Human-Robot Interaction Playground

Matteo Calabria[a,b], **Giulia Benintendi**[a,b], **and Stefano Abondio**[a,b]

[a]Sorbonne University; [b]University Of Brescia

This report was compiled on November 28, 2025

Imitation learning offers a direct alternative to designing complex reward functions, but Behavioral Cloning (BC) often fails due to distributional shift: when a learned policy reaches states not covered by the expert demonstrations, errors accumulate and performance quickly degrades. This report investigates interactive imitation-learning methods that address this limitation. We present a systematic study of DAgger and a replay-buffer DAgger variant, evaluating their behavior across three MuJoCo environments of increasing complexity: Walker2d-v5, HalfCheetah-v5, and Humanoid-v5. We compare their performance under different numbers of demonstrations and we examine how expert quality influences imitation by using multiple experts and experts with different proficiency levels. We also compare SAC- and PPO-based experts to highlight how algorithmic biases affect imitation. Overall, the study demonstrates that interactive correction consistently outperforms passive supervised learning, leading to better generalization and data efficiency.

Imitation Learning | Behavioral Cloning | DAgger | MuJoCo

While Reinforcement Learning (RL) provides a general framework for control, it relies on carefully engineered reward functions, which are often difficult and time-consuming to design. Imitation Learning (IL) sidesteps this issue by learning directly from expert demonstrations. However, the simplest form of IL, Behavioral Cloning, treats the problem as supervised learning and suffers from distributional shift: because the learned policy π is imperfect, it visits a state distribution d_π that gradually drifts away from the expert distribution d_{expert}. In these unfamiliar states, the policy is poorly defined, errors compound over time and tasks that appear solved in the training data can fail catastrophically at test time.

This problem is particularly acute in robotics, where expert demonstrations are expensive and limited. In that setting, data efficiency is not a bonus but a hard constraint. Interactive IL methods, such as DAgger and its variants, aim to correct for distribution shift by allowing the learner to query the expert on its own on-policy states and, ideally, to focus on the "weak states" where it makes the largest mistakes.

In this work, we address the following questions:

1. How do BC, standard DAgger, and DAgger with a replay buffer trade off final performance and number of demonstrations?
2. How does the expert's RL algorithm (SAC vs. PPO) affect the downstream imitation learner?
3. Is the highest-performing RL agent also the best teacher for an imitation learner?

Our contributions are:

- A comparison between BC, DAgger and replay-buffer DAgger across different numbers of demonstrations.
- An analysis of the "best agent vs. best expert" phenomenon, including SAC vs. PPO and experts with different levels of expertise.
- An open-source implementation of our experimental framework to ensure full reproducibility of our findings.[*]
- Pre-trained expert policies publicly available for benchmarking.[†]

We now briefly review the theoretical background of the IL algorithms considered in this study; full pseudo-code implementations are provided in Appendix A.

1. Background and Theory

In imitation learning, the agent learns from expert demonstrations, provided by a pre-trained policy ($\pi^\star$) or a human demonstrator, and aims to learn a policy π_θ that mimics the expert's behavior when rolled out in the environment.

In our experiments, experts are RL agents trained with SAC or PPO, allowing us to query them for actions at arbitrary states. Now we briefly review the three imitation-learning algorithms used in this study: Behavioral Cloning (BC), Dataset Aggregation (DAgger), and a replay-buffer variant of DAgger that prioritizes learning from high-error regions of the state distribution[1].

1.1. Behavioral Cloning (BC). Behavioral Cloning is the most direct approach to imitation learning, framing the problem as a supervised learning task. Given a dataset of expert state-action pairs $\mathcal{D} = \{(s_i, a_i^\star)\}$, BC trains a policy π_θ to reproduce the expert's action at each state, typically using mean squared error as loss function. A full pseudocode description is given in Algorithm 1. BC assumes that if a policy can accurately mimic the expert's actions on states the expert has visited, it will successfully replicate the expert's behavior.

In sequential decision making this assumption fails: BC is trained under $d_{\pi^\star}$ but deployed under d_{π_θ}, and each action affects future states. Small errors early in a trajectory can push the agent into states poorly covered by $\mathcal{D}$, where the policy is less accurate. If BC has per-step error at most ε under $d_{\pi^\star}$, the performance gap relative to the expert can grow as $O(\varepsilon T^2)$ over a horizon T [2]. BC is therefore reliable mainly when the demonstrations cover the relevant parts of the state space or when the environment is robust to small

[*] https://github.com/maedmatt/hri-playground

[†] https://huggingface.co/maedmatt/models

Significance Statement

Imitation learning offers a practical alternative to engineered reward functions, but simple Behavioral Cloning suffers from distributional shift, leading to fast error accumulation. By comparing different number of demonstrations, multiple experts, and policies trained with SAC or PPO, we show how data aggregation strategies influence generalization and stability.

deviations. To address this limitation, interactive methods, such as DAgger, were introduced allowing the agent to request additional expert supervision on the states it actually visits.

1.2. Dataset Aggregation (DAgger). DAgger addresses distributional shift by iteratively collecting on-policy data [3]. In this algorithm, a mixed policy, controlled by a parameter β, is executed in the environment. For the states the mixed policy visits, the learner queries the expert for the correct action labels. This new on-policy data is then aggregated with all previous data to retrain the policy. By iteratively correcting the learner on its own state distribution, DAgger keeps training focused on the states that matter for the current policy, rather than only on those visited by the expert.

