Multi-Task Offline Reinforcement Learning

Experimental Evaluation of the Generalizability of the Soft Actor-Critic + Behavioral Cloning Algorithm

Axel Otto Geist[1]

Supervisor(s): Matthijs T.J. Spaan[1]**, Max Weltevrede**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Axel Otto Geist
Final project course: CSE3000 Research Project
Thesis committee: Matthijs T.J. Spaan, Max Weltevrede, Elena Congeduti

Abstract

This paper examines the generalization capabilities of the Soft Actor-Critic (SAC) algorithm when combined with Behavioral Cloning (BC) in a MiniGrid Four-Room Environment. Reinforcement learning (RL), particularly offline, is important for tasks where interactions with the environments are risky or costly, and this research focuses on multi-task environments where generalizability to new tasks is crucial. Our findings indicate that SAC+BC can achieve generalization performance close to BC. Notably, while BC shows robustness across various dataset characteristics (quality, diversity, size), SAC alone struggles without integrating BC, highlighting the enhancement in generalization brought by this hybrid approach. Furthermore, an increased data size only enhances generalizability when introducing greater diversity. However, these results are constrained by hardware limitations, suggesting that further hyperparameter optimization and using more seeds could validate and possibly enhance our findings, demonstrating that SAC+BC is even more effective than shown. The implementation details and the source code for this study are available on GitHub at https://github.com/AxelGeist/multi-task-offline-reinforcement-learning.

1 Introduction

Reinforcement learning (RL) has experienced strong growth in certain domains during the last few years [1] due to its promise of solving complex decision problems. Unlike standard RL, often called online RL, where agents continuously interact with and learn from their environment, offline RL allows agents to learn solely from a pre-existing fixed dataset without further interaction with the environment. This dataset comprises a set of experiences previously gathered from the environment, enabling learning from past interactions. Offline RL is particularly interesting in a multi-task environment where agents try to learn from multiple tasks and then try to make a good decision on a new, unseen task. This approach is relevant for domains where interactions with the environment are either risky or too costly, such as autonomous driving and medical diagnostics. [2].

Studies by Levine et al. [3] and Kumar et al. [4] have laid the groundwork for understanding offline RL. Despite progress, integrating offline RL methods into multi-task scenarios remains challenging, as they often underperform compared to approaches like Behavioral Cloning (BC) [5]. BC is a strong baseline that can be used to show how good offline RL can generalize to new, unseen tasks in a multi-task setting [6].

Central to this research is the paper [6], which examines the generalizability of offline RL algorithms across varied tasks. This research aims to reproduce and extend the conclusions of [6]. Building on the findings of [6], the Soft Actor-Critic (SAC) algorithm in combination with BC is chosen to be investigated in a simplified MiniGrid environment called four-room [7, 8], with the goal to determine whether the generalization problems persist under these conditions.

SAC is an advanced RL algorithm that optimizes a stochastic policy in an off-policy manner, incorporating entropy to encourage exploration [9]. The choice to investigate SAC+BC stems from the hypothesis that the hybrid approach can mitigate some of the common challenges associated with SAC when used alone. Specifically, SAC's ability to explore and adapt to complex environments may complement BC's capability to quickly imitate effective behaviors from expert data, potentially leading to better generalization across multiple tasks.

The primary research questions on which this study is based are therefore:

1. Can SAC combined with BC effectively generalize to new tasks within a multi-task RL environment?
2. What characteristics of the offline dataset are critical for the success or failure of SAC+BC in such settings?

The structure of this paper is designed to answer these questions through a comprehensive analysis of both theory and practical experiments. This introduction is followed by a review of related papers in Section 2, which provides an in-depth analysis of previous research, highlights critical achievements, and identifies the gaps this study aims to address. The following background chapter in Section 3 will formalize the concepts used in this study. Section 4 then presents the experimental setup and describes the datasets generated for this study and the benchmarks used to assess the generalization capabilities of SAC+BC. In the subsequent sections, the results of the experiments are systematically presented. Finally, the paper concludes by summarizing the main findings, highlighting their contribution to the broader field of offline multi-task RL, and suggesting directions for future research.

Our key contributions are the following:

- Our findings indicate that SAC+BC has no significant generalization gap compared to BC when using optimal or suboptimal data.
- We demonstrate that SAC+BCs performed best when trained on higher quality (optimal) datasets.
- Our results show that increasing the amount of data improves the generalizability of SAC+BC if and only if it leads to a greater variety of data.

2 Related Work

Generalization Various studies have looked at the generalization capabilities of RL. Notable works include Hansen et al. [10], Mediratta et al. [6], Raileanu et al. [11], Cobbe et al. [12], and Farebrother et al. [13]. Studies such as Lyle et al. [14] and Kirk et al. [15] have focused on training online RL agents that generalize to new transition and reward

functions. These works are similar to ours as they explore how well specific RL algorithms can generalize to new tasks, although they utilize online RL algorithms while our focus is on offline RL. There is a more similar paper by Mazoure et al. [16], which trains an offline RL agent with contrastive learning. However, it is more focused on improving the generalizability of contrastive learning, while we focus on finding out how well SAC+BC can generalize and under what circumstances. The most closely related work is by Mediratta et al. [6], which examines the Generalization Gap. While their study investigates multiple offline RL agents against various baselines in two environments, we aim to reproduce their findings with a narrower focus. Our study will examine a single offline RL agent (SAC+BC), two baselines (BC and SAC), and one environment (four-room), none of which, except for the BC baseline, are part of the paper from Mediratta et al. [6].

Offline RL+BC Several researchers have already investigated the combination of RL with BC, mainly with online RL approaches [17]. TD3+BC is an example of integrating offline RL with BC and is known for its simplicity and state-of-the-art performance [17]. Like SAC+BC, TD3+BC is an off-policy algorithm, but it is not applicable to environments with discrete action spaces, such as the four-room environment [18] used in this study. Another related work is [19], which evaluates SAC+BC in continuous action space environments. Using offline data, this approach combines pre-training with BC and online fine-tuning. In contrast, to assess its generalizability, this research evaluates SAC+BC in a discrete action space environment exclusively in the offline and multi-task setting.

3 Background

This chapter introduces the most important concepts for this study. First, Offline RL is described, followed by the concepts of multi-task setting and generalizability, followed by an explanation of BC, SAC, and SAC+BC.

3.1 Offline RL

In RL, an agent learns decision-making by interacting with an environment within a Markov Decision Process (MDP) [20]. An MDP is formally defined as a tuple $M = (S, A, T, R, p_0, \gamma)$, where S is a set of states, A is a set of actions, $T : S \times A \to P(S)$ represents the transition probability function, which maps each state-action pair to a probability distribution over the states, $R : S \times A \to R$ is the reward function, specifying the immediate reward received after taking action in a state, $p_0 : P(S)$ represents the initial state distribution, specifying the likelihood of starting in each state, γ is the discount factor, with $0 \leq \gamma < 1$, indicating the present value of future rewards. [21] This aims to learn a policy (π) that maximizes the discounted cumulative reward.

RL is categorized into online and offline modes. In online RL, the agent learns by actively interacting with the environment in real time. Offline RL, or batch RL, trains the agent using a pre-collected, fixed dataset without further environment interaction. This method is suitable when real-time interaction is impractical or risky. However, it faces challenges like distribution shift, where the dataset may not fully capture the variety of real-world scenarios the agent might encounter. [3]

3.2 Multi-Task Setting and Generalizability

RL tasks can be single-task or multi-task. In a single-task setting, an agent learns to master one specific task by developing a policy (π) to maximize its expected reward (r) in a given environment (E). In contrast, a multi-task setting requires the agent to learn a versatile policy applicable to multiple related tasks across various environments (E_1, E_2, ..., E_n), each with unique objectives and reward structures. [22]

With the multi-task setting, it is possible to test the agent's ability to adapt to new, unseen tasks that share characteristics with the learned tasks. If the agent performs good on new, unseen tasks, then the agent generalizes well. Thus, generalizability in a multi-task setting refers to the ability of an agent to perform well in several different tasks for which it has not been specifically trained but which are similar to the training tasks.

