Offline Reinforcement Learning

Jannik Rehberger
Karlsruher Insititut für Technologie (KIT)
Karlsruhe, Germany
uudsi@student.kit.edu

Karam Daaboul
Karlsruher Institute of Technology (KIT)
Karlsruhe, Germany
daaboul@kit.edu

I. Abstract

This paper investigates Offline Reinforcement Learning (Offline RL), a methodology that enables the development of robust decision-making algorithms through the use of extensive datasets, without the need for further real-time interaction with the environment. This characteristic is especially beneficial in contexts where real-time interactions are either unethical or economically unfeasible, such as in autonomous driving, healthcare, and robotics. The paper elucidates the core principles of Offline RL, contrasts it with traditional Reinforcement Learning approaches, and discusses various algorithms and benchmarks. A critical assessment concludes the paper, highlighting Offline RL's vast potential and the practical challenges it faces in real-world applications.

II. Introduction

In recent years, machine learning has made significant advancements across various domains. Wherever large volumes of data are generated, machine learning can be employed to optimize processes. Reinforcement Learning (RL) is a subset of machine learning focused on developing optimal strategies for decision sequences to achieve the best long-term outcomes. It is important to note that some decisions might be highly effective in the short term but less so in the long run. Traditional RL algorithms continuously explore their environment, thereby discovering strategies that yield the best results [1]. Remarkable progress has been made in this field, with algorithms now capable of defeating world champions in complex strategy games like Starcraft [2]. However, these methods have two major drawbacks: first, they require data to be collected from scratch, which can necessitate numerous iterations for more complex tasks; second, they cannot benefit from pre-existing data as they do not incorporate it into their strategies. The concept behind Offline RL is that the agent no longer needs to interact with the environment but can develop its strategy based solely on previously collected data [3]. This data might come from machines performing similar tasks or from observing human agents. In many fields, a vast amount of data has already been accumulated, making it possible to leverage for Offline RL. While these methods have achieved promising results in simpler tasks such as Atari games, chess, or basic robotic tasks, they still face limitations in more complex real-world applications [3]. The reasons why Offline RL currently does not match the performance of Online RL methods will be discussed in detail in Chapter V.

III. Why Offline RL

Offline RL is particularly valuable in situations where interaction is either not possible or prohibitively expensive. This is especially relevant in fields such as autonomous driving, robotics, and healthcare, where errors could incur unacceptable costs. Thus, a strategy must be viable before it is deployed [4]. Offline RL promises to derive highly effective decision strategies solely from pre-collected data. Moreover, it addresses a significant challenge faced by Online RL: while Online RL algorithms may require millions of iterations to learn complex tasks, Offline RL leverages existing large datasets, thereby saving computational time and resources.

The datasets used do not necessarily need to be perfectly tailored to the current task, which minimizes or even eliminates the need for additional data collection [3]. The following sections will discuss specific applications where Offline RL can be effectively utilized, shedding light on the factors that must be considered for Offline RL methods to perform well in real-world settings.

A. Decision Making in Healthcare

An exemplary scenario for Offline RL in healthcare is the patient diagnosis and treatment process [5]. This can be modeled as a Markov Decision Process (MDP), where actions correspond to various treatment and testing methods, and states represent patient symptoms and test results. Traditional RL methods can be perilous in this context because experimenting with new approaches could cause significant harm to patients. Therefore, using historical data to determine optimal treatments is more appropriate in this scenario. The data would consist of real patient cases where the actions were chosen by their physicians.

In the healthcare sector, Offline RL encounters distinct challenges. The datasets often disproportionately feature cases from severe disease trajectories because more intense treatments are administered to critically ill patients, who also exhibit higher mortality rates. This imbalance can mislead an algorithm to erroneously conclude that treatments increase mortality, simply because the data under-represents mildly ill patients who receive fewer treatments and consequently are less frequently recorded.