Here $\beta_i \in [0, 1]$ controls how often we trust the expert label versus the learner's own action: early iterations use large β_i to closely follow the expert, while later iterations gradually shift towards reinforcing the learner's decisions. The states s_t always come from rollouts of the current learner, so the aggregated dataset tracks the evolving state distribution d_{π_θ} instead of remaining fixed at the expert's distribution. Algorithm 2 provides the full DAgger implementation.

1.3. DAgger with Replay Buffer. Pure DAgger aggregates all data ever collected and the dataset grows with every iteration. While this captures the full history of corrections, it also means that many samples may become less relevant as the learner's behavior changes and training repeatedly on the entire history can be redundant and slow. To address this, we use a replay-buffer variant that focuses on critical states where the agent's action diverges most from the expert's action [1]. The replay buffer stores only the highest-error states from each iteration, ensuring training prioritizes the most challenging regions while maintaining computational efficiency. The complete procedure is formalized in Algorithm 3.

Having established the theoretical foundations, we now describe our experimental setup, including the environments, expert policies, and training protocols used to evaluate these algorithms.

2. Experimental Setup

2.1. Environments. We conduct experiments on three MuJoCo continuous control tasks from Gymnasium: Walker2d-v5, HalfCheetah-v5 and Humanoid-v5. Walker2d-v5 features a 17-dimensional observation space and 6-dimensional continuous action space for bipedal locomotion. HalfCheetah-v5 involves quadrupedal (planar) running with similar dimensionality but different dynamics. Humanoid-v5 presents the most complex scenario with high-dimensional state and action spaces requiring coordinated control across multiple joints [4]. These environments are of increasing complexity, from from bipedal locomotion to quadrupedal running to full humanoid control, and allow us to test how IL algorithms perform as task difficulty grows.

2.2. Expert Policy Training. The choice of expert policy is critical in imitation learning, as teacher quality and characteristics directly influence learner performance. We trained RL agents using Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) for 5 million and 20 million timesteps across all three environments. All training runs were tracked and monitored using Weights & Biases (wandb) to ensure reproducibility and facilitate performance analysis.

The two algorithms differ fundamentally in their learning approach [5]:

1. **SAC[6]**: off-policy algorithm, that learns from a replay buffer, allowing each transition to be reused many times, making it more sample-efficient. SAC also maximizes expected return and entropy, a measure of randomness in the policy, encouraging broader exploration and producing stochastic policies.

2. **PPO[7]**: on-policy algorithm, it must collect fresh data for each update, but this constraint provides more stable, reliable learning. PPO prevents from large policy updates, leading to more conservative, deterministic and predictable behavior.

These differences create distinct teaching profiles: SAC provides varied, exploratory demonstrations while PPO offers consistent, stable trajectories.

2.3. Experts Performance Analysis. Figure 1 and Table 1 shows the performances of each model, evaluated over 100 episodes, across all the environments. SAC outperformed PPO in all scenarios and the gap gets wider as task complexity increases.

In **Walker2d-v5**, the least complex of the three environment, SAC-20M achieves 5861.18 ± 73.99, while PPO-20M stabilizes at 4593.26 ± 885.92, with substantial higher variability.

In **HalfCheetah-v5**, the performance gap widens considerably. SAC reaches high performance early and continues improving. The 20M SAC agent obtains 14963.06 ± 198.41, about three times the score of PPO-20M. SAC also shows lower variance, indicating more stable learning. The larger gap compared to Walker2d indicates that HalfCheetah's more challenging dynamics amplify SAC's advantages.

The contrast is even sharper in **Humanoid-v5**. SAC-20M stabilizes near 7150.23 ± 1588.78, while PPO-20M stops at 664.02 ± 122.27, almost an order of magnitude difference. Humanoid's high-dimensional and unstable dynamics amplifies the strengths of SAC—broad exploration, while PPO improves only marginally from 5M to 20M, confirming early saturation.

Training-scale effects follow the same pattern. SAC improves reliably from 5M to 20M in all environments, while PPO benefits less from additional data and sometimes not at all, as in Humanoid.

We also benchmarked against expert policies from the **Minari dataset** (Table 1). Minari experts achieved 11448.22 ± 96.24 in HalfCheetah-v5 (lower than to our SAC-5M), 6956.61 ± 15.95 in Walker2d-v5 (comparable to our SAC-20M), and 8127.00 ± 46.46 in Humanoid-v5 (slightly exceeding our best trained agent). These values serve as upper-bound performance targets, showing that higher-level experts exist for more challenging tasks.

