Closing the Generalization Gap in Offline Reinforcement Learning with WSAC-N

Maxime Museur[1]
Supervisor(s): Matthijs 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: Maxime Museur
Final project course: CSE3000 Research Project
Thesis committee: Matthijs Spaan, Max Weltevrede, Elena Congeduti

Abstract

Recent work has shown that offline reinforcement learning (RL) does not generalize well to new environments compared to behavioral cloning (BC). We propose WSAC-N, an ensemble model of soft actor-critics with weights to de-emphasize actions with high variance. We compare the zero-shot generalization abilities of WSAC-N with the baseline BC in a four-room maze-like environment, testing on unseen tasks. Our findings indicate that WSAC-N has worse zero-shot generalization compared to BC, aligning with previous work. Additionally, we investigate the impact of dataset characteristics on generalization, finding that dataset size has a negligible impact, while the quality of trajectories generally has a positive effect. These results are consistent with prior research. The code for this paper can be found on GitHub.

1 Introduction

Offline reinforcement learning (RL) is a branch of reinforcement learning where, unlike online reinforcement learning, the agent cannot perform actions in the environment (Levine et al., 2020). Offline models can only rely on a static dataset of trajectories. This is useful when the online setting is expensive (e.g. robotics (Singh et al., 2022)) or dangerous (e.g. healthcare (Liu et al., 2020)). On top of that, even if online RL is available, we might still prefer to use previously collected data instead, e.g. when the task requires large datasets to generalize.

Recent work (Ishita Mediratta et al., 2024), however, has shown that state-of-the-art offline RL models[1] do not generalize as well to new environments as behavioral cloning (BC). This shows that more research needs to be done to improve the generalization abilities of offline RL beyond imitation learning.

This work evaluates the ability of two different methods to generalize to new tasks without requiring online fine-tuning or an adaptation period (referred to as zero-shot generalization). We compare a baseline method BC with our proposed algorithm WSAC-N, which combines concepts from two existing methods: SAC-N (An et al., 2021) and SUNRISE DQN (Lee et al., 2021). SAC-N is an ensemble of soft actor-critics (SAC) (Haarnoja et al., 2018), a standard online reinforcement learning (RL) algorithm. SUNRISE DQN, on the other hand, is an online ensemble of actor-critics that uses weights to reduce the impact of actions with high variance or uncertainty. WSAC-N is introduced as a method where SAC-N is modified using the weighting mechanism from SUNRISE DQN. However, in our experiments, WSAC-N shows inferior performance compared to BC on the test sets when it comes to generalization. Additionally, we investigate certain characteristics of the dataset, namely size and quality of the trajectories, and how they influence the generalization abilities of our algorithms. We find that higher quality trajectories generally have a positive impact on zero-shot generalization, while the scale of the dataset does not significantly affect generalization. These findings align with previous research (Ishita Mediratta et al., 2024).

We use the environment from Weltevrede et al. (2023) (see figure 1) to generate train and test datasets. We test for zero-shot generalization by evaluating the model on unseen tasks on test sets provided by the environment. This relatively simple environment makes it manageable to iterate often, or reason about specific scenarios. We would also like to note that simple environments have often shown surprising results that would not have been found in more complex environments, e.g. Baird's counterexample (Baird, 1995) or the environment Riverswim (Szita and Lorincz, 2008).

2 Background

Offline RL. This study examines how well some offline RL algorithms generalize in Markov decision processes. A Markov decision process (MDP) is defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, d_0, r, \gamma)$, where $\mathcal{S}$ is a set of states $\mathbf{s} \in \mathcal{S}$, which may be either discrete or continuous (i.e., multi-dimensional vectors), $\mathcal{A}$ is a set of actions $\mathbf{a} \in \mathcal{A}$, which similarly can be discrete or continuous, T defines a conditional probability distribution of the form $T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ that describes the dynamics of the system, d_0 defines the initial state distribution $d_0(\mathbf{s}_0)$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines a reward function, and $\gamma \in (0, 1]$ is a scalar discount

[1] Namely BCQ (Fujimoto et al., 2019), CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2021), BCT (Chen et al., 2021), and DT (Chen et al., 2021)

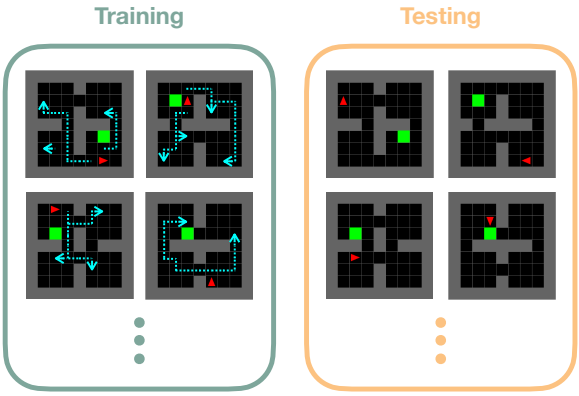

Figure 1: Examples of the 4-room grid environment that we use to train and test our algorithms for zero-shot generalization.

factor. In RL, the goal is to learn a policy π that maximizes the expected reward under the trajectory distribution $\mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \gamma^t \right]$. Offline RL is different to online RL because it cannot perform actions in an environment. Instead, the algorithm can only rely on a static dataset $\mathcal{D}$ of trajectories. $\mathcal{D}$ usually does not contain all possible state-action pairs. This means that the model will eventually have to evaluate actions that are not in the dataset. Errors due to incorrectly evaluating these actions is called distributional shift, and makes it harder to generalize.