3.3 BC

BC is a method in imitation learning where an agent learns to mimic expert behavior by training on a dataset of state-action pairs. Unlike RL algorithms that rely on reward signals, BC uses supervised learning to learn the policy from expert demonstrations directly [23]. The process can be formally defined as follows: Let $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$ be a dataset of N state-action pairs, where $s_i \in \mathcal{S}$ is a state and $a_i \in \mathcal{A}$ is the corresponding action taken by the expert in that state. The objective of BC is to learn a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ parameterized by θ that maps states to actions by minimizing the loss function. The loss function ℓ measures the difference between the predicted action $\pi_\theta(s_i)$ and the expert action a_i. Common choices for ℓ include the mean squared error for continuous actions or the negative log-likelihood for discrete actions. The experimental setup section describes in detail the loss function implemented in this study. The learned policy π_θ aims to generalize the expert's behavior to unseen states by effectively mapping states to actions based on the expert's demonstrations [23]. BC is simple to implement and does not require a reward function, making it a strong baseline for offline RL regarding generalization capabilities [6]. However, BC has limitations, such as poor performance with suboptimal or insufficiently diverse data [23].

3.4 SAC

SAC is a model-free, off-policy RL algorithm designed for environments with both continuous and discrete action spaces. It combines the benefits of value-based and policy-based methods while incorporating entropy regularization to encourage exploration. SAC maximizes the expected reward and entropy, balancing exploration and exploitation [9]. Thus, SAC introduces novel assumptions, including entropy regularization, the use of stochastic policies, and clipped double Q-learning to address overestimation bias by maintain-

ing two Q-networks and using the minimum Q-value for updates [9, 24].

3.5 SAC+BC

SAC+BC aims to improve the SAC algorithm by integrating a BC term. The BC term will be added to the policy loss [17], which should minimize the discrepancy between the agent's actions and those done in the offline dataset. The experimental setup section formalizes in detail the policy loss function implemented in this study. This should ensure that SAC behaves closer to the policy used to create the dataset and adds some stability.

4 Experimental Setup

The experimental setup describes the simulated environment, the details of the datasets created, the implementation of the algorithms, the tuning of the hyperparameters, and the measurement of the performance of these algorithms. Figure 6 shows an overview of the setup.

4.1 Simulation Environment

This study aims to assess whether SAC+BC can generalize in a multi-task setting and identify dataset properties that influence its generalization performance. The first step is to select the environment in which the algorithm will be trained and evaluated. A simplified version of the MiniGrid four-room environment is used, as it is straightforward enough to train offline RL algorithms within a reasonable timeframe on a standard laptop. Despite its simplicity, it remains complex enough to draw meaningful conclusions about generalization capabilities. This environment has also been validated by prior RL generalization research. [21]

We utilize three configurations of the environment as described in [21]: Train, Test_Reachable, and Test_Unreachable. Each configuration consists of 40 tasks, each defined by its starting location, goal location, and the topology of a four-room layout. The Train configuration is designed for agent training, while the Test configurations evaluate the agent's performance. In the Test_Reachable configuration, tasks only differ from those in the Train set by their starting locations. The goal locations and topology remain unchanged. Consequently, the agent can locate the goal from any new starting point by applying knowledge acquired during training. Conversely, the Test_Unreachable tasks involve variations in goal locations or topology, which prevent the agent from reaching the goal solely by following the learned policy. The rationale behind these two test configurations is to differentiate between reachable and unreachable generalization scenarios. We expect that performance on reachable tasks will be superior, as these conditions are more prevalent in the training data. [21].

4.2 Datasets

After defining the environment, the next step is to create multiple datasets to train the offline RL model. These datasets are created with various characteristics, such as quality and size, as existing research suggests that these attributes significantly influence the generalization capabilities of offline RL algorithms [6]. Data diversity can also positively affect the generalization ability of an RL algorithm [6]. Therefore, the impact of data diversity will be indirectly assessed through data quality, given that datasets derived from suboptimal and mixed policies are inherently more diverse than those from optimal policies.

Thus, the following datasets are created:

1. Optimal dataset with experience from 40 episodes
2. Suboptimal dataset with experience from 80 episodes
3. Mixed optimal-suboptimal dataset with experience from 80 episodes

The optimal dataset was created using the optimal policy specified by the environment [7]. Therefore, this dataset has a success rate of 100% in every episode, i.e. every decision made by the agent leads to the most optimal result. The suboptimal dataset will be created with a 50% success rate in each episode to represent scenarios of imperfect decision-making. Suboptimal datasets are created by training a Deep Q-Network (DQN) model using the stable-baseline3 library [25] with a CNN architecture based on [26] until it successfully completes 20 out of 40 tasks within the environment. The training will be halted once this milestone is reached and the current DQN model is saved. This trained model then interacts with the training environment to generate the dataset. The mixed dataset is generated by uniformly selecting between optimal and suboptimal policies (50% each) to determine the action taken.

4.3 Implementation Details of BC

The baseline algorithm must now be chosen, which will help to compare and verify the generalization capabilities of SAC+BC. BC is chosen as it has demonstrated strong performance as a baseline for offline RL algorithms [6]. The BC algorithm used in this study is based on the DiscreteBC algorithm from the d3rlpy library [27]. The algorithm makes use of a loss function designed to minimize the negative log-likelihood of the observed actions, represented as [27]:

$$\mathcal{L}_{\text{BC}}(\theta) = E_{a_t, s_t \sim \mathcal{D}} \left[-\sum_a p(a|s_t) \log \pi_\theta(a|s_t) \right]$$

In this formula, θ represents the parameters of the policy π_θ. The term $E_{a_t, s_t \sim \mathcal{D}}$ denotes the expectation taken over the distribution of state-action pairs (s_t, a_t) sampled from the dataset $\mathcal{D}$. The term $p(a|s_t)$ is a one-hot vector representing the observed action a taken in state s_t from the dataset, meaning $p(a|s_t) = 1$ for the action a that was taken and 0 for all other actions. Lastly, $\log \pi_\theta(a|s_t)$ is the log probability of taking action a given state s_t under the policy π_θ. This setup aims to match the policy closely to the actions in the training dataset [27].

4.4 Implementation Details of SAC and SAC+BC

The second baseline is the standard SAC implementation. This implementation is derived from the SAC-N version

available on GitHub [28], based on the work by An et al. [29]. The primary distinction from the SAC-N implementation, which utilizes N Q-ensemble networks, is that this standard SAC implementation uses only 2 Q-networks. The SAC actor loss is formulated as follows [28]:

$$\mathcal{L}_{\text{SAC}}(\theta) = E_{s\sim\mathcal{D},a\sim\pi_\theta}\left[\pi(a|s)\left(\alpha\log(\pi(a|s)) - Q_{\min}(s,a)\right)\right]$$

where $\pi(a|s)$ represents the policy distribution over actions a given state s, $Q_{\min}(s,a)$ is the minimum Q-value across the two critic networks for the action-state pair, and α is the temperature parameter that balances the trade-off between exploration and exploitation.

Note that SAC+BC integrates a simplified BC term into the SAC actor loss function to improve computational efficiency. Calculating the log probability only for actions sampled from the replay buffer avoids the overhead of summing over all possible actions. The SAC+BC actor loss, incorporating the simplified BC term, is defined as:

$$\mathcal{L}_{\text{SAC+BC}}(\theta) = \mathcal{L}_{\text{SAC}}(\theta) + \beta E_{s\sim\mathcal{D},a_{\text{rb}}\sim\mathcal{D}}\left[-\log\pi(a_{\text{rb}}|s)\right]$$

where a_{rb} are the actions from the replay buffer representing desired actions, and β is a hyperparameter that controls the influence of the behavior cloning term on the overall loss function. This term pulls the policy distribution towards actions known to be effective from past experiences or expert demonstrations.