2.4. Best Teacher vs Best Expert. Given SAC's consistent better performances across all environments, a natural question arises: *"does the best-performing agent necessarily make the best teacher?"* SAC's exploration and stochasticity might help

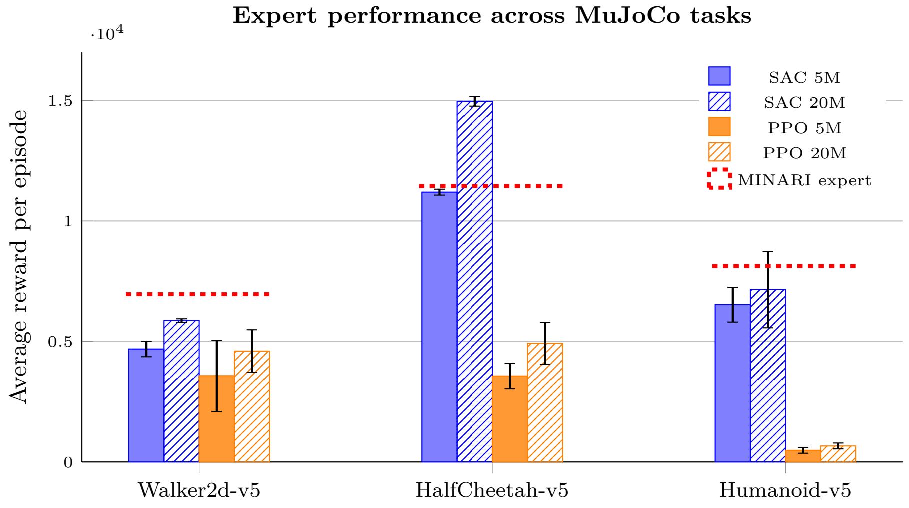

Fig. 1. Average reward (mean ± std over 100 episodes) for SAC and PPO experts in Walker2d-v5, HalfCheetah-v5 and Humanoid-v5.

Table 1. Expert Policy Performance Across Environments

	Walker2D	HalfCheetah	Humanoid
SAC 5M	4679.77 ±321.98	11196.44 ±122.84	6521.45 ±720.85
SAC 20M	5861.18 ±73.99	14963.06 ±198.41	7150.23 ±1588.78
PPO 5M	3568.65 ±1470.93	3558.59 ±524.94	484.13 ±122.03
PPO 20M	4593.26 ±885.92	4916.30 ±871.87	664.02 ±122.27
Minari Expert	6956.61 ±15.95	11448.22 ±96.24	8127.00 ±46.46

Values report mean episode returns over 100 evaluation episodes; standard deviations are shown below each mean.

learners see diverse recovery behaviors, while PPO's consistency might make patterns easier to imitate. To test this, we run a targeted comparison in Walker2d-v5 using both SAC and PPO as experts. We return to this question in section 3.1, where we empirically compare SAC and PPO as teachers in Walker2d-v5. Each expert was trained for 20,000,000 timesteps using 8 parallel environments (SubprocVecEnv) with a fixed random seed (42)[‡] for reproducibility. Model checkpoints were saved every 250,000 steps to allow intermediate policy analysis. We now describe how expert demonstrations were collected and how imitation learning policies were trained.

2.5. Demonstrations Collection.

We collected demonstration datasets by rolling out expert policies. Since SAC learns a stochastic policy, we used the mean action (`deterministic=True` in Stable Baselines3 [9]) to generate consistent, high-quality trajectories. We gathered 10, 30, and 100 episodes per environment, each running until termination or 1,000 steps.

With demonstration datasets prepared, we go into the details of the training protocol for each imitation learning algorithm.

2.6. Imitation Learning Training Protocol.

- **BC**: We trained BC policies using a feedforward neural network with two hidden layers of 256 units each, ReLU activations and tanh output activation. Before training, we computed observation normalization mean and standard deviation from the demonstration dataset and applied z-score normalization to stabilize training. Training proceeded for 100 epochs with adaptive batch size based on dataset dimensions, learning rate 3×10^{-4} and Adam optimizer. The trained BC policies are both baselines and initialization checkpoints for DAgger variants.

- **DAgger**: Initialized from a pretrained BC policy. We perform N iterations (where N varies by experiment), with each iteration consisting of: (i) rolling out the current policy π_θ to collect x new trajectories, (ii) querying the expert policy π^* for optimal actions at all visited states and (iii) aggregating these newly-labeled state-action pairs with all previous data to retrain the policy, progressively shifting from exploration under the expert to exploitation under the learned policy. The full dataset grows with each iteration, accumulating all state-action pairs ever collected. Policy performance is evaluated every iteration through deterministic rollouts to track learning progress.

- **DAgger with Replay Buffer**: Follows the same iterative structure as standard DAgger but adds a selective memory mechanism. For each trajectory collected, we identify the top-k = 100 states with highest error: these are the critical states where the policy most needs correction. The training dataset for each iteration combines: (i) the newly collected on-policy data from the current iteration, and (ii) all critical states stored in the replay buffer R from previous iterations. This prioritization ensures the policy repeatedly sees states where it historically struggled, without storing the entire history of data. The replay buffer has a maximum capacity of 200,000 transitions; when this limit is exceeded, we remove the oldest entries in a FIFO manner.

[‡] The answer to life, the universe, and everything [8].

3. Results

We evaluate all imitation learning policies, following the same protocol used for expert evaluation (Figure 1): 100 episode rollouts per policy.

Unless stated otherwise, all experiments use SAC-20M as the expert policy, as it achieved the highest performance across all environments (Table 1).