B. Tasks in Robotics

In robotics, Online RL methods are generally well-suited for many applications [6]. However, there are scenarios where the

goal is to learn strategies for a series of individual actions that are effective across multiple similar environments and situations, such as the various process steps required to prepare different dishes. For such cases, an extensive dataset is necessary not only to master a single action but also to generalize these actions effectively. In this context, a database containing every action a robot has ever performed would be beneficial. This allows some actions to be executed without prior data collection if the action is a combination of previously learned actions (for example, the preparation of a soup containing onions and carrots can be learned if the robot has previously learned how to make a soup with onions and meat and another with cucumbers and carrots) [3]. Thus, Offline RL can be effectively applied to tasks involving many individual subtasks.

C. Interactive Dialogue Strategies

Dialogues can be viewed as a sequential decision-making process, modelable as a Markov Decision Process (MDP). This is particularly true for dialogues with specific objectives, such as customer service chatbots. Since the goal is to facilitate successful interactions with humans, data collection inherently involves human interactions, which can be very time-consuming when developing an effective bot. This effort can be significantly reduced by utilizing historical interactions. Such data are now collected during all interactions intended to replace human resources with bots, providing a rich resource for training and refining automated dialogue systems [3].

IV. Background

A. Reinforcement Learning

Reinforcement Learning (RL) fundamentally involves learning to make optimal decisions by an agent that receives a reward after each action, as detailed by [1]. This reward can be positive or negative. RL is an active process where the agent interacts with its environment by performing actions denoted by a, with the environment initially in a state s. When an action is executed, the environment transitions to a new state, and the agent receives a reward $r(s, a)$ dependent on both the state and the chosen action. The agent's goal is to develop a policy $\pi(a \mid s)$ that maximizes expected cumulative rewards. This policy can be either deterministic or stochastic. Importantly, rewards are unknown before the action is taken and are only received afterward.

RL is typically divided into three categories: Online RL, Off-Policy RL, and Offline RL. In Online RL, the classical form of RL, an agent interacts with its environment and updates its strategy based on the rewards it receives. The success of Online RL depends heavily on continuous, repetitive interactions with the environment, where the agent has access only to data collected through its own interactions. Off-Policy RL algorithms can also utilize previously collected data stored in a buffer D. This additional information enables the algorithm to improve more quickly and efficiently.

The RL algorithm's objective is to maximize the sum of expected rewards over time, defined as:

$$\max_{\pi} \sum_{t=1}^{T} E_{s_t, a_t \sim \pi}[r(s_t, a_t)] \tag{1}$$

This equation aims to maximize the cumulative expected reward over a specified period.

The Q-function $Q^{\pi}(s, a)$ is defined as the expected cumulative reward of taking action a in state s and following policy π thereafter:

$$Q^{\pi}(s_t, a_t) = \sum_{t'=t}^{T} E_{s_{t'}, a_{t'} \sim \pi}[r(s_{t'}, a_{t'}) \mid s_t, a_t] \tag{2}$$

This equation describes the Q-values for the state and action at time t under the strategy π [3]. The iterative process of executing an action, receiving a reward, and adjusting the strategy continues until the strategy is considered sufficiently effective.

B. Offline Reinforcement Learning

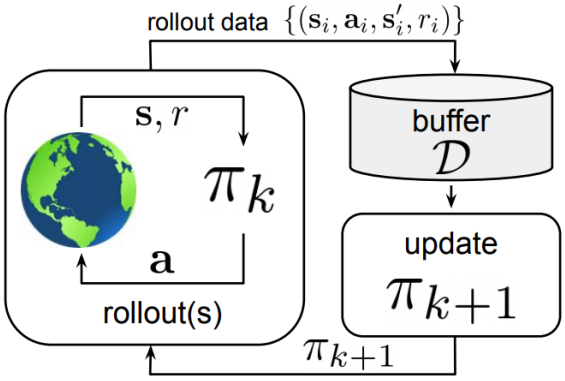
Offline Reinforcement Learning (Offline RL), sometimes referred to in the literature as "batch RL" or "fully off-policy RL," allows the agent to develop its strategy solely from an existing dataset, without further interaction with the environment. The key idea is to derive a better strategy π using an existing dataset and an associated behavior policy π_{β}. This dataset might include data from a robot that has previously solved a similar task or from a person who has driven a car. The datasets should be as diverse as possible to enable the agent to derive an optimal strategy for its task based solely on these existing data, without additional environmental interaction.