4.5 Hyperparameter Tuning

Before comparing the performance of SAC+BC, SAC, and BC, the hyperparameters of each algorithm need to be tuned. This step is crucial to ensure that each algorithm performs at its best in the given environment, as suboptimal hyperparameters could lead to incorrect conclusions due to poor performance [30]. It is also important to make a similar effort to tune the hyperparameters for each algorithm, including the baseline, to ensure a fair comparison.

The Optuna framework was used to tune the parameters of BC, SAC, and SAC+BC. Optuna uses a tree-structured parzen estimator (TPE), which proposes hyperparameter values based on the defined ranges in an objective function. TPE models the probability of achieving better trial results and intelligently samples new hyperparameters to explore the most promising areas efficiently. This approach helps to optimize the search process so that it quickly converges to optimal hyperparameter settings for the given objective, which in our case is to achieve high mean rewards. [31]

BC, SAC, and SAC+BC are specifically tuned on the optimal, suboptimal, and mixed datasets. The tuning for each algorithm is done with three hyperparameters, which is not optimal. However, more parameters would have made tuning infeasible since the tuning is limited by the local machine used. Each hyperparameter combination is run with five different seeds (0, 1, 2, 3, 4) to increase the stability of each trial and reduce the stochastic/random nature of each algorithm. The range of hyperparameters used per algorithm and dataset can be seen in Tables 1, 2, and 3.

Due to the demanding nature of hyperparameter tuning and the limitations of the local machine used, the range of training steps was reduced to allow a manageable number of trials. Initial training and evaluation were performed on the training environment using default parameters with 50,000 training steps. These results helped determine the appropriate range of training steps for the tuning with Optuna. For example, SAC performed best at around 25,000 training steps, leading to the selection of a range from 15,000 to 30,000 steps for this algorithm. The number of trials was set based on the overall time needed for each algorithm's tuning process. SAC, which needs the most time because of the higher number of training steps, was limited to 20 trials, while BC and SAC+BC were each set to 50 trials.

The results of the hyperparameter tuning can be found in the appendix under Figures 7, 8, 9, 10, 11, 12, 13, 14, and 15. Based on these results, the best hyperparameters were selected as shown in Tables 4, 5, and 6. Note that the hyperparameters were selected based on their performance in the training set, as using the test sets for validation would compromise the integrity of the test set.

4.6 Generalization Evaluation Metrics

The average reward and standard deviation are calculated to assess the algorithms' performance. Specific pairs of seeds are used for training and evaluation: each model is trained with one of five seeds (10, 11, 12, 13, 14) and then evaluated with a corresponding seed (20, 21, 22, 23, 24), respectively. For example, the model trained with seed 10 is evaluated with seed 20, the model with seed 11 with seed 21, and so on. Thus, in the end, every mean reward and standard deviation is the average of five results.

5 Experimental Results

This section presents the results for BC, SAC, and SAC+BC in the MiniGrid Four-Room Environment. We start with the learning curve and then present the generalization results under different dataset conditions.

5.1 Learning Curves

Behavioral Cloning (BC) converges rapidly with a perfect mean reward of 1.0 on optimal datasets. It only achieves mean rewards of 0.3 and 0.2, respectively, on reachable and unreachable tasks. With suboptimal datasets, it peaks at 0.4 in training environments and 0.2 in reachable environments, slightly decreasing in unreachable environments. Mixed dataset training enhances performance, reaching mean rewards of 0.4 in reachable and 0.25 in unreachable environments after 50,000 steps (Figures 16, 17, 18).

SAC performs poorly in all environments, failing to achieve rewards with optimal and suboptimal datasets. With a mixed dataset, it achieves a very low mean reward below 0.2 in all environments (Figures 19, 20, 21).

SAC+BC initially excels with the optimal dataset, achieving full rewards within 1,000 steps, but its performance declines over time, dropping to a mean reward of 0.05 by 50,000 steps, with similar decreases in both reachable and unreachable environments. With the suboptimal dataset, SAC+BC stabilizes at a mean reward of 0.4 in the training environment after 30,000 to 50,000 steps, maintaining mean rewards at around 0.15 in the reachable and unreachable environments, with slightly better performance in the reachable setting. Contrastingly, SAC+BC underperforms with the mixed dataset, as one seed's near-zero rewards significantly reduce the average. However, the remaining seeds perform comparably to BC (Figures 22, 23, 24).

5.2 Generalization to New Environments using Optimal Data

This section examines the generalization performance of SAC+BC, using BC and SAC as the baseline for generalization. The analysis uses an optimal dataset with 40 episodes and 20,000 training steps to train the models. Figure 1 displays the performance of BC, SAC, and SAC+BC across different environments, such as train, reachable test, and unreachable tasks.

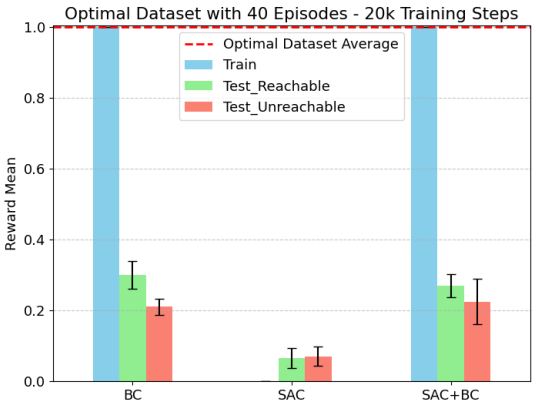

Figure 1: Performance on the Optimal Dataset. The error bar represents the standard deviation. Results are averaged over five seeds.

Figure 1 illustrates that SAC+BC, when trained on optimal data, performs comparably to BC across all environments, indicating no significant generalization gap. As expected, SAC+BC and BC show slightly better reachable generalization than unreachable generalization. Conversely, SAC exhibits consistently low performance, with mean rewards below 0.1 across all tested environments and achieving no rewards in the training environment. This suggests that SAC lacks generalization and learning capabilities in this setup, and the inclusion of the BC term is likely a crucial factor in the improved performance of SAC+BC.

5.3 Generalization to New Environments using Mixed Optimal-Suboptimal Data

This subsection evaluates the generalization ability of SAC+BC to new environments when trained on a mixed dataset over 50,000 training steps, as shown in Figure 2. BC consistently outperforms both SAC and SAC+BC across all environments under these conditions. Notably, SAC+BC shows significant variability in performance, evidenced by a large standard deviation. This variability stems from inconsistencies where SAC+BC performed comparably to BC in four out of five seeds but drastically underperformed in the fifth, achieving nearly zero rewards across all environments. This suggests that while SAC+BC can potentially match BC's generalization performance, it lacks reliability, which leads to a significant generalization gap to BC.

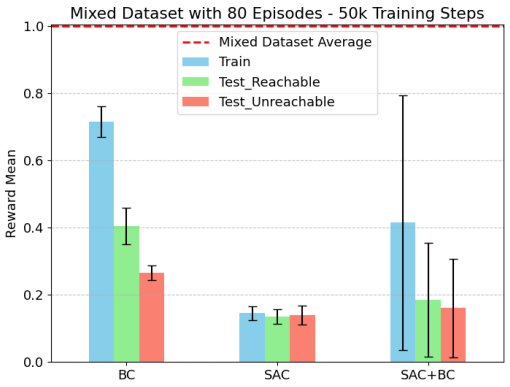

Figure 2: Performance on the Mixed Dataset. The error bar represents the standard deviation. Results are averaged over five seeds.

Moreover, both BC and SAC+BC show better results than SAC, though the margin between SAC+BC and SAC is relatively small in the test environments. This indicates that while SAC+BC improves upon SAC's capabilities, the enhancement is modest and does not bridge the gap to BC, especially in terms of consistent generalization to new environments. To thoroughly understand the instability in SAC+BC's performance, an analysis with significantly more than five seeds would be required. However, based on the results presented, the results are consistent with [6], which states that BC outperforms state-of-the-art offline RL methods when learning from a mixed expert-suboptimal dataset.

5.4 Generalization to New Environments using Suboptimal Data

This section explores the generalization capabilities of SAC+BC when trained on a suboptimal dataset consisting of 80 episodes over 20,000 training steps shown in Figure 3.