Since reward scales differ substantially, we report performance as *expert recovery*, in Equation 1, that describe the percentage of expert performance achieved by the learned policy:

$$\text{Expert Recovery (\%)} = \frac{\bar{R}_\pi}{\bar{R}_{\pi^\star}} \times 100 \qquad [1]$$

where $\bar{R}_\pi$ is the mean episode return of the learned policy and $\bar{R}_{\pi^\star}$ is the mean return of the expert. Uncertainties are computed via standard error propagation from the evaluation rollouts.

We begin by comparing SAC and PPO as expert teachers to determine whether the best-performing agent is also the best teacher. Based on these findings, we then examine how demonstration quantity affects Behavioral Cloning, followed by an analysis of DAgger and DAgger with replay buffer iterations.

3.1. SAC vs PPO as Expert Teachers. In Section 2.4, we posed the question: does the best-performing agent necessarily make the best teacher? SAC outperforms PPO across all environments (Table 1), but its stochastic, exploratory behavior might produce less consistent demonstrations than PPO's conservative, deterministic policy. To test this, we trained BC policies on Walker2D using both SAC-20M and PPO-20M as experts.

Table 2. BC performance on Walker2D-v5: SAC vs PPO experts.

	10 demos	30 demos	100 demos
Raw reward			
SAC-20M	494 ± 274	883 ± 474	1415 ± 964
PPO-20M	2651 ± 1884	3125 ± 2266	4778 ± 607
Expert recovery (%)			
SAC-20M	8.4 ± 0.5	15.1 ± 0.8	24.2 ± 1.6
PPO-20M	57.7 ± 4.2	68.0 ± 5.1	104.0 ± 2.4

Values report mean ± standard deviation over 100 evaluation episodes. Expert recovery computed relative to each expert's own performance.

Results reveal that PPO is a substantially better teacher than SAC on Walker2d-v5 (Table 2). BC trained on PPO demonstrations achieves higher raw rewards across all dataset sizes. More strikingly, BC trained on PPO exceeds 100% expert recovery at 100 demonstrations, suggesting that the learner can match or even improve upon PPO's performance.

This finding confirms that the best-performing agent is not necessarily the best teacher. PPO's conservative, consistent behavior produces demonstrations that are easier to imitate, while SAC's exploratory stochasticity creates more varied trajectories that confuse the learner.

However, this advantage comes with a critical limitation: PPO fails to learn competent policies in more complex environments. In HalfCheetah-v5 and Humanoid-v5, PPO achieves only a fraction of SAC's performance (Table 1), making it unsuitable as an expert regardless of its teaching efficiency.

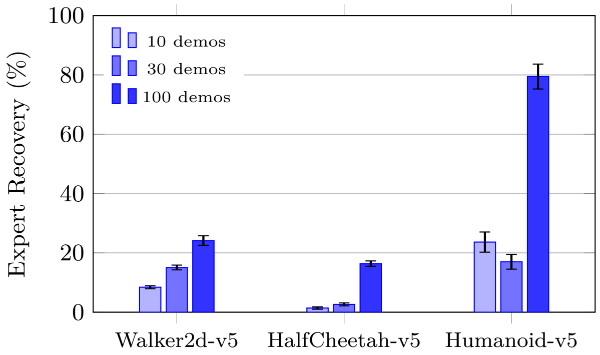

Fig. 2. Expert recovery of Behavioral Cloning policies across environments and demonstration quantities. Error bars show propagated standard errors over 100 evaluation episodes. Higher is better; 100% indicates expert-level performance.

3.2. Behavioral Cloning: Effect of Demonstration Quantity. We trained BC policies using 10, 30, and 100 expert demonstrations per environment to assess how dataset size affects imitation quality.

Results reveal two distinct patterns (Figure 2). In Walker2d-v5 and HalfCheetah-v5, BC performance increases monotonically with dataset size but remains well below expert level: even with 100 demonstrations, BC recovers only 24.2% ± 1.6% and 16.4% ± 0.9% respectively. This confirms the distributional shift problem discussed in section 1.1; small errors push the policy into states not covered by the demonstrations, where mistakes compound.

Humanoid presents an apparent exception, reaching 79.5% ± 4.5% expert recovery with 100 demonstrations. However, this result is misleading: the SAC-20M expert for Humanoid achieves only 87% of the Minari baseline (Table 1), indicating a suboptimal teacher. Furthermore, rollouts from this unreliable expert include outright failures (episodes where the humanoid falls within 500 steps) and these low-quality trajectories contaminate the demonstration dataset. This explains the non-monotonic trend (30 demos underperforming 100 demos): larger datasets have a higher probability of including such failure modes, degrading rather than improving the learned policy.

Investigating the Humanoid Anomaly. To test the contamination hypothesis, we collected filtered demonstration datasets containing only successful episodes (length > 500 steps) and retrained BC policies. Table 3 shows the results.

Filtering restored monotonic scaling: with quality-controlled demonstrations, more data consistently improves performance ($5.7\% \pm 0.2\% \to 21.1\% \pm 3.0\% \to 67.1\% \pm 1.5\%$). This confirms that the original non-monotonicity was caused by failed episodes contaminating larger datasets. The suboptimal SAC-20M expert produces unreliable rollouts—approximately 10% of collected episodes fail before 500 steps—and these failure trajectories teach BC incorrect behaviors.

Even with filtered demonstrations, BC recovers only 67% of expert performance at 100 demonstrations, consistent with the distributional shift limitation observed in Walker2D and HalfCheetah.