We combine two algorithms that use uncertainty measures. This will help avoid distributional shift by preferring actions with a high certainty. We hypothesize that this also leads to better generalization.

Zero-shot generalization. When we refer to generalization in this paper, we mean zero-shot generalization (Kirk et al., 2023). An agent exhibits zero-shot generalization if it performs well on testing tasks that differ from the training tasks without any online fine-tuning or adaptation period.

SAC-N. SAC-N is an ensemble of soft actor-critic (SAC) algorithms (Haarnoja et al., 2018). It aims to maximize the following objective function:

$$J(\theta) = \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_\theta(.|\mathbf{s})} \left[\min_{i=1,\ldots,N} \mathcal{Q}_{\phi_i}(\mathbf{s}, \mathbf{a}) - \beta \log_{\pi_\theta}(\mathbf{a}|\mathbf{s}) \right] \tag{1}$$

Here $\mathcal{Q}_\phi(\mathbf{s}, \mathbf{a})$ follows the principles of Q-learning (Mnih et al., 2015). The Q-function is a state-action value function. The Q-function represents a state-action value function, which is essentially a neural network designed to learn the cumulative discounted reward starting from state $\mathbf{s}$ and action $\mathbf{a}$. We chose SAC-N because it selects the minimum Q-value from the ensemble, which penalizes high-variance estimates (Fujimoto et al., 2019). This encourages the algorithm to prefer actions from the dataset, thereby helping to prevent distributional shift. We hypothesize that this leads to improved generalization.

SUNRISE weights. The weights from (Lee et al., 2021) are defined as follows. The loss function is given by:

$$\begin{aligned} \mathcal{L}_{WQ}(\tau_t, \theta_i) &= w(s_{t+1}, a_{t+1}) \left(Q_{\theta_i}(s_t, a_t) - r_t - \gamma \bar{V}(s_{t+1}) \right)^2 \\ \text{with} \quad \bar{V}(s_t) &= \mathbb{E}_{a_t \sim \pi_\phi} \left[Q_{\bar{\theta}}(s_t, a_t) - \alpha \log \pi_\phi(a_t|s_t) \right] \end{aligned} \tag{2}$$

where $\tau_t = (s_t, a_t, r_t, s_{t+1})$ represents a transition, $a_{t+1} \sim \pi_{\phi_i}(a|s_t)$, and $w(s, a)$ is a confidence weight based on an ensemble of target Q-functions:

$$w(s, a) = \sigma \left(-\bar{Q}_{\texttt{std}}(s, a) * T \right) + 0.5, \tag{3}$$

where $T > 0$ is a temperature parameter, σ is the sigmoid function, and $\bar{Q}_{\texttt{std}}(s, a)$ is the empirical standard deviation of all target Q-functions $\{Q_{\bar{\theta}_i}\}_{i=1}^{N}$." We hypothesize that these weights help prevent

distributional shift by down-weighting actions with high variance (or high uncertainty). This reduces the likelihood of the agent choosing out-of-distribution actions, thereby minimizing errors from incorrectly evaluating actions not present in $\mathcal{D}$. Additionally, we observed a discrepancy between the paper and the code regarding the treatment of actors. While the paper states that the actors are not weighted, the code actually does weight them. We have chosen to follow the paper's approach as it has been empirically shown to perform better.

4-room grid environment. We use the environment from Weltevrede et al. (2023). It is a 4-room grid world derived from the MiniGrid benchmark's FourRooms environment (see figure 1). The environment is fully observable with three possible actions: move forward, turn left, and turn right. During training and testing, the main differences lie in the initial states (or tasks), which vary in the 4-room topology, the agent's starting location and direction, and the goal location. The agent is trained on a fixed number of start states and tested on unseen start states from the same distribution. In this environment, reachability is determined by the goal location and topology. If two states share these attributes, they are mutually reachable; otherwise, they are not. The training dataset consists of 40 different starting states varying in location, direction, goal location, and topology. We can then test for generalization by testing on 40 new, unseen start states. There are two test sets:

- 100% reachable test set: the agent location and direction are different while the topology and goal location remain consistent with training states.
- 0% reachable test set: the agent location, direction, and the grid topology are unseen while the goal location is unchanged.

We evaluate performance on both test sets to gain insights into different types of generalization. For instance, high performance on the unreachable test set might suggest that the algorithm has effectively learned the underlying latent representation of the dataset.

3 Related Work

Generalization in RL. A large body of research has focused on improving generalization in reinforcement learning (RL) through regularization techniques to prevent overfitting (Song et al., 2019; Igl et al., 2019; Farebrother et al., 2020; Wang et al., 2020) and data augmentation methods (Ye et al., 2020; Lee et al., 2020; Laskin et al., 2020; Yarats et al., 2021; Hansen et al., 2021; Ko and Ok, 2023; Zhao et al., 2024). Additionally, considerable research proposes algorithms specifically designed to enhance generalization in RL (Osband et al., 2016; Agarwal et al., 2020; Igl et al., 2019; Touati and Ollivier, 2021; Roberta Raileanu, 2021; Mazoure et al., 2021; Carvalho et al., 2023). However, these efforts are predominantly for online RL whereas we aim to improve generalization for offline RL.

In offline RL, the primary challenge to generalization is distributional shift, which occurs when evaluating actions outside the distribution of the training data (Levine et al., 2020). Our approach addresses this issue by applying uncertainty techniques from (An et al., 2021; Lee et al., 2021) to mitigate the impact of distributional shift.

