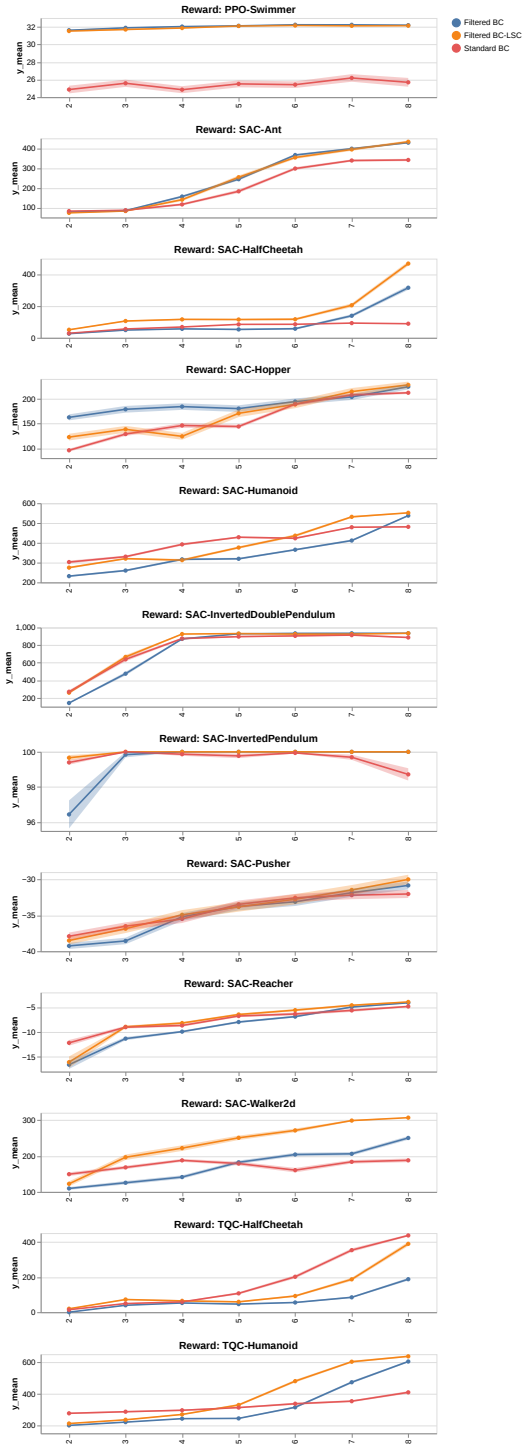


Figure 8: Effect of skill-based curricula across budgets and skill levels in MuJoCo

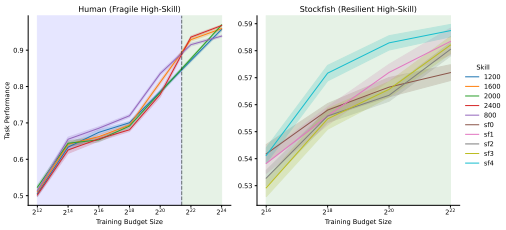


Figure 9: Effect of training BC models on human and engine demonstrations in chess by skill level. Budget regions where low-skill training outperforms high-skill is highlighted in blue.

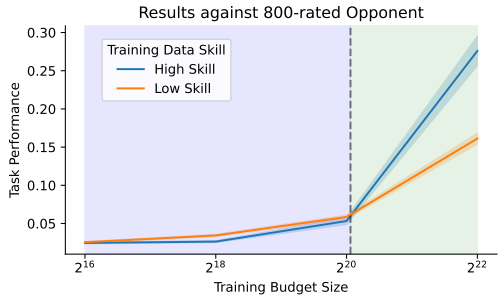


Figure 10: Effect of training BC models on human and engine demonstrations in chess by skill level, evaluating against an 800-rated opponent. Budget regions where low-skill training outperforms high-skill is highlighted in blue.

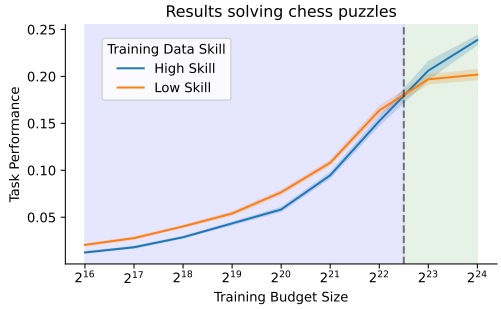


Figure 11: Effect of training BC models on human and engine demonstrations in chess by skill level, evaluated using puzzle solve rate. Budget regions where low-skill training outperforms high-skill is highlighted in blue.