SAC underperforms with the suboptimal dataset across all environments. In contrast, SAC+BC shows a notable improvement over SAC. Although it does not surpass BC's overall performance, SAC+BC achieves nearly identical results to

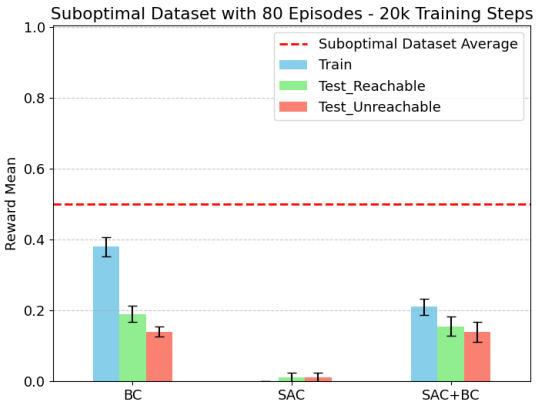

Figure 3: Performance on the Suboptimal Dataset. The error bar represents the standard deviation. Results are averaged over five seeds.

BC on the unreachable tasks and performs only slightly worse on the reachable tasks. However, BC clearly outperforms SAC+BC in the training environment, achieving a mean reward just below 0.4 compared to SAC+BC's just above 0.2. These results indicate that the difference in the unreachable generalization between BC and SAC+BC is almost zero and is also marginal for the reachable generalization.

5.5 The Effect of Data Diversity on Generalization

The impact of data diversity on generalization has been implicitly assessed through optimal, suboptimal, and mixed datasets, each embodying varying levels of diversity. The mixed dataset, comprising experiences from both optimal and suboptimal policies, is considered the most diverse. While one can argue about which of the optimal or suboptimal datasets is more diverse, the key observation is that if SAC+BC achieves better generalization with the mixed dataset than the optimal and suboptimal datasets, this suggests that greater diversity could improve generalization.

In the preceding sections, we explored how the SAC+BC model generalizes to new environments, revealing no generalization gap when trained on the optimal or suboptimal datasets. However, a significant gap was observed with the mixed dataset (Figure 2). While BC generalizes best on the mixed dataset, SAC+BC generalizes best on the optimal dataset. This suggests that data quality may play a more critical role than diversity in the performance of SAC+BC, which shows a performance decline from optimal to mixed and then to suboptimal datasets. However, the considerable variability in performance, highlighted by the large standard deviations in Figure 2, indicates potential instability when using diverse data sources. This instability makes it difficult to draw reliable conclusions about the impact of data diversity on SAC+BC. There can be many reasons for the instability, e.g., incorrect tuning of the hyperparameters or the use of too few seeds.

Conversely, SAC generalizes like BC best with the mixed dataset, achieving consistent performance across the training and both testing environments. Although the performance is not high, the added diversity in the mixed dataset might have played a crucial role in the improved performance. Thus, greater diversity seems to increase the generalizability of BC and SAC, which is consistent with previous research [6].

5.6 The Effect of Data Size on Generalization

The influence of dataset size on generalization was investigated by creating datasets with 40, 80, 200, and 400 episodes. These sizes were selected based on the representation of the tasks in the training environment. At 40 episodes, each of the 40 tasks appears once during the dataset's creation. For 400 episodes, however, each task appears ten times, so each task appears more frequently within the larger dataset.

Figure 25 shows that increasing the optimal dataset size has no significant effect on performance for all algorithms.

However, when training the models on the mixed dataset, as shown in Figure 4, increasing the data size enhanced the performance significantly of both BC and SAC+BC across all environments, particularly in the training environment. In contrast, the data size had no noteworthy effect on SAC.

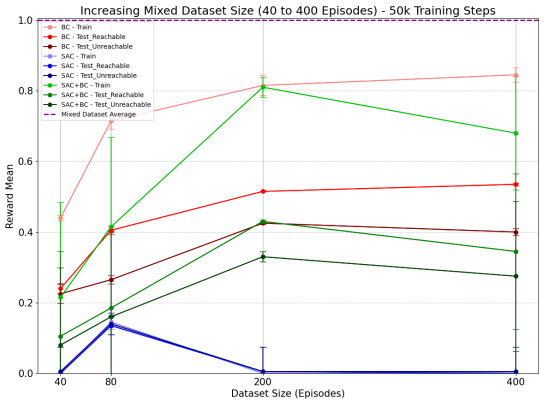

Figure 4: Effect of an Increasing Mixed Dataset Size with 50k Training Steps. The error bar represents the standard deviation. Results are averaged over five seeds.

Training the models with the suboptimal dataset showed in Figure 5 that increasing the data size affected all algorithms positively. SAC, in particular, performed well in all environments, increasing from a mean reward of just over 0 to around 0.25-0.38, while the average of the suboptimal dataset was 0.5. In fact, SAC performed just as well as SAC+BC when the dataset was extended to 400 episodes. However, BC also had a big jump in the training environment and even slightly outperformed the dataset.

Overall, increasing the dataset size affected the generalization of BC, SAC, and SAC+BC positively when trained on subop-

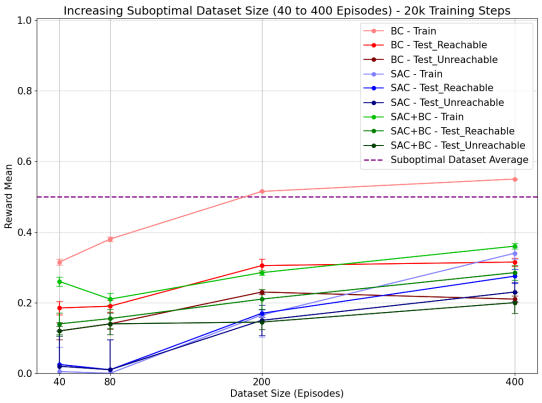

Figure 5: Effect of an Increasing Suboptimal Dataset Size with 20k Training Steps. The error bar represents the standard deviation. Results are averaged over five seeds.

timal and mixed datasets. However, it had no effect when trained on the optimal dataset.

6 Responsible Research

6.1 Ethics

In scientific research, particularly in machine learning, ethical data handling is essential. The datasets used for learning and evaluating algorithms such as BC, SAC, and SAC+BC have been conducted transparently and justifiably. This study strictly avoids fabricating results or unjustifiably discarding data, as such practices would compromise the integrity of this research findings. Moreover, addressing bias is equally critical. Each data source has been evaluated for its origins and ethical implications, with particular attention to biases that might strengthen societal inequalities. This research also adheres to the principles outlined in the *Netherlands Code of Conduct for Research Integrity*, such as honesty, scrupulousness, transparency, independence, and responsibility [32]. All contributions are original, properly credited, and cited. This paper follows stringent plagiarism policies and clearly defines authorship criteria to ensure integrity.

6.2 Reproducibility

To ensure the reproducibility of the findings, this research adheres strictly to the FAIR principles (Findable, Accessible, Interoperable, and Reusable) [33].

Findable: The datasets and algorithms from this study are available in a public GitHub repository, with datasets in a clearly labeled ”datasets” folder and algorithms in the root directory, ensuring easy replication and attribution of results. GitHub Link: https://github.com/AxelGeist/multi-task-offline-reinforcement-learning. ***Accessible:*** Since the GitHub repository is public, anyone can access the datasets and algorithms used to generate all the results. ***Interoperable:*** The datasets are structured in d4rl format [34] to ensure easy integration into common offline RL algorithms. This simplifies various scientific investigations through seamless combination with different data sources. ***Reusable:*** The experiments used specific seeds to ensure replicability. A requirements.txt file in the GitHub repository details the software libraries needed to recreate the study’s computational environment and reproduce the results accurately.