Furthermore, while various environments exist that offer benchmarks for testing generalization in online RL (Juliani et al., 2019; Küttler et al., 2020; Samvelyan et al., 2021; Frans and Isola, 2023; Albrecht et al., 2022; Yuan et al., 2023; Weltevrede et al., 2023), benchmarks specifically addressing generalization in offline RL remain scarce. As far as we are aware, (Ishita Mediratta et al., 2024) stands as the sole existing benchmark that addresses this aspect. Our study, in collaboration with our research group, aims to fill this gap by evaluating the generalization performance of offline RL algorithms in a distinct environment.

The research most similar to ours is (Ishita Mediratta et al., 2024). They demonstrate that state-of-the-art offline RL algorithms[2] do not generalize as well as BC. However, our research differs in that

[2]Namely BCQ (Fujimoto et al., 2019), CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2021), BCT (Chen et al., 2021), and DT (Chen et al., 2021)

we use a different environment and different algorithms. Additionally, they tune hyperparameters online, whereas we tune them offline. We chose offline hyperparameter tuning because it aligns with the offline nature of our training. Tuning hyperparameters online can lead to optimal settings that may not be suitable for offline training. Moreover, in general, online hyperparameter tuning contradicts the purpose of offline RL—if tuning can be done online, there is no longer a need to train on a static dataset[3].

4 Methodology

We define WSAC-N as SAC-N enhanced with weights from SUNRISE. The loss function for each Q-function Q_{ϕ_i} in WSAC-N is then given by:

$$\mathcal{L}\left(\phi_i\right) = \frac{1}{|B|} \sum_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\in B} \left(w\left(s', a'\right) \left(Q_{\phi_i}\left(\mathbf{s}, \mathbf{a}\right) - y\left(r, \mathbf{s}'\right) \right)^2 \right)$$

where the target function $y(r, \mathbf{s}')$ is:

$$y(r, \mathbf{s}') = r + \gamma \left(\min_{j=1,\ldots,N} Q_{\phi'_j}(\mathbf{s}', \mathbf{a}') - \beta \log \pi_\theta(\mathbf{a}' \mid \mathbf{s}') \right), \quad \mathbf{a}' \sim \pi_\theta(\cdot \mid \mathbf{s}') \tag{4}$$

The policy of the actor is updated by doing gradient ascent on the objective function described in equation 1.

WSAC-N incorporates two uncertainty measures to reduce variance. We hypothesize that this approach leads to better generalization with suboptimal data because the actor is more likely to select actions that have already been observed in the dataset $\mathcal{D}$ compared to other offline RL algorithms like SAC-N. However, we also believe that, compared to BC, WSAC-N is more likely to choose out-of-distribution actions because it is not explicitly trained to imitate the dataset $\mathcal{D}$.

After hyperparameter tuning, we hypothesize that this balance between selecting more certain actions and those outside the distribution becomes optimal. The choice of two uncertainty metrics allows for a potentially more cautious approach if hyperparameter tuning indicates it's necessary.

5 Experimental Setup

5.1 Datasets

We generate the datasets by recording the trajectories of various policies. The starting points, goals, and topologies are determined by the training configuration of the environment (Weltevrede et al., 2023). The datasets are categorized as follows:

- **Expert dataset:** Contains trajectories generated by an optimal policy provided by the environment.
- **Suboptimal dataset:** Contains trajectories generated by a suboptimal policy. It is a DQN model with the same layers as the expert model and was trained until it reached about 50% of the expert performance.
- **Random dataset:** Contains trajectories generated by a random policy.
- **Mixed expert-suboptimal dataset:** Contains a mix of expert and suboptimal trajectories. Each action has a 50% chance of being an expert action or an action from a DQN model that achieved 50% performance of the expert model.

After generating the datasets, we perform offline hyperparameter tuning with validation sets. We use a grid search method and expand the set of hyperparameters if the optimal combination is at the boundary. After completing this process for each dataset category, we conduct our experiments.

[3]It is sometimes still possible to fine-tune online after offline training.

5.2 Experiments

We compare WSAC-N and baseline BC on their generalization abilities across all dataset categories. Training is paused every 2000 steps to evaluate performance on the test sets. By plotting performance with confidence intervals and adjusting until 95% confidence intervals no longer overlap, we determine which algorithm better generalizes across different dataset categories. We expect that WSAC-N will perform comparably to BC on the expert dataset and better on all the suboptimal datasets. For more details on hyperparameter tuning and training, see Appendix A.

Using the results from these experiments, we also investigate the effect of certain characteristics of the dataset. We hypothesize that higher quality trajectories within the dataset will lead to better generalization by our algorithms. Additionally, we expect that the size of the dataset will have a positive effect on generalization ability. We verify these hypotheses by plotting the rewards obtained for varying degrees of the characteristics under investigation.