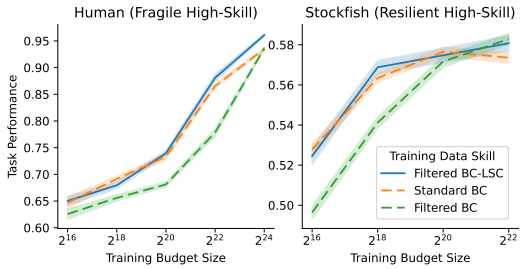


Figure 12: Effect of training BC models on human and engine demonstrations using skill-based curricula in chess by skill level.

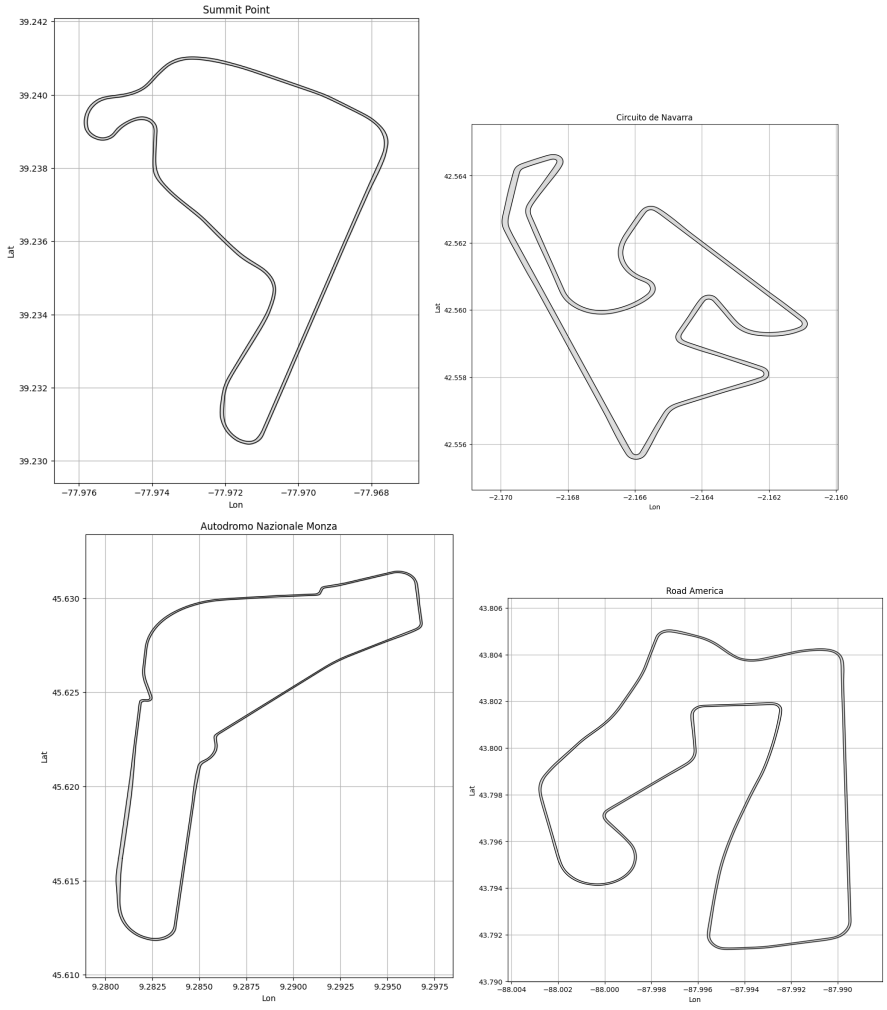


Figure 13: Diagrams of each of the four tracks we evaluate BC on.

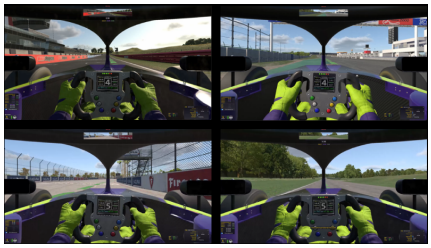


Figure 14: Screenshots of the racing simulator iRacing.