Theoretically, any Off-Policy RL algorithm can be adapted for use as an Offline RL algorithm. For example, Q-Learning can be applied without additional exploration of the environment. However, this approach is not straightforwardly effective due to several challenges such as distributional shift and the need for careful handling of the data to avoid biases.

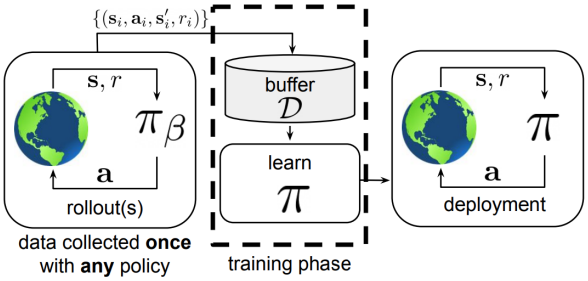
Offline RL proceeds in three main steps: First, data is collected to generate a behavior policy π_{β} from real-world data. Next, a new strategy π is learned and trained over many iterations using the collected static dataset. Finally, the strategy is deployed without further post-hoc adjustments. By leveraging these steps, Offline RL aims to overcome the limitations of needing extensive environmental interactions, making it particularly valuable in scenarios where such interactions are impractical or costly.

V. Challenges in Offline RL

Current Offline RL methods have yet to meet the expectations placed upon them. Several factors contribute to these challenges, some of which are more apparent than others. One of the more obvious issues is the lack of further exploration due to the static nature of the dataset D. This limitation makes it impossible to cover all possible action spaces, which can result in missing out on actions that could generate very high rewards. Currently, there is no solution to this problem, so



(a) Off-Policy RL: In addition to classical (Online) RL, a buffer D is introduced here, containing $\pi_0 - \pi_k$. With this additional information, a better strategy can be found more quickly. [3]



(b) Offline RL: Offline RL proceeds in three steps: 1. Generation of a Behaviour Policy π_β from real-world data; 2. A new strategy π is learned and trained over many iterations using the collected static dataset; 3. The strategy is deployed without further post-hoc adjustments. [3]

Fig. 1: Comparative overview of Off-Policy and Offline RL strategies.

it is generally assumed that the dataset D sufficiently covers high-reward actions to facilitate learning [3].

Another problem arises if the behavior policy π_β used to generate the dataset does not satisfy the Markov property [3]. This typically happens when data is collected by humans, who make decisions based not only on the current state but also on past events, thereby violating the Markov property. Since this property is fundamental for most Offline RL methods, it may be impossible to find an optimal strategy using such a dataset.

A less obvious issue is that Offline RL fundamentally relies on answering counterfactual questions, essentially hypothesizing about what might have happened under different circumstances [3]. This is crucial because the goal is not merely to replicate the behavior patterns in the dataset but to achieve better results. Therefore, the agent must perform actions that differ from those in the dataset. This leads to one of the core problems of Offline RL: distributional shift (DS). When the algorithm takes actions that are not represented in the collected data, its errors can increase, either positively or negatively. Because the algorithm aims to maximize Q-values, it may identify very high Q-values outside the dataset. However, since the quality of these actions cannot be evaluated due