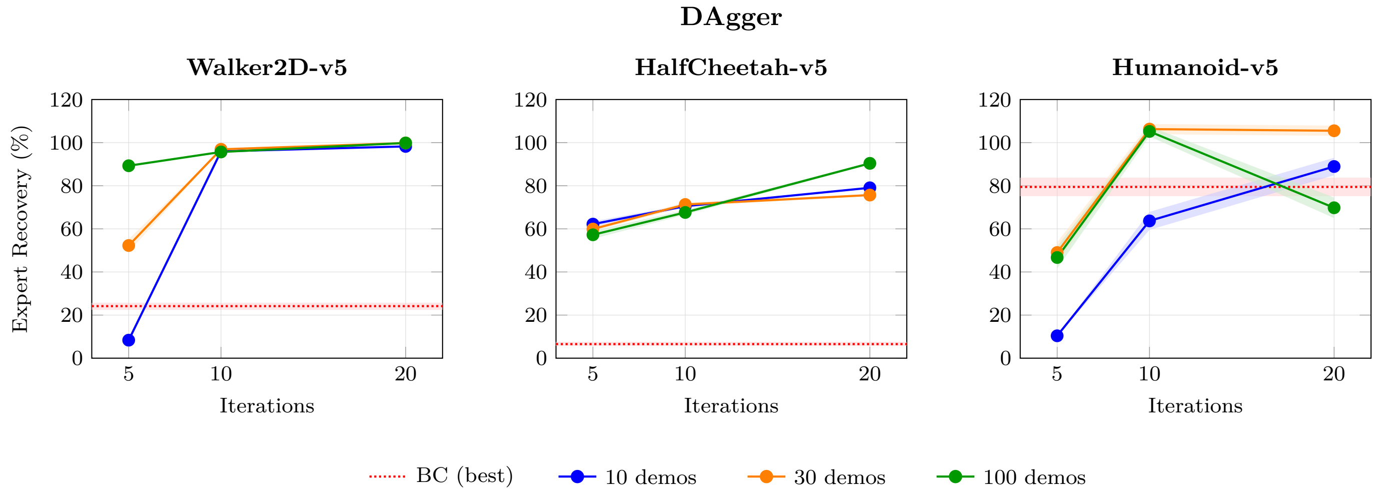

Fig. 3. DAgger expert recovery across iterations for different demonstration quantities. The red dotted line indicates best BC performance (100 demonstrations). Shaded regions indicate mean ± standard error over 100 evaluation episodes.

Table 3. BC performance on Humanoid-v5 with filtered demonstrations (episodes > 500 steps only).

	10 demos	30 demos	100 demos
Demonstration statistics			
Episodes collected	11	32	114
Episodes discarded	1	2	14
Mean demo return	7548.1	7524.4	7542.3
Std demo return	31.4	139.7	137.3
BC evaluation			
Expert recovery (%)	5.7 ± 0.2	21.1 ± 3.0	67.1 ± 1.5
Mean return	408.9	1505.5	4795.9
Std return	76.2	2088.6	3284.6

Episodes discarded indicates failed demonstrations (length < 500 steps) removed during filtering. BC evaluation reports mean ± standard error over 100 rollouts.

3.3. DAgger: Interactive Correction. We trained DAgger policies initialized from the BC checkpoints described is section 3.2 (trained on 10, 30, and 100 demonstrations respectively) and ran 5, 10, and 20 DAgger iterations to evaluate the effect of interactive corrections.

Unlike BC, DAgger allows the agent to query the expert on states it actually visits during rollout, directly addressing distributional shift. We focus on three aspects of its behavior: data efficiency relative to BC, performance evolution across iterations and the effect of demonstration quantity. Results in Figure 3 show that while all environments benefit from DAgger, the learning behaviour depends on environment complexity and on the quantity and quality of demonstrations.

Data Efficiency over Behavioral Cloning. DAgger consistently improves BC when demonstrations are limited. In Walker2d-v5, BC with 10 demonstrations reaches only 8.42% ± 0.50% expert recovery, while DAgger at 20 iterations achieves 98.25% ± 0.13%. A similar pattern appears in HalfCheetah-v5, where BC with 10 demonstrations leads to –3.25% ± 0.24% (negative values occur when the agent moves left instead of right). DAgger recovers 79.05% ± 0.18% at 20 iterations.

The effect is even more evident in Humanoid-v5. With 10 demonstrations, BC obtains 23.63% ± 3.40%, while DAgger at 20 iterations reaches 88.92% ± 4.06%. Starting from BC with 30 demonstrations, DAgger reaches 105.51% ± 2.35%, exceeding the SAC-20M expert. However, these results reflect both DAgger's correction capability and the contamination issues discussed in Paragraph 3.2 with larger datasets including failed demonstrations that destabilize learning at higher iteration counts.

Effect of Number of Iterations. While all three environments show fast early improvement, their convergence patterns reveal how task complexity impacts the efficiency of iterative correction. Walker2d-v5 converges quickly, HalfCheetah-v5 requires a larger number of iterations and Humanoid-v5 exposes non-monotonic behavior. In Walker2d-v5, improvements are fast. Starting from BC with 10 demonstrations, recovery increases from 8.36% ± 0.43% at 5 iterations to 95.95% ± 0.21% at 10 iterations, with only limited gains to 98.25% ± 0.13% by 20 iterations. Only a small number of correction cycles is needed to reach near-expert behavior.