6 Experimental Results

6.1 Generalization Using Expert Data

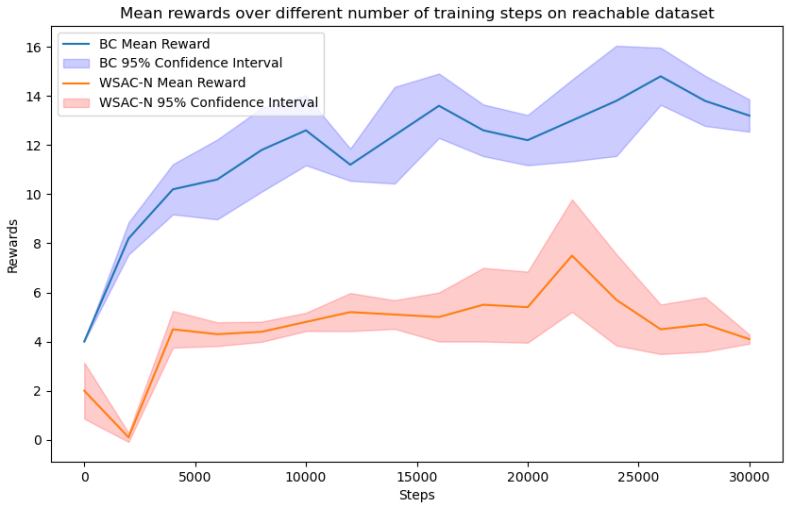

Figure 2: Mean rewards over different numbers of training steps, with **expert dataset** as training and tested on unreachable test set

Figure 2 illustrates the mean rewards over various training steps, using the expert dataset for training and testing on the unreachable test set. A 95% confidence interval is applied for accuracy. The graph shows that the confidence intervals do not overlap, indicating that WSAC-N performs worse than BC when trained on expert data. This result aligns with our expectations and supports previous research suggesting that BC can outperform offline RL algorithms when trained on expert trajectories (Levine et al., 2020; Ishita Mediratta et al., 2024). Similar rewards are observed when training on the reachable test set, as shown in Figure 6 in Appendix B.

6.2 Generalization Using Non-Expert Data

Figure 3 shows the mean rewards over various training steps, using suboptimal datasets for training and testing on the unreachable test set, with a 95% confidence interval. The non-overlapping confidence intervals indicate that WSAC-N performs worse when trained on suboptimal data, contrary to our expectations. This finding also contradicts previous research, which suggests that offline RL should outperform BC when trained on suboptimal data (Kumar et al., 2020; Kostrikov et al., 2021; Bhargava et al., 2024). However, it is important to note that the previous studies were conducted with

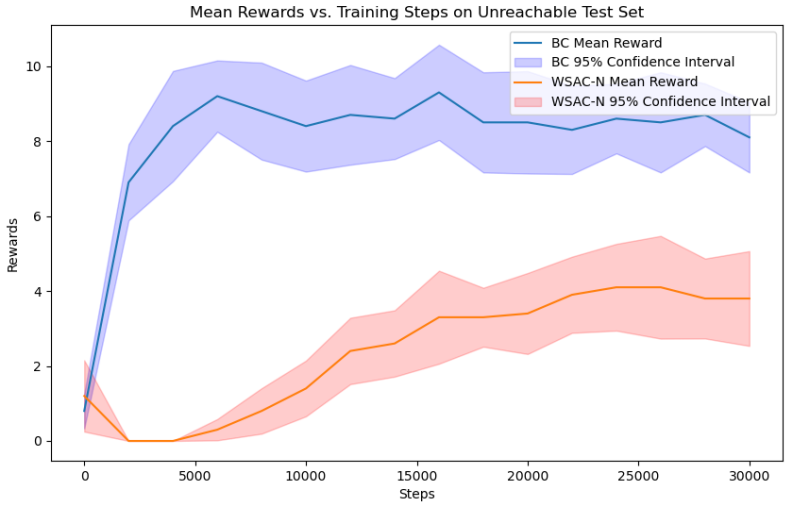

Figure 3: Mean rewards over various training steps, using **suboptimal datasets** for training and testing on the unreachable test set.

training and testing within the same environment. Our results align with research indicating that BC outperforms offline RL algorithms when training and testing in different environments (Ishita Mediratta et al., 2024). We see similar results when testing on the reachable test set, and when trained on the mixed expert-suboptimal and random datasets (see Figure 7 in Appendix B, Figures 9 and 8 in Appendix C, and Figures 10 and 11 in Appendix D respectively)

6.3 Effect of Dataset Quality on Generalization

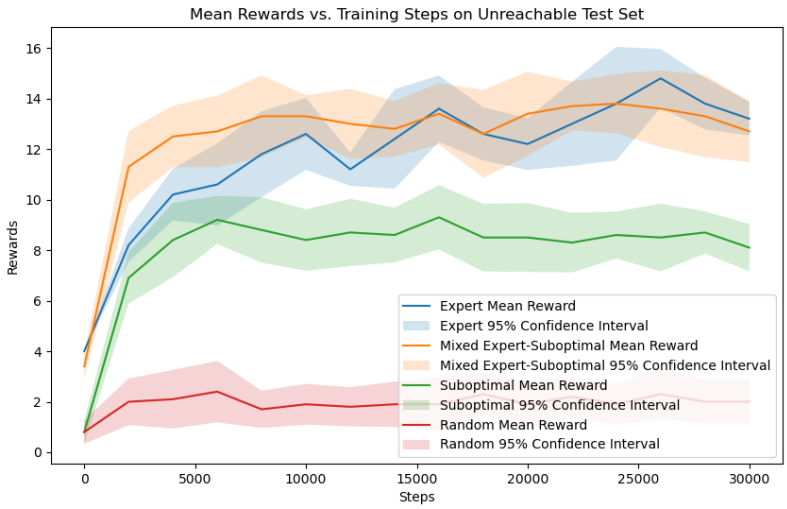

Figure 4: Mean rewards of BC over various training steps, using **different qualities** of the trajectories for training, and testing on the unreachable test set.

Figure 4 shows the mean rewards of BC over various training steps, using different qualities of datasets for training, and testing on the unreachable test set. In general, the better the quality of the dataset, the higher the mean rewards. This is in line with our expectations and previous research(Ishita Mediratta et al., 2024).