7 Discussion

Data Quality Data quality significantly influences generalization. The experiments showed that SAC+BC and BC displayed similar test performance on optimal and suboptimal datasets. However, BC outperformed SAC+BC on the mixed dataset. This shows that the generalization gap between BC and SAC+BC was zero on the optimal dataset and minimal on the suboptimal dataset. Despite a noticeable generalization gap between BC and SAC+BC on the mixed dataset, the performance of SAC+BC was significantly influenced by sensitivity to a particular seed, indicating that the actual gap might differ. The gap may be only marginal, meaning that SAC+BC generalizes best in a mixed dataset, which is also the case for BC and SAC. However, using only five seeds limits the reliability of these results, so further investigation is required to assess the performance of SAC+BC on the mixed dataset accurately. Conversely, SAC consistently underperformed in the offline setting across all dataset qualities, which is to be expected from an algorithm that was primarily developed for online RL. Overall, SAC+BC demonstrates a smaller generalization gap to BC compared to other offline RL methods [6], showing that adding the BC term significantly enhances SAC’s generalization capabilities.

Data Size The impact of data size on model generalization varies significantly depending on the dataset’s characteristics. Our findings reveal that increasing the data size enhances generalization for BC, SAC, and SAC+BC when using mixed and suboptimal datasets. However, data size increases do not improve performance with optimal data. In cases of optimal data, where actions are typically deterministic, enlarging the dataset does not introduce new types of transitions, thus not contributing to diversity. Conversely, for mixed and suboptimal datasets, which are compiled using stochastic actions, increasing the dataset size adds a variety of new transitions, enhancing data diversity. These results are thus consistent with previous research indicating that simply increasing the amount of data without increasing diversity does not significantly improve adaptation to new environments. Furthermore, this study notes a significant improvement in SAC’s overall performance with larger suboptimal datasets. This enhancement likely stems from the fact that a more extensive suboptimal dataset closely simulates online environmental interactions, which are rich in unique experiences. This increase in data diversity appears to benefit SAC, which relies on diverse data to approximate real-world online learning scenarios more effectively. Overall, the study

suggests that BC and SAC+BC improve generalization when the dataset is enlarged, but only if this increases diversity.

Data Diversity The study indicates that data diversity enhances the generalization capabilities of both BC and SAC, as both generalize best on the most diverse data (mixed). This aligns with findings from previous research such as [6] and [21], which suggest that diversity generally increases generalizability. However, the impact of diversity on SAC+BC is less clear. Since data diversity has not been explicitly analyzed, the effect of data diversity needs to be figured out by implicitly analyzing data quality and size. When analyzing data quality, SAC+BC performs best with the optimal dataset and worst with the suboptimal dataset, with performance on the most diverse dataset (mixed) falling in between. Thus, data quality analysis did not help determine the impact of data diversity. However, as mentioned before, SAC+BC might be able to perform better on the mixed dataset than in Figure 2 if more seeds are used or if the hyperparameters of SAC+BC are better tuned so that it performs more stably, which would then possibly show that diversity increases the generalizability of SAC+BC. However, this could not be shown with the current experimental setup, so this should be considered in future studies. Analyzing data size shows that increasing data size, combined with diversity, positively affects the generalization capabilities of BC and SAC+BC.

Limitations This study is subject to several limitations. The experiments were conducted in only one environment, which is relatively simple, limiting the results to this environment. In addition, the tuning of the hyperparameters was limited by the computational capacities of the local computer used. This limitation led to suboptimal values of the hyperparameters, which are reflected in the learning curves. This problem was particularly evident in SAC+BC, where the performance did not converge even after 50,000 training steps for the suboptimal and mixed dataset. In addition, SAC+BC significantly dropped its performance on the optimal dataset after approximately 21,000 training steps. Furthermore, a low number of seeds was used (five), which can be seen from the unstable learning curves in Figure 17 and the large standard variation for SAC+BC on the mixed dataset in Figure 2.

Future Work Several steps can be taken in future research to address these limitations and enhance the robustness and applicability of the findings. Future studies should consider employing a more diverse array of environments, including those that are more complex and varied. This approach would help in assessing the generalization capabilities in more complex environments. Additionally, by using more powerful computational resources, hyperparameter tuning can be conducted more extensively and potentially more optimally. This could enhance the performance of the models, particularly in terms of stability across various datasets. Furthermore, increasing the number of seeds for the experiments could provide a more reliable and stable evaluation of the models. This increase would help mitigate the effects of variability seen in the mixed dataset and provide a clearer picture of the model's performance. For models like SAC+BC, extending the number of training steps beyond 50,000 might allow the models more time to converge, particularly on suboptimal or mixed datasets.

8 Conclusion

The purpose of this study is to determine whether SAC+BC could effectively generalize to new tasks within a multi-task RL environment. It was demonstrated that SAC+BC could generalize in multiple scenarios comparably or nearly as well as BC, thus showing better reachable and unreachable generalization than many other offline RL algorithms [6]. However, BC remains a strong baseline, as it mostly matches or exceeds the performance of SAC+BC. This answers the first research question and indicates that SAC combined with BC is effective in new task generalizations given specific datasets.

To answer the second research question, which offline dataset characteristics are critical for the success of SAC+BC in a multi-task RL environment, the quality, size, and diversity of the datasets play an important role. Dataset quality is crucial, with the performance of SAC+BC peaking with high-quality data, becoming less stable with mixed-quality data, and lowest with suboptimal data, highlighting the need for more research to better understand the impact of mixed datasets. Size enhances generalizability only when it introduces greater diversity, so simply increasing the amount of data does not necessarily improve generalizability. Therefore, a careful selection of dataset attributes is required to ensure generalizability for SAC+BC.

Despite these results, the experiments encountered significant limitations due to the constraints of the available hardware. Therefore, it would be advisable to undertake a more thorough tuning of hyperparameters and the use of more than five seeds. Such an effort could more robustly validate the strength of the results and potentially show whether SAC+BC could close the generalization gap to BC.

References

[1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022.

[2] J. Fu, V. Kumar, O. Nachum, G. Tucker, and S. Levine, "Dexterity from a distance: Learning manipulation from remote supervision," *Journal of Robotics and Automation*, vol. 35, no. 4, pp. 123–135, 2020.

[3] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3675–3682, 2020.

[4] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *arXiv preprint arXiv:2006.04779*, 2020.

[5] A. Mediratta and A. Rajeswaran, "A closer look at the effectiveness of behavioral cloning for offline reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 200–212, 2023.

[6] I. Mediratta, Q. You, M. Jiang, and R. Raileanu, "The generalization gap in offline reinforcement learning," *arXiv preprint arXiv:2312.05742*, 2024.

[7] M. Weltevrede, "Four room: A simple 4-room grid world environment to test generalisation behaviour of rl agents," 2022, accessed: Apr. 28, 2024. [Online]. Available: https://github.com/MWeltevrede/four_room

[8] MiniGrid Documentation, "Four room," 2024, accessed: Jun. 5, 2024. [Online]. Available: https://minigrid.farama.org/environments/minigrid/FourRoomsEnv/

[9] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2019.

[10] N. Hansen and X. Wang, "Generalization in reinforcement learning by soft data augmentation," *arXiv preprint arXiv:2011.13389*, 2021.

[11] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus, "Automatic data augmentation for generalization in deep reinforcement learning," *arXiv preprint arXiv:2006.12862*, 2021.

[12] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," *arXiv preprint arXiv:1912.01588*, 2020.

[13] J. Farebrother, M. C. Machado, and M. Bowling, "Generalization and regularization in dqn," *arXiv preprint arXiv:1810.00123*, 2020.

[14] C. Lyle, M. Rowland, W. Dabney, M. Kwiatkowska, and Y. Gal, "Learning dynamics and generalization in reinforcement learning," *arXiv preprint arXiv:2206.02126*, 2022.

[15] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of zero-shot generalisation in deep reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 76, p. 201–264, Jan. 2023. [Online]. Available: http://dx.doi.org/10.1613/jair.1.14174

[16] B. Mazoure, I. Kostrikov, O. Nachum, and J. Tompson, "Improving zero-shot generalization in offline reinforcement learning using generalized similarity functions," *arXiv preprint arXiv:2111.14629*, 2021.

[17] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *arXiv preprint arXiv:2106.06860*, 2021.