HalfCheetah-v5 shows a more gradual progression. Starting from BC with 100 demonstrations, performance increases across all iterations counts: 57.25% ± 1.85% (5 iterations) → 67.61% ± 1.10% (10 iterations) → 90.39% ± 0.45% (20 iterations), with significant gains even in the final phase. In contrast to Walker2d-v5, performance continues improving between iterations 10 and 20, indicating that HalfCheetah-v5 requires more correction phases to converge.

Humanoid-v5 shows the most complex dynamics. Starting from BC with 30 demonstrations, DAgger reaches 106.28% ± 2.36% at 10 iterations, already exceeding the expert, and maintains similar performance at 20 iterations (105.51% ± 2.35%). However, starting from BC with 100 demonstrations, performance peaks at 10 iterations (105.20% ± 2.34%) but drops to 69.78% ± 4.76% at 20 iterations, indicating instability due to inconsistent demonstrations. As discussed in Section 3.2, filtering short or inconsistent trajectories mitigates this effect and restores monotonic improvement.

DAgger's full-history aggregation improves over BC, but this strategy raises an efficiency question: could we achieve similar or better results by focusing computational resources

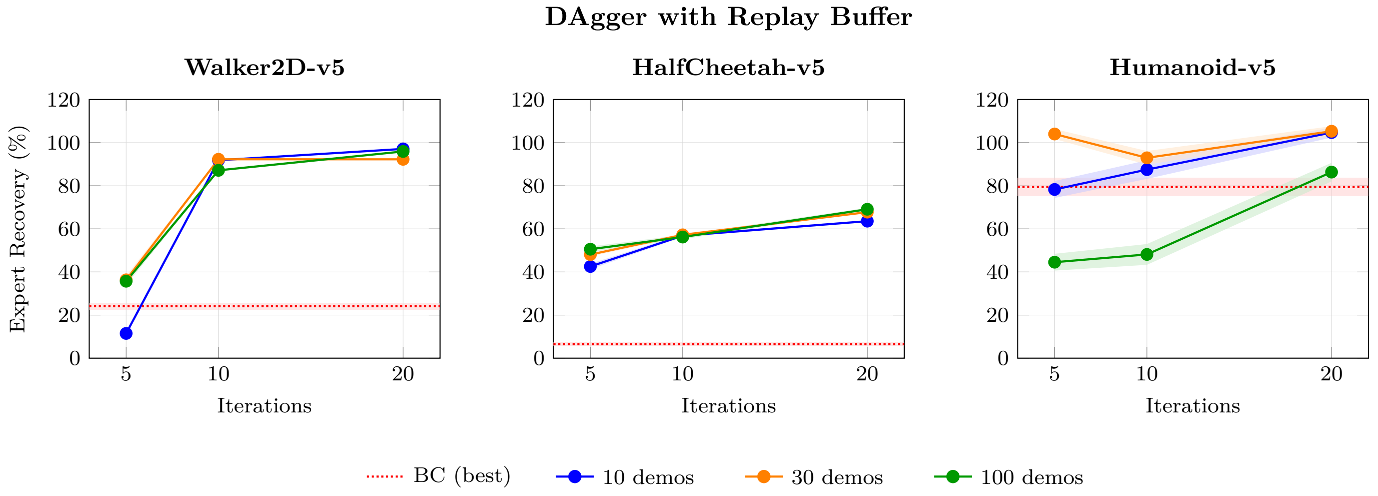

Fig. 4. DAgger with Replay Buffer expert recovery across iterations for different demonstration quantities. The red dotted line indicates best BC performance (100 demonstrations). Shaded regions indicate mean ± standard error over 100 evaluation episodes.

on the most informative states? We explore this through the replay-buffer variant.

3.4. DAgger with Replay Buffer: Selective Memory. The replay-buffer variant tests a specific hypothesis: prioritizing the highest-error states should improve data efficiency by avoiding redundant updates on already-mastered behaviors. Our results show that this hypothesis holds only under particular conditions, depending on environment complexity, state-space dimensionality and the quality of the demonstrations used.

Environment-Dependent Effectiveness. Replay-buffer effectiveness depends strongly on environment complexity. In Walker2d-v5, classic DAgger shows comparable performances with the replay-buffer variant. At 20 iterations, both DAgger and DAgger-RB reach near-expert recovery levels (>90% recovery), with both methods converging to similar performance levels. The selective mechanism discards part of the trajectory history, but in Walker2d-v5 this does not alter the overall learning behaviour.

In HalfCheetah-v5, retaining the full history becomes essential. Here, DAgger achieves 79.05% ± 0.18% → 75.68% ± 0.35% → 90.39% ± 0.45% at 20 iterations. The replay-buffer variant instead reaches 63.57% ± 0.52% → 67.76% ± 0.30% → 69.09% ± 0.68%, indicating that the full trajectory distribution is necessary for learning HalfCheetah's running behaviour.