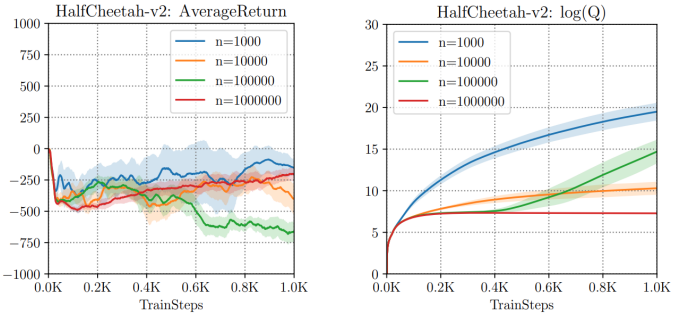


Fig. 2: The left graph shows the actual performance of the algorithm, while the right graph shows the estimated performance by the algorithm. The actual performance is very poor, whereas the Q-values are very high. [3]

to their absence from the dataset, this can lead to significant overestimation. Consequently, the algorithm diverges further from the dataset, believing it will perform well, but actually failing in reality [7].

This discrepancy is illustrated in Figure 2. It is important to note that the y-axis is on a logarithmic scale, indicating that the algorithm significantly overestimates its performance while it is actually extremely poor. One might assume that the underlying data is insufficient and that no better results can be achieved with this data. However, this hypothesis can be dismissed, as Behavioral Cloning (BC) can produce good results. BC is a method where human cognitive skills are captured and then replicated by a program [8]. In our case, BC replicates the best actions from the dataset, resulting in significantly better outcomes than with a standard Off-Policy RL algorithm.

Furthermore, the number of iterations is not very relevant to the algorithm's performance, indicating that this is not a typical overfitting problem that can be solved with more or diversified data. Therefore, applying Off-Policy methods to Offline environments is not straightforward. The following chapter will introduce methods that address the DS problem in various ways.

VI. Methods

There are several algorithms that implement the fundamental concept of Offline RL. These methods can yield varying outputs depending on the task at hand. In the following sections, we will discuss a few of these algorithms. It is important to note that due to the large number of existing methods, this paper cannot cover all of them. Additionally, Offline RL is a rapidly evolving field, and new algorithms may soon surpass the current state of the art. Nevertheless, we will now introduce some of the existing algorithms.

A. Policy Constraint

Policy Constraint methods are among the first and most widely used techniques in Offline RL. The main idea behind these methods is to limit the distributional shift by constraining

the agent's policy. The aim is to keep the learned policy π close to the behavior policy π_β. By doing so, the exploration of the algorithm is restricted, resulting in fewer out-of-distribution (OOD) states and consequently less deviation in Q-values. This is achieved by adding a constraint to the maximization condition of the Q-function, ensuring that the divergence between the two distributions π and π_β does not exceed a predefined threshold:

$$\pi_\theta = \arg\max_\pi \mathbb{E}_{s\sim D, a\sim\pi(a|s)}[Q(s,a)] \quad \text{s.t.} \quad D(\pi(a \mid s), \pi_\beta(a \mid s)) \leq \epsilon \tag{3}$$

Thus, the unconstrained optimization problem becomes a constrained one. The choice of ϵ is crucial for performance; if ϵ is too small, the policy cannot deviate sufficiently from the behavior policy to improve, although this keeps the distributional shift minimal. Different methods in this family vary in how they define the divergence between the two distributions and how they implement the constraint. One common approach is to use Kullback-Leibler (KL) divergence or other forms of f-divergence to measure the distance between the policies.

A practical implementation of a Policy Constraint method is the Bootstrapping Error Accumulation Reduction (BEAR) algorithm [7]. Bootstrapping error is a source of instability in current methods, arising from bootstrapping actions that lie outside the dataset. BEAR addresses this by using support constraints, where the divergence $D(\pi, \pi_\beta)$ is defined based on the support of the behavior policy. A popular metric for this is Maximum Mean Discrepancy (MMD):

$$D(\pi_\theta, \pi_\beta) = \text{MMD}(\pi_\theta, \pi_\beta)$$

By keeping the learned policy close to the behavior policy to manage divergence, Policy Constraint methods aim to stabilize the learning process. This makes them particularly valuable in scenarios where extensive environmental interactions are impractical or costly.