Unexpectedly, the confidence intervals for the expert and mixed expert-suboptimal datasets overlap. This may be due to the additional transitions in the mixed dataset, which includes 50% suboptimal actions. These suboptimal actions sometimes lead to longer paths to the goal, resulting in more transitions in the dataset $\mathcal{D}$.

A similar pattern is observed in the WSAC-N results, shown in Figure 12 in Appendix E.

6.4 Effect of Dataset Size on Generalization

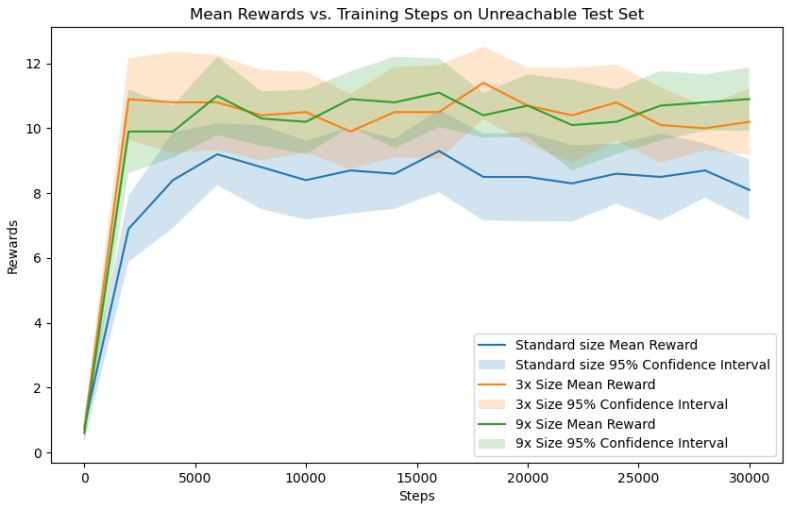

Figure 5: Mean rewards of BC over various training steps, using different **dataset sizes** for training, and testing on the unreachable test set.

Figure 5 shows the mean rewards of BC over various training steps using different dataset sizes for training and testing on the unreachable test set. We compare the standard size with datasets that are 3 times and 9 times larger. To control for diversity, all datasets are generated using the same set of seeds across different sizes. This means each dataset size group uses identical seeds as the other groups. The overlapping confidence intervals indicate that we cannot confidently determine which dataset size performs better. Notably, the mean reward for the dataset 9 times larger than the standard size is very close to that for the dataset 3 times larger. This suggests that, while keeping data diversity fixed, dataset size has a negligible impact on the generalization of our algorithms, which aligns with prior research (Ishita Mediratta et al., 2024).

7 Discussion

7.1 Bias

A common bias when comparing algorithms in reinforcement learning (RL) is putting more effort into tuning the hyperparameters of one algorithm over another, typically favoring the newly proposed algorithm over the baseline. To avoid this bias, we ensure that we test the same number of hyperparameter combinations for all algorithms being compared. However, this approach can slightly disadvantage algorithms with a larger number of hyperparameters. These more complex algorithms might not reach their full potential because many possible hyperparameter combinations remain untested. For instance, in our study, WSAC-N has more hyperparameters than BC, leading to more "untested" combinations for WSAC-N. This suggests that WSAC-N could potentially perform better than what our current results indicate. Future research could address this by ensuring that the number of tested hyperpa-

rameters is proportional to the total number of hyperparameters for each algorithm (i.e. by tuning x% of hyperparameters for each algorithm).

7.2 Effect of Seeds

During hyperparameter tuning, we found that random seed selection significantly influences algorithm performance, more so than many individual hyperparameters. For instance, in BC algorithms, seeds had a larger impact than any single tunable parameter. Similarly, in WSAC-N, seed choice almost matched the influence of the most critical hyperparameter, amount of critics.

To mitigate this, we averaged results across three randomly generated prime number seeds per hyperparameter combination. This approach helps stabilize performance, but it's worth noting that using different seeds could still lead to different optimal hyperparameters and results. Future research should explore multiple hyperparameter sets to better understand this sensitivity.

8 Responsible Research

Our research emphasizes reproducibility and transparency. To this end, we have open-sourced all relevant materials, including the code, datasets, procedures for generating datasets, rewards obtained during evaluation, and scripts for plotting graphs. Additionally, we provide detailed descriptions of our training methodologies and hyperparameter tuning processes in Appendix A.

From an ethical perspective, our work focuses on enhancing the generalization capabilities of offline RL. This approach is particularly valuable in scenarios where online RL poses significant risks, such as in healthcare settings (Liu et al., 2020).

Finally, we are committed to promoting responsible research practices by encouraging scrutiny and further development of our findings.

9 Future Work

Future research could explore several avenues to enhance the generalization capabilities of offline RL. One promising direction is combining offline RL with fine-tuning, as demonstrated by previous studies like Chebotar et al. (2021), which have shown its effectiveness as a pretraining method in practical applications. Another promising approach could involve experimenting with different ensemble techniques to improve model robustness. Additionally, adopting data augmentation strategies similar to those used in online RL, applying regularization techniques akin to classical machine learning methods such as classification, or leveraging algorithms designed to better capture underlying dataset representations like embeddings could potentially enhance generalization performance.

Furthermore, future studies could investigate different workflow optimizations to assess their impact on offline RL generalization. For instance, exploring cross-validation methods for hyperparameter tuning or evaluating various training stopping conditions could provide insights into improving model stability and performance across diverse environments.