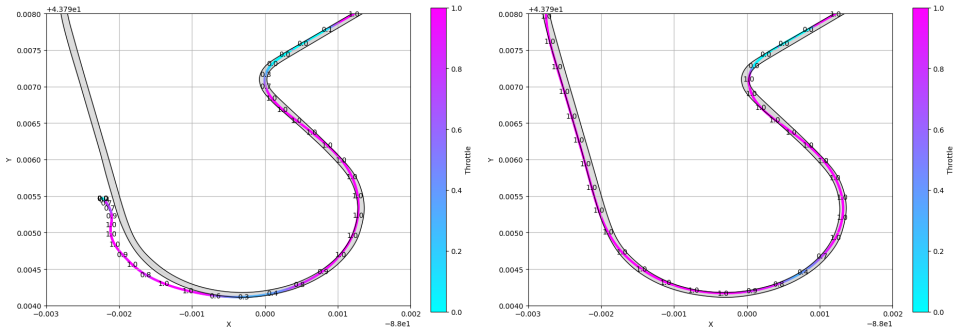


Figure 15: Failed versus successful trajectory on Road America.

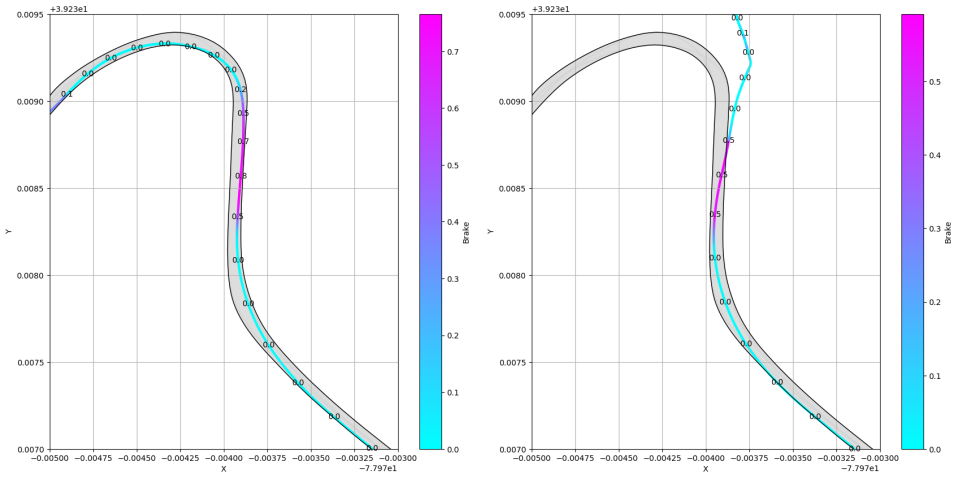


Figure 16: Failed versus successful trajectory on Summit Point.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In our abstract/introduction, we discuss three main contributions: (1) proposing fragility as a measure, (2) showing how this is linked to low-skill outperformance, and (3) connecting our findings to curriculum learning. We discuss these in Section 3.2, Section 4, and Section 5, respectively.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a paragaph on the limitation section in the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper provides an exact condition for low-skill BC to outperform high-skill BC in terms of imitation-induced degradation. For the subsequent diagnostic relationship between fragility and BC performance, we clearly state the assumptions under which the relationship is derived, including the additive equal-error special case in the main text and broader unequal-error, state-dependent, and non-additive cases in Appendix A. We also discuss cases where the current fragility measure may be insufficient and how it could be refined.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide references all the code and data needed to replicate our results in Tree, MuJoCo, and Chess. In racing, we are unable to provide references to the code and data due to an agreement with our provider, but provide details on how to reproduce the results in Section E

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide references all the code and data needed to replicate our results in Tree, MuJoCo, and Chess. In racing, we are unable to provide references to the code and data due to an agreement with our provider, but provide details on how to reproduce the results in Section E, and the general algorithmic approach is identical to those in the other sections.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The exact details for training is specified in the codebase. We also provide details on model hyperparameters in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Each of our plots contains standard error bars over experimental seeds and we report statistical significance in each of the sections when discussing differences across treatments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In the Appendex in each of the sections, we describe our compute cluster resource requirments and time of execution, including an estimation for preliminary work which didn't make it into the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and conform with the NeurIPS Code of Ethics

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We provide an impact statement in Appendix E.2

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: There are no high-risk applications in our paper.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: Our code provided is entirely our own. We have cited the models and data in MuJoCo and Chess that we use, which have been provided freely. We have complied with the terms provided by our provider for the Racing data.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We have provided our licensed code.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: We did not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: We did not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Our core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.