In Humanoid-v5, as shown in Table 4, the trend reverses. Starting from BC with 10 demonstrations, at 20 iterations, DAgger-RB reaches 104.68% ± 2.33%, compared to DAgger's 88.92% ± 4.06%. The replay-buffer variant is still advantageous as demonstrations increase, achieving stronger or comparable results with fewer iterations.

Table 4. Humanoid-v5: DAgger and DAgger-RB expert recovery (ER) across iterations and demonstration quantities.

	10 demos	30 demos	100 demos
DAgger (5 iter)			
ER (%)	10.39 ± 0.24	49.02 ± 4.81	89.28 ± 0.18
DAgger-RB (5 iter)			
ER (%)	78.27 ± 3.92	104.00 ± 2.31	35.70 ± 1.81*
DAgger (10 iter)			
ER (%)	88.92 ± 4.06	106.28 ± 2.36	105.20 ± 2.34
DAgger-RB (10 iter)			
ER (%)	104.00 ± 2.31	112.28 ± 3.34	48.13 ± 4.82*
DAgger (20 iter)			
ER (%)	88.92 ± 4.06	105.29 ± 2.34	69.78 ± 4.76*
DAgger-RB (20 iter)			
ER (%)	104.68 ± 2.33	105.51 ± 2.35	86.33 ± 4.32*

Values report mean ± standard error over 100 evaluation episodes. Values annotated with * correspond to performance degradation or unexpectedly low values due to demonstration contamination.

Full-history DAgger accumulates many redundant samples from already-mastered regions, reducing the effect of informative corrections. The replay-buffer formulation counteracts this by consistently emphasizing high-error states, focusing learning on the parts of the state distribution most relevant for stabilizing full-body control. This behaviour reflects that, with high-dimensional control tasks, early rollouts cover only a limited region of the state space and focusing on high-error states behaves more efficiently than full-history aggregation.

Computational Motivation. Beyond performance, DAgger-RB offers computational advantages. Pure DAgger's dataset grows as $\mathcal{D}_t = \mathcal{D}_0 + t \times n_{\mathrm{traj}} \times \text{avg\_steps}$, exceeding 200,000 samples after 20 iterations. The replay-buffer variant bounds storage at approximately the current iteration's data plus accumulated critical states (roughly 1,000 per iteration), leading to 2–5× faster training at high iteration counts. This makes DAgger-RB particularly appealing in environments where both dimensionality and rollout length make full-history aggregation computationally expensive.

4. Conclusion

We presented a systematic comparison of imitation learning algorithms—Behavioral Cloning, DAgger, and DAgger with Replay Buffer—across three MuJoCo environments of increasing complexity. We now revisit the research questions posed in the introduction.

Q1: How do BC, standard DAgger, and DAgger with a replay buffer trade off final performance and number of demonstrations? BC suffers severely from distributional shift, recovering at most 24.2% ± 1.6% of expert performance on Walker2D and 16.4% ± 0.9% on HalfCheetah even with 100 demonstrations.

Interactive methods dramatically improve this: DAgger achieves near-expert performance (>95% recovery) within 10–20 iterations across environments. The replay buffer variant shows environment-dependent convergence patterns, with comparable performance to standard DAgger in Walker2d-v5, repeatedly lower performance in HalfCheetah-v5 and stronger and earlier convergence in Humanoid-v5. These results indicate that selective memory has a more impactful effect on high-dimensional tasks where redundant samples reduce correction signals.

Q2: How does the expert's RL algorithm (SAC vs. PPO) affect the downstream imitation learner? SAC and PPO produce qualitatively different teaching profiles. PPO's conservative, deterministic behavior generates demonstrations that are easier to imitate—BC trained on PPO exceeded 100% expert recovery on Walker2d-v5. However, PPO fails to learn competent policies in complex environments, limiting its practical utility as an expert despite its teaching efficiency.

Q3: Is the highest-performing RL agent also the best teacher for an imitation learner? No. Our results demonstrate a clear decoupling between agent performance and teaching effectiveness. PPO substantially outperforms SAC as a teacher on Walker2d-v5 despite achieving lower rewards. This finding has practical implications: when both algorithms succeed, practitioners should consider teaching efficiency, not just raw performance, when selecting an expert.

Beyond these questions, we identified that expert reliability critically affects BC performance. Suboptimal experts that produce failed demonstrations can cause non-monotonic scaling, where more data degrades performance. Quality filtering restored expected behavior, highlighting the importance of demonstration curation.

4.1. Limitations. Our study has several limitations. First, we evaluated only simulated experts with unlimited query access; real-world imitation learning often involves human demonstrators with limited availability. Second, our environments, while spanning a range of complexity, are all locomotion tasks; generalization to manipulation or other domains remains untested. Third, we used a single network architecture and hyperparameter configuration for all BC and DAgger experiments. Fourth, we limited our expert training to SAC and PPO without extensive hyperparameter tuning or exploring alternative algorithms. Other methods such as TQC achieve substantially higher performance on Humanoid (10370 ± 1542 in the Minari dataset[§]), and using stronger experts would likely improve imitation learning outcomes, particularly for complex tasks where our SAC-20M expert remained suboptimal.

4.2. Future Work. Several directions deserve further investigation. Extending this analysis to real robotic systems with human demonstrators would test whether our findings transfer beyond simulation. Exploring hybrid approaches—such as using PPO for initial BC training followed by SAC-guided DAgger refinement—could combine the benefits of both expert types. Finally, investigating adaptive demonstration filtering during online learning could address the contamination problem without requiring manual curation.