An essential area for future investigation would be a detailed analysis of factors contributing to varying generalization outcomes among different algorithms. For example, understanding why WSAC-N, in our case, may underperform could involve examining whether its approach is overly conservative or if its policy struggles with selecting actions outside the training distribution. Such insights could lead to targeted improvements in algorithm design and training strategies aimed at enhancing overall generalization capabilities in offline RL settings.

10 Conclusion

This study explored the generalization capabilities of offline reinforcement learning (RL) algorithms, specifically examining WSAC-N, a hybrid of SAC-N and SUNRISE weights. Our goal was to address distributional shift and improve zero-shot generalization.

Our results showed that the baseline behavioral cloning (BC) algorithm consistently outperforms WASC-N, both with expert and suboptimal data. This highlights the challenges of achieving reliable generalization in offline RL, even with advanced uncertainty measures. While WSAC-N incorporates promising uncertainty measures, it requires further refinement to consistently outperform simpler methods like BC.

The quality of datasets significantly impacted generalization, with higher quality datasets leading to improved performance. However, increasing dataset size without enhancing diversity had negligible effect on performance. This indicates that the quality of the data is more critical than sheer quantity for generalization in offline RL.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning, 2020.

Joshua Albrecht, Abraham J. Fetterman, Bryden Fogelman, Ellie Kitanidis, Bartosz Wróblewski, Nicole Seo, Michael Rosenthal, Maksis Knutins, Zachary Polizzi, James B. Simon, and Kanjun Qiu. Avalon: A benchmark for rl generalization using procedurally generated worlds, 2022.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble, 2021.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 30–37. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: https://doi.org/10.1016/B978-1-55860-377-6.50013-X. https://www.sciencedirect.com/science/article/pii/B978155860377650013X.

Prajjwal Bhargava, Rohan Chitnis, Alborz Geramifard, Shagun Sodhani, and Amy Zhang. When should we prefer decision transformers for offline reinforcement learning?, 2024.

Wilka Carvalho, Andrew Lampinen, Kyriacos Nikiforou, Felix Hill, and Murray Shanahan. Feature-attending recurrent modules for generalization in reinforcement learning, 2023.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills, 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021.

Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and regularization in dqn, 2020.

Kevin Frans and Phillip Isola. Powderworld: A platform for understanding generalization via rich task distributions, 2023.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation, 2021.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck, 2019.

Qingfei You Ishita Mediratta, Minqi Jiang, and Roberta Raileanu. The generalization gap in offline reinforcement learning, 2024.

Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning, 2019.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, January 2023. ISSN 1076-9757. doi: 10.1613/jair.1.14174. http://dx.doi.org/10.1613/jair.1.14174.

Byungchan Ko and Jungseul Ok. Efficient scheduling of data augmentation for deep reinforcement learning, 2023.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment, 2020.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data, 2020.

Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning, 2020.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning, 2021.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

Siqi Liu, Kay See, Kee Ngiam, Leo Celi, Xingzhi Sun, and Mengling Feng. Reinforcement learning for clinical decision support in critical care: A comprehensive review (preprint). *Journal of Medical Internet Research*, 22, 02 2020. doi: 10.2196/18477.

Bogdan Mazoure, Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Improving zero-shot generalization in offline reinforcement learning using generalized similarity functions, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. https://api.semanticscholar.org/CorpusID:205242740.

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions, 2016.

Rob Fergus Roberta Raileanu. Decoupling value and policy for generalization in reinforcement learning, 2021.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research, 2021.

Bharat Singh, Rajesh Kumar, and Vinay Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55, 02 2022. doi: 10.1007/s10462-021-09997-9.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning, 2019.

Istvan Szita and András Lorincz. The many faces of optimism: A unifying approach. pages 1048–1055, 01 2008. doi: 10.1145/1390156.1390288.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards, 2021.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization, 2020.

Max Weltevrede, Matthijs T. J. Spaan, and Wendelin Böhmer. The role of diverse replay for generalisation in reinforcement learning, 2023.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021.

Chang Ye, Ahmed Khalifa, Philip Bontrager, and Julian Togelius. Rotation, translation, and cropping for zero-shot generalization, 2020.

Zhecheng Yuan, Sizhe Yang, Pu Hua, Can Chang, Kaizhe Hu, and Huazhe Xu. Rl-vigen: A reinforcement learning benchmark for visual generalization, 2023.

Ziqi Zhao, Zhaochun Ren, Liu Yang, Fajie Yuan, Pengjie Ren, Zhumin Chen, jun Ma, and Xin Xin. Offline trajectory generalization for offline reinforcement learning, 2024.

A Training and Hyperparameter Tuning Details

We tune the hyperparameters offline by selecting a set of hyperparameter-value pairs for each algorithm. It is crucial that the number of hyperparameters to be tuned is the same for all algorithms to avoid introducing bias. Unequal hyperparameter tuning effort can lead to incorrect conclusions about the comparative performance of the algorithms.

We conduct hyperparameter tuning using grid search with 3 seeds[4] for each combination. If optimal hyperparameters are found at the boundary of the set, we expand the hyperparameter set and repeat the tuning process for all algorithms to maintain consistency.

We test the hyperparameter combinations on a validation set to prevent overfitting on the training set and to avoid using the test set, which is intended for final evaluation. Using the test set for hyperparameter tuning would be inappropriate, as it would invalidate the test set as an independent measure of performance.

Our validation set comprises 10 out of the 40 paths to the target in the training set. With 3 seeds per combination, this results in 3 validation sets per hyperparameter combination, providing a form of cross-validation.