[18] OpenAI, "Twin delayed ddpg (td3)," accessed: May 27, 2024. [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/td3.html#twin-delayed-ddpg

[19] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2021.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[21] M. Weltevrede, M. T. J. Spaan, and W. Böhmer, "The role of diverse replay for generalisation in reinforcement learning," *arXiv preprint arXiv:2306.05727*, 2023.

[22] M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, and H. van Hasselt, "Multi-task deep reinforcement learning with popart," *arXiv preprint arXiv:1809.04474*, 2018.

[23] A. Kumar, J. Hong, A. Singh, and S. Levine, "When should we prefer offline reinforcement learning over behavioral cloning?" *arXiv preprint arXiv:2204.05618*, 2022.

[24] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[25] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: https://www.nature.com/articles/nature14236

[27] T. Seno and M. Imai, "d3rlpy: An offline deep reinforcement learning library," *arXiv preprint arXiv:2111.03788*, 2022.

[28] C. Horsch, "offline-sac-n," 2024, accessed: May 21, 2024. [Online]. Available: https://github.com/chorsch/offline-sac_n

[29] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified q-ensemble," *arXiv preprint arXiv:2110.01548*, 2021.

[30] H. J. P. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of tuning hyperparameters of machine learning algorithms," *arXiv preprint arXiv:2007.07588*, 2020.

[31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[32] The Association of Universities in the Netherlands (VSNU), "Netherlands code of conduct for research integrity," 2018, accessed: Jun. 3, 2024. [Online]. Available: https://www.nwo.nl/en/netherlands-code-conduct-research-integrity

[33] M. Barker, N. P. Chue Hong, D. S. Katz *et al.*, "Introducing the fair principles for research software," *Scientific Data*, vol. 9, no. 622, 2022. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1038/s41597-022-01710-x

[34] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2021.

A Structure of the Experimental Setup

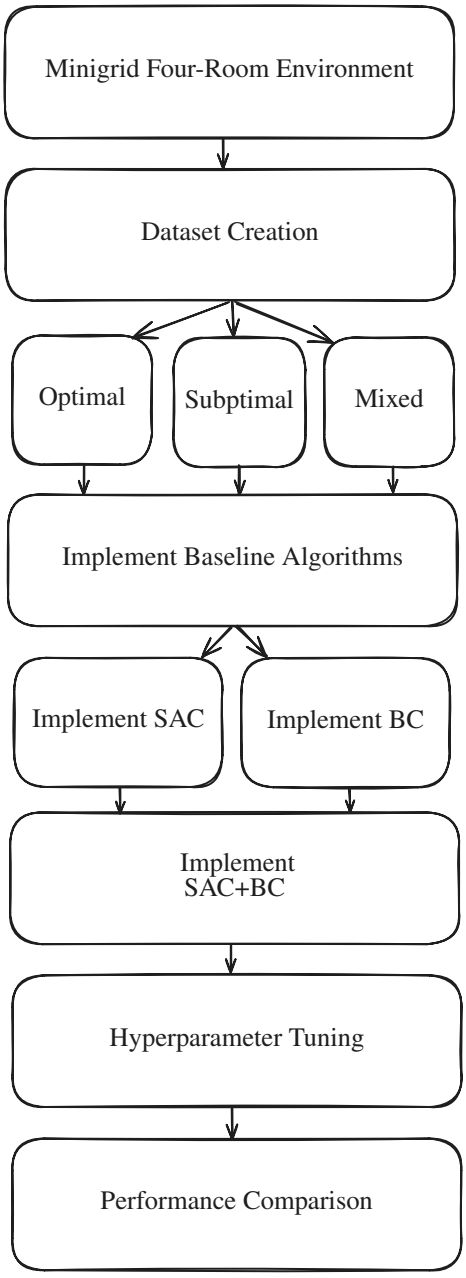

Figure 6: Structure of the Experimental Setup

B Hyperparameter Tuning

Dataset	Hyperparameter	Range/Options
Optimal	Training Steps	50 to 1000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Batch Size	[32, 64, 100, 128]
Suboptimal	Training Steps	400 to 4000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Batch Size	[32, 64, 100, 128]
Mixed	Training Steps	50 to 1000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Batch Size	[32, 64, 100, 128]

Table 1: Ranges used for Hyperparameter Tuning on BC with Optuna [31].

Dataset	Hyperparameter	Range/Options
Optimal	Training Steps	15000 to 30000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Gamma	0.95 to 0.99
Suboptimal	Training Steps	15000 to 30000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Gamma	0.95 to 0.99
Mixed	Training Steps	15000 to 30000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Gamma	0.95 to 0.99

Table 2: Ranges used for Hyperparameter Tuning on SAC with Optuna [31].

Dataset	Hyperparameter	Range/Options
Optimal	Training Steps	50 to 1000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Beta	0 to 1.0
Suboptimal	Training Steps	500 to 3000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Beta	0 to 1.0
Mixed	Training Steps	50 to 1000
	Learning Rate	$[1 \times 10^{-4}, 3 \times 10^{-4}]$
	Beta	0 to 1.0

Table 3: Ranges used for Hyperparameter Tuning on SAC+BC with Optuna [31].

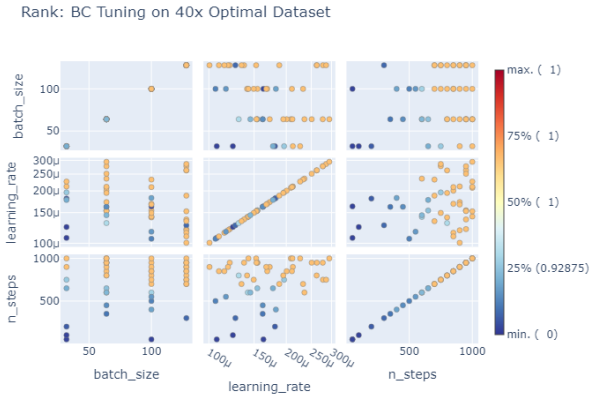

Figure 7: Hyperparameter Tuning Results for BC on the Optimal Dataset. Results are averaged over 5 seeds.

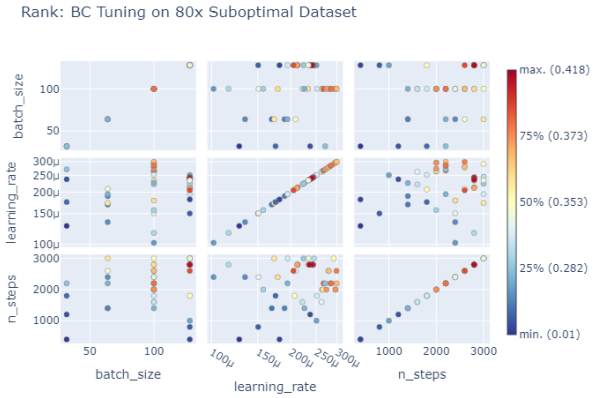

Figure 8: Hyperparameter Tuning Results for BC on the Suboptimal Dataset. Results are averaged over 5 seeds.

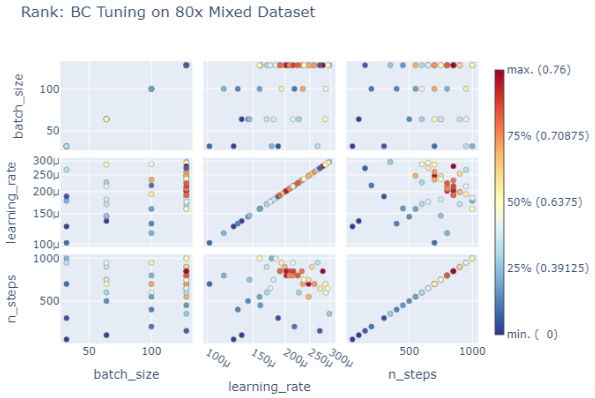

Figure 9: Hyperparameter Tuning Results for BC on the Mixed Dataset. Results are averaged over 5 seeds.