References

[1] Aditya Prakash, Aseem Behl, Eshed Ohn-Bar, Kashyap Chitta, and Andreas Geiger. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[2] Dylan J. Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning, 2024. URL https://arxiv.org/abs/2407.15007.

[3] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] James Mock and Suresh Muknahallipatna. A comparison of ppo, td3 and sac reinforcement algorithms for quadruped walking gait generation. *Journal of Intelligent Learning Systems and Applications*, 15:36–56, 01 2023. .

[6] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019. URL https://arxiv.org/abs/1812.05905.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[8] Douglas Adams. *The Hitchhiker's Guide to the Galaxy*. Pan Books, London, 1979.

[9] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

A. Algorithms Pseudocode

This appendix provides complete pseudocode for the three imitation learning algorithms evaluated in this work. These implementations correspond to the theoretical descriptions in section 1.

[§] https://huggingface.co/farama-minari/Humanoid-v5-TQC-expert

Algorithm 1 Behavioral Cloning implementation

```
1. Train an expert policy $\pi^\star$ using RL (PPO or SAC).
2. Rollout $\pi^\star$ and record demonstration trajectories.
3. Build a dataset of state–action pairs $\mathcal{D} = \{(s_i, a_i^\star)\}$ from these trajectories.
4. Initialize a policy network $\pi_\theta$.
5. For each epoch $e = 1, \ldots, N_{\text{epochs}}$:
   (a) Sample minibatch $B \subset \mathcal{D}$.
   (b) Compute loss
       $$\mathcal{L}_B(\theta) = \frac{1}{|B|} \sum_{(s,a^\star) \in B} \|\pi_\theta(s) - a^\star\|_2^2.$$
   (c) Update $\theta$ with a gradient step on $\mathcal{L}_B(\theta)$ (Adam).
6. Save the trained BC policy $\pi_\theta$.
```

Algorithm 2 DAgger implementation

```
1. Load expert policy $\pi^\star$.
2. Create the environment and initialize policy network $\pi_\theta$ from a BC checkpoint
3. Initialize aggregated dataset $\mathcal{D} \leftarrow \emptyset$.
4. For DAgger iterations $i = 1, \ldots, N$:
   (a) Set $\beta_i \leftarrow \beta_{\text{decay}}^{i-1}$.
   (b) Set $\mathcal{D}_{\text{new}} \leftarrow \emptyset$.
   (c) For each of $M$ trajectories:
       i. Roll out current learner policy $\pi_\theta$ to obtain trajectory $\{(s_t, a_t)\}$ where $a_t = \pi_\theta(s_t)$.
       ii. For each state $s_t$ in the trajectory:
           A. Query expert action $a_t^\star = \pi^\star(s_t)$.
           B. With probability $\beta_i$, set label $\ell_t \leftarrow a_t^\star$; otherwise $\ell_t \leftarrow a_t$.
           C. Add $(s_t, \ell_t)$ to $\mathcal{D}_{\text{new}}$.
   (d) Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{\text{new}}$.
   (e) Train $\pi_\theta$ on $\mathcal{D}$ for $E$ epochs with MSE loss and Adam.
   (f) Evaluate $\pi_\theta$ on several rollouts and save the policy.
```

Algorithm 3 DAgger with Replay Buffer

```
1. Load expert policy $\pi^\star$ and initialize policy $\pi_\theta$ from BC.
2. Initialize replay buffer $\mathcal{R}$ with capacity $B_{\max}$.
3. For DAgger iterations $i = 1, \ldots, N$:
   (a) Set $\beta_i \leftarrow \beta_{\text{decay}}^{i-1}$.
   (b) Set $\mathcal{D}_{\text{new}} \leftarrow \emptyset$ and $\mathcal{D}_{\text{crit}} \leftarrow \emptyset$.
   (c) For each of $M$ trajectories:
       i. Roll out $\pi_\theta$ to obtain trajectory $\{(s_t, a_t)\}$ where $a_t = \pi_\theta(s_t)$.
       ii. For each state $s_t$ in the trajectory:
           A. Query expert action $a_t^\star = \pi^\star(s_t)$ and compute error $e_t = \|a_t - a_t^\star\|^2$.
           B. With probability $\beta_i$, set $\ell_t \leftarrow a_t^\star$; otherwise $\ell_t \leftarrow a_t$.
           C. Add $(s_t, \ell_t)$ to $\mathcal{D}_{\text{new}}$.
       iii. Select the top-$k$ states with highest $e_t$ in this trajectory and add $(s_t, a_t^\star)$ for those states to $\mathcal{D}_{\text{crit}}$.
   (d) Form the training dataset
       $$\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{new}} \cup \mathcal{R}$$
   (e) Train $\pi_\theta$ for $E$ epochs on $\mathcal{D}_{\text{train}}$ using MSE loss with uniformly sampled minibatches.
   (f) Append all samples in $\mathcal{D}_{\text{crit}}$ to the replay buffer $\mathcal{R}$.
   (g) If $|\mathcal{R}| > B_{\max}$, keep only the most recent $B_{\max}$ entries.
   (h) Evaluate $\pi_\theta$ and save the policy.
```