We train the models with $\mathcal{N}$ random seeds on $\mathcal{N}$ datasets for each dataset category[5]. Note that we train on the full dataset, which includes the previously created validation sets. We freeze the training every 2000 steps to evaluate the current algorithm on the reachable and unreachable test set. We store the rewards obtained during each evaluation so we can easily increase the amount $\mathcal{N}$. We then plot the performance of the algorithms over training steps with a 95% confidence interval. We increment $\mathcal{N}$ until the 95% confidence intervals do not overlap, ensuring that we can confidently determine which algorithm generalizes better on the test sets for every category.

B Results with Reachable Test Set

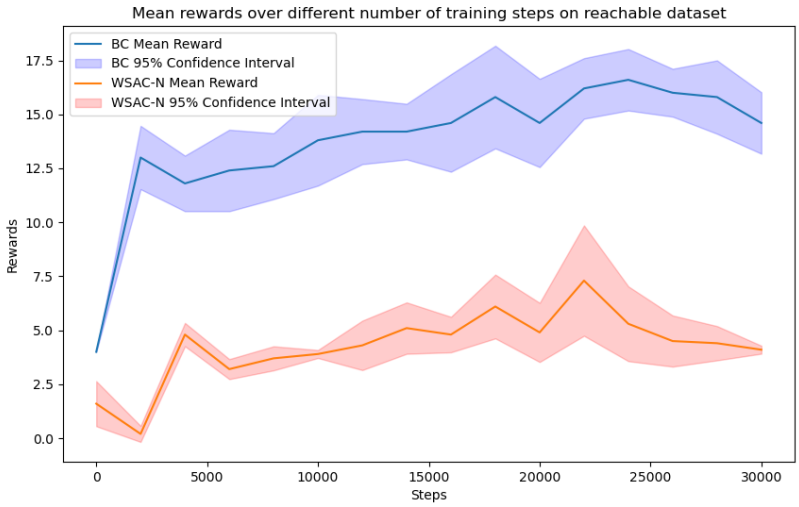

Figure 6: Mean reward with expert dataset as training over different numbers of training steps

[4]All seeds are generated using a random prime number generator.
[5]Note that the expert dataset category only has 1 possible dataset because the optimal policy is deterministic

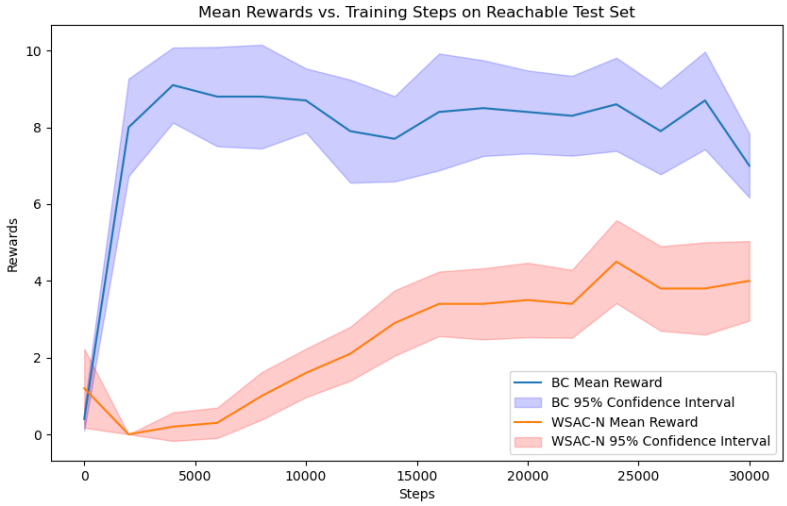

Figure 7: Mean reward with suboptimal datasets as training over different numbers of training steps

C Results with Mixed Expert-Suboptimal as Training Sets

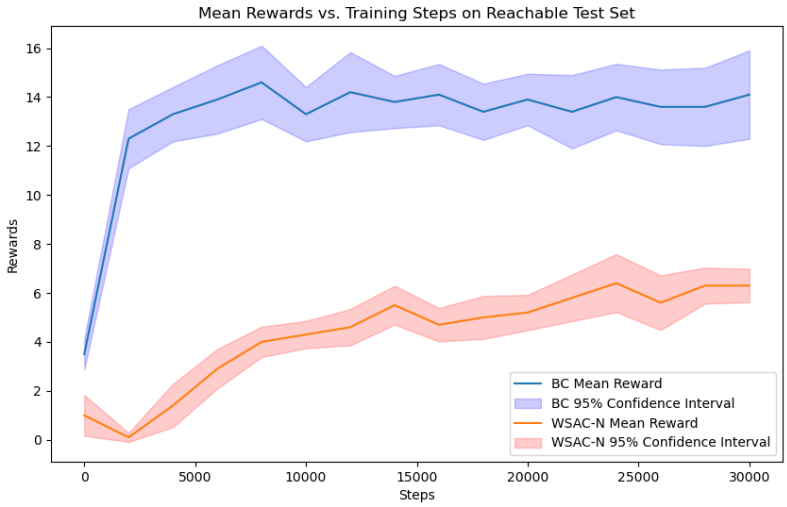

Figure 8: Mean reward with mixed expert-suboptimal datasets as training over different numbers of training steps