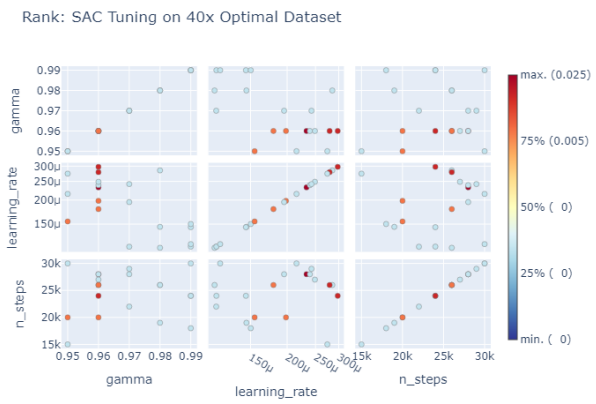

Figure 10: Hyperparameter Tuning Results for SAC on the Optimal Dataset. Results are averaged over 5 seeds.

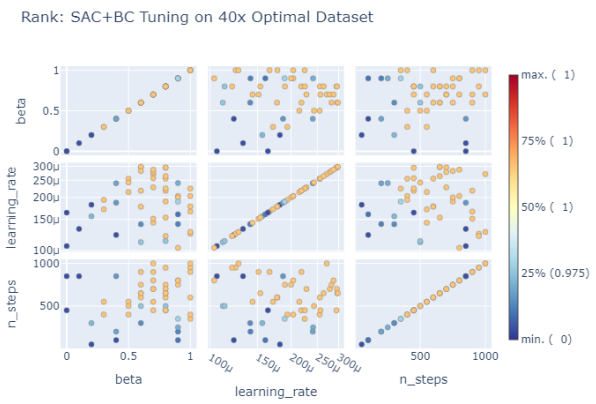

Figure 13: Hyperparameter Tuning Results for SAC+BC on the Optimal Dataset. Results are averaged over 5 seeds.

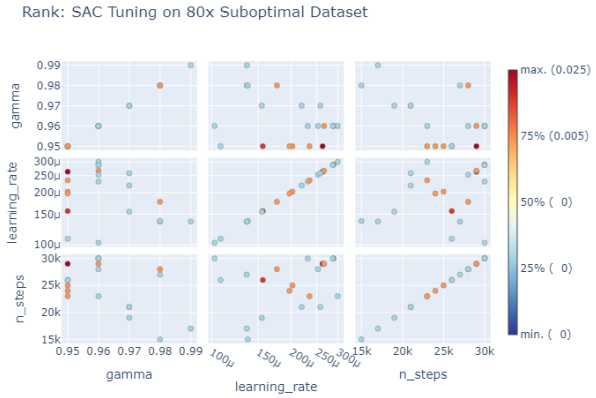

Figure 11: Hyperparameter Tuning Results for SAC on the Suboptimal Dataset. Results are averaged over 5 seeds.

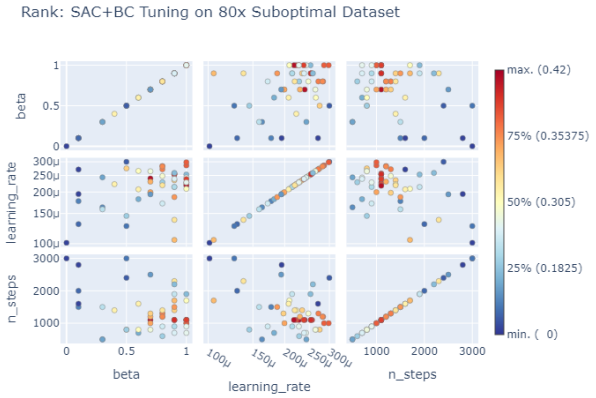

Figure 14: Hyperparameter Tuning Results for SAC+BC on the Suboptimal Dataset. Results are averaged over 5 seeds.

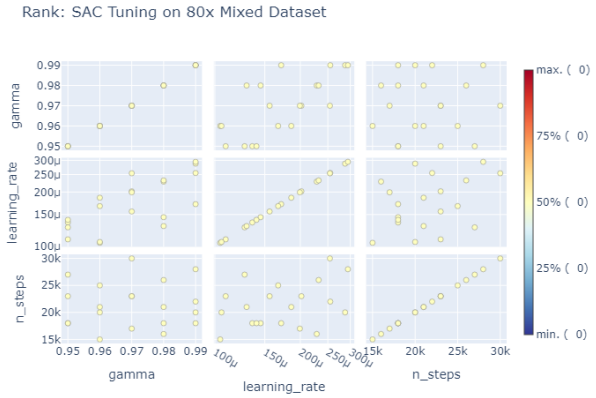

Figure 12: Hyperparameter Tuning Results for SAC on the Mixed Dataset. Results are averaged over 5 seeds.

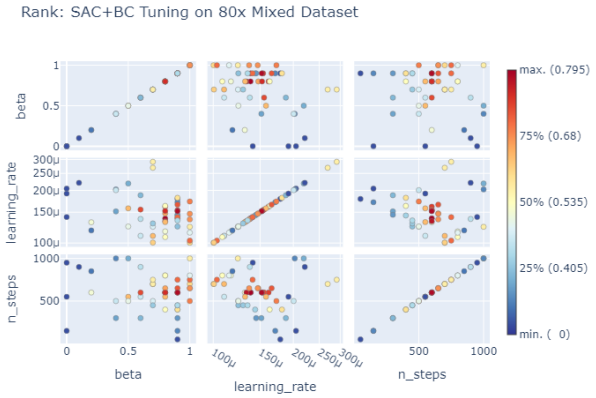

Figure 15: Hyperparameter Tuning Results for SAC+BC on the Mixed Dataset. Results are averaged over 5 seeds.

Dataset	Hyperparameter	Value
Optimal	Learning Rate	1.5×10^{-4}
	Batch Size	100
Suboptimal	Learning Rate	2.4×10^{-4}
	Batch Size	128
Mixed	Learning Rate	2.8×10^{-4}
	Batch Size	128

Table 4: Selected hyperparameters for BC. The hyperparameters that are not displayed are based on the default values of DiscreteBC in the d3rlpy library [27].

Dataset	Hyperparameter	Value
Optimal	Learning Rate	2.3×10^{-4}
	Gamma	0.96
	Hidden Dimensions	256
	Critics	2
	Tau	0.005
	Buffer Size	1,000,000
	Batch Size	256
	Epochs	5000
	Updates on Epoch	10
Suboptimal	Learning Rate	2.6×10^{-4}
	Gamma	0.95
	Hidden Dimensions	256
	Critics	2
	Tau	0.005
	Buffer Size	1,000,000
	Batch Size	256
	Epochs	5000
	Updates on Epoch	10
Mixed	Learning Rate	2.56×10^{-4}
	Gamma	0.99
	Hidden Dimensions	256
	Critics	2
	Tau	0.005
	Buffer Size	1,000,000
	Batch Size	256
	Epochs	5000
	Updates on Epoch	10

Table 5: Selected hyperparameters for SAC. Learning rate and gamma were selected using Optuna [31] and the remaining parameters were default values from [28].

Dataset	Hyperparameter	Value
Optimal	Learning Rate	3.0×10^{-4}
	Beta	1.0
	Gamma	0.99
	Hidden Dimensions	256
	Critics	2
	Tau	0.005
	Buffer Size	1,000,000
	Batch Size	256
	Epochs	5000
	Updates on Epoch	10
Suboptimal	Learning Rate	2.3×10^{-4}
	Beta	1.0
	Gamma	0.99
	Hidden Dimensions	256
	Critics	2
	Tau	0.005
	Buffer Size	1,000,000
	Batch Size	256
	Epochs	5000
	Updates on Epoch	10
Mixed	Learning Rate	1.4×10^{-4}
	Beta	1.0
	Gamma	0.99
	Hidden Dimensions	256
	Critics	2
	Tau	0.005
	Buffer Size	1,000,000
	Batch Size	256
	Epochs	5000
	Updates on Epoch	10

Table 6: Selected hyperparameters for SAC+BC. Learning rate and Beta were selected using Optuna [31] and the remaining parameters were default values from [28].

C Learning Curves

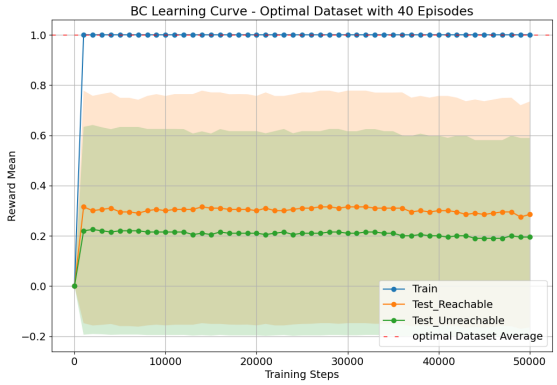

Figure 16: Learning Curve of BC with the Optimal Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

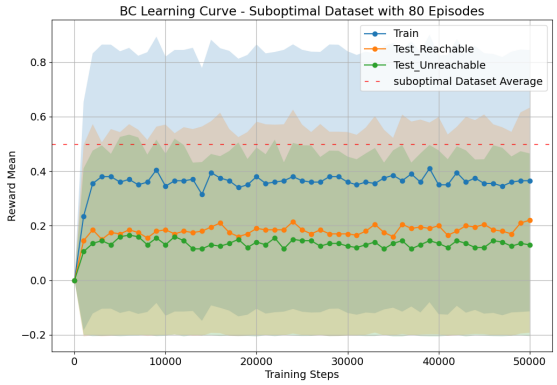

Figure 17: Learning Curve of BC with the Suboptimal Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

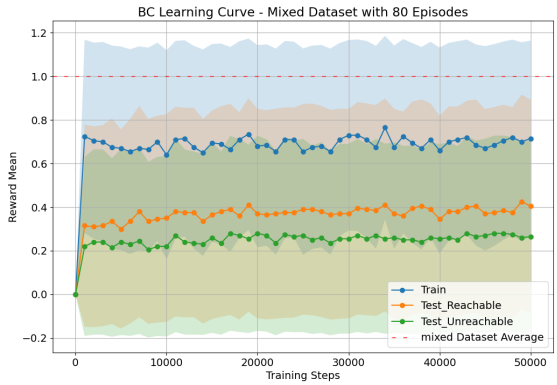

Figure 18: Learning Curve of BC with the Mixed Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

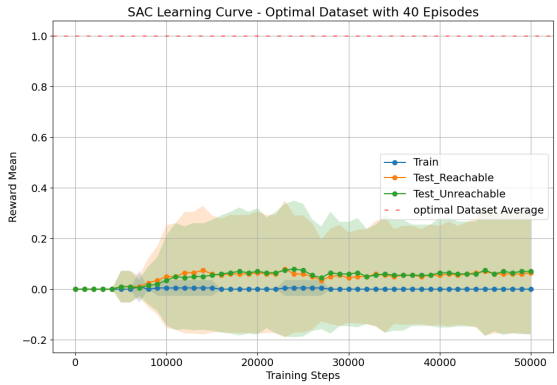

Figure 19: Learning Curve of SAC with the Optimal Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

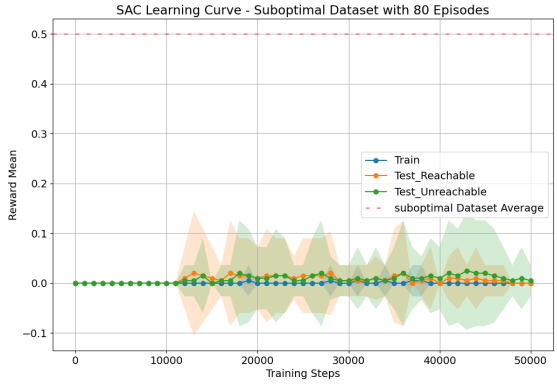

Figure 20: Learning Curve of SAC with the Suboptimal Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

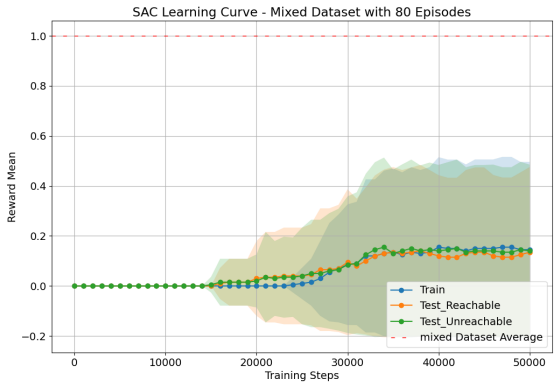

Figure 21: Learning Curve of SAC with the Mixed Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

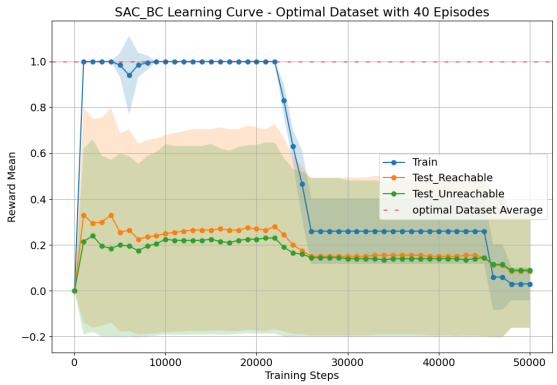

Figure 22: Learning Curve of SAC+BC with the Optimal Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

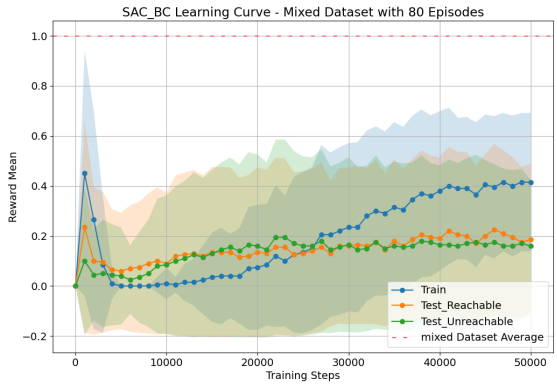

Figure 24: Learning Curve of SAC+BC with the Mixed Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

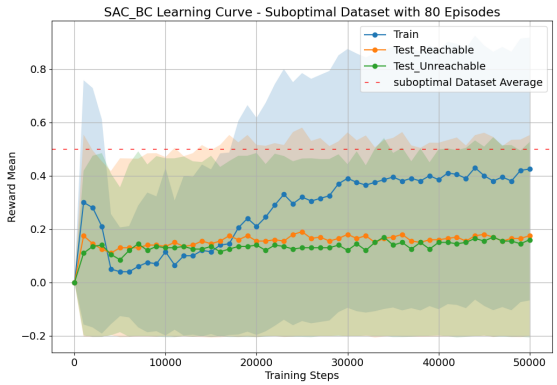

Figure 23: Learning Curve of SAC+BC with the Suboptimal Dataset. The standard deviation is displayed as a shaded area with the corresponding colour. Results are averaged over 5 seeds.

D The Effect of Data Size on Generalization

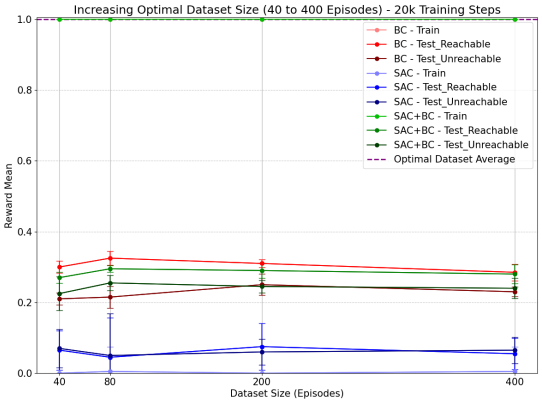

Figure 25: Effect of an Increasing Optimal Dataset Size with 20k Training Steps. The error bar represents the standard deviation. Results are averaged over five seeds.