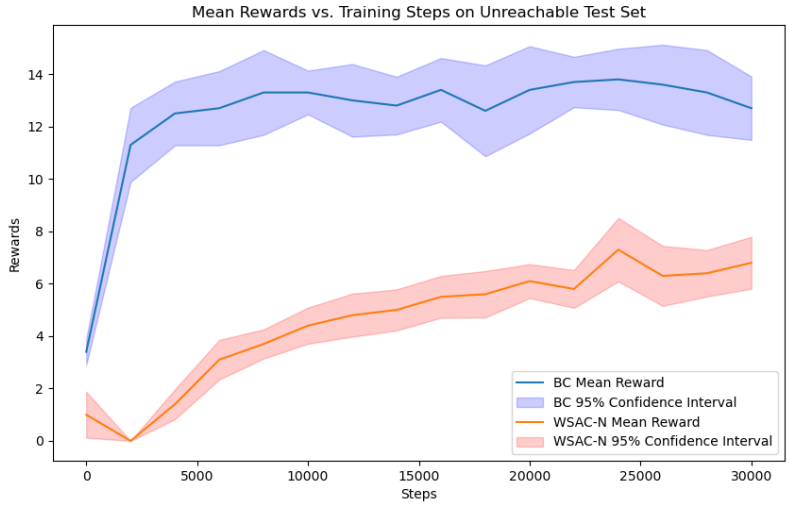

Figure 9: Mean reward with mixed expert-suboptimal datasets as training over different numbers of training steps

D Results with Random Datasets as Training Sets

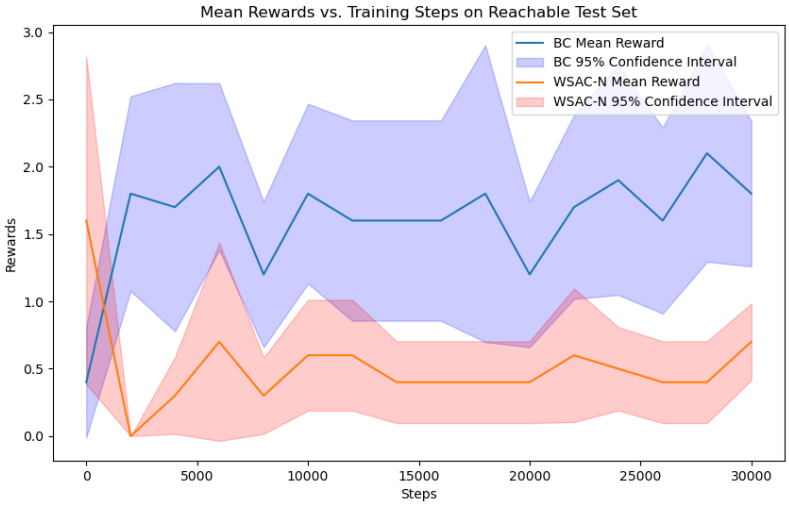

Figure 10: Mean reward with random datasets as training over different numbers of training steps

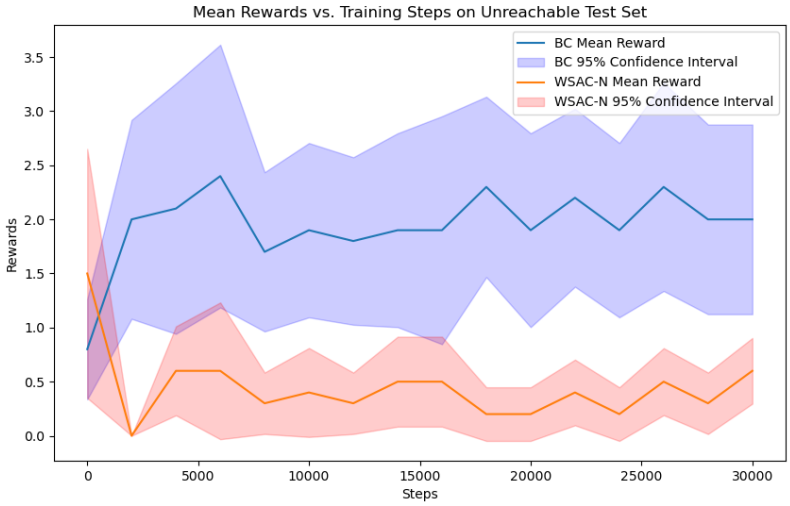

Figure 11: Mean reward with random datasets as training over different numbers of training steps

E Quality of Datasets on Generalization

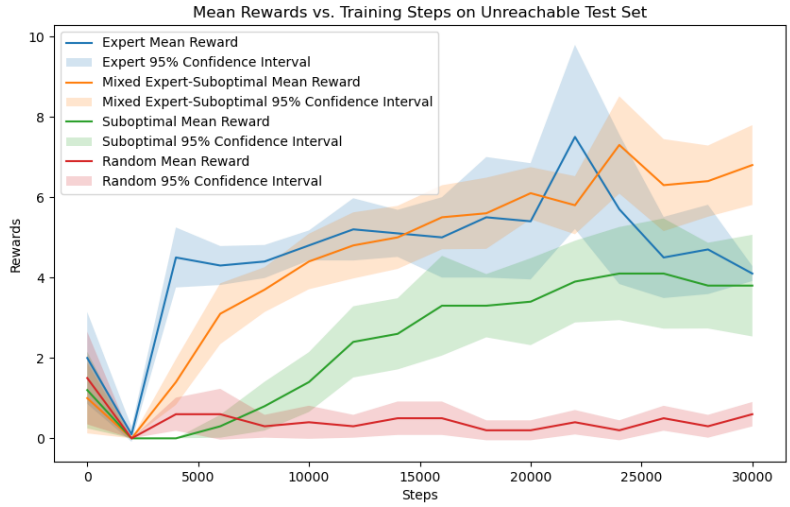

Figure 12: Mean rewards of WSAC-N over various training steps, using different qualities of datasets for training, and testing on the unreachable test set.