B. Value-Regularization

One of the main disadvantages of Policy Constraint methods is the necessity of estimating the behavior policy π_β. If the behavior policy is incorrectly estimated, for example, when fitting a unimodal policy to multimodal data, the performance of policy constraint methods can deteriorate dramatically. Additionally, these methods tend to be overly conservative. When we know that certain states have all actions with zero reward, constraining the policy there is unnecessary and can lead to suboptimal performance.

To address these issues, Value-Regularization methods focus on directly tackling the problem of overestimated Q-values. Instead of constraining the policy, these methods constrain the Q-function. A prominent example of this approach is Conservative Q-Learning (CQL) [9]. CQL explicitly evaluates unknown actions more conservatively by assigning them a lower expected value, thereby preventing overestimation of the Q-values. This ensures that the expected value of a policy involving unknown actions remains low.

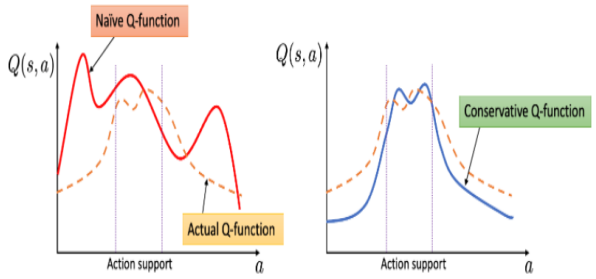


Fig. 3: Left: The naive Q-function is significantly overestimated outside the action support. Right: The conservative Q-function is consistently lower than the actual Q-function outside the action support. [9]

Value-Regularization can be implemented in two main ways. The first method involves incorporating a penalty term into the Q-function:

$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a'\sim\pi_\theta(a'|s')}[Q(s',a')] - \alpha D(\pi_\theta, \pi_\beta) \tag{4}$$

However, this method often results in an overly conservative Q-function. A more practical approach is to modify the objective function to first identify high Q-values and then reduce them:

$$\hat{Q}^\pi = \arg\min_Q \max_\mu \alpha \mathbb{E}_{s\sim D, a\sim\mu(a|s)}[Q(s,a)] + \mathbb{E}_{(s,a,s')\sim D}\left[(Q(s,a) - (r(s,a) + \gamma \mathbb{E}_\pi[Q(s',a')]))^2\right] \tag{5}$$

This formulation consists of two parts: the first part minimizes the Bellman error ($\min_Q$), and the second part finds and reduces high Q-values ($\max_\mu$). By applying this regularization to the Q-function, it ensures that the estimated Q-values are always less than or equal to the true Q-values, providing a lower bound for all Q-values.

By focusing on the Q-function and reducing overly high Q-values to realistic levels, Value-Regularization methods such as CQL provide a robust approach to managing distributional shift. This method has shown to be highly effective, as demonstrated in subsequent benchmark evaluations.

C. Offline Model-Based RL

Model-Based Reinforcement Learning (RL) algorithms primarily rely on the ability to estimate the transitions of a dataset $T(s_{t+1} \mid s_t, a_t)$ using a model $T_\psi(s_{t+1} \mid s_t, a_t)$. This model also includes a reward function $r_\psi(s_t, a_t)$. In principle, large and diverse datasets can be used for model building, making this approach promising for offline environments. In fact, Model-Based methods often outperform Model-Free approaches in terms of performance [4]. However, Model-Based methods also suffer from the effects of distributional shift (DS).

To address this issue, the strategy is modified by artificially reducing the reward for unexplored actions. This keeps the executed actions close to the model, ensuring that the model-based estimation of the Q-value remains accurate. It is crucial to strike the right balance between the potential improvement from exploring new actions and the risk of distributional shift when departing from the known action space. This is achieved by first finding a lower bound for the expected reward, which is then maximized in the second step.

There are additional challenges specific to Model-Based RL. Some Markov Decision Processes (MDPs) are difficult to model, especially those with long horizons and high-dimensional observations. Hybrid methods that combine Model-Based and Model-Free approaches can be promising in such cases. Furthermore, it remains an open question whether Model-Based methods can theoretically outperform Model-Free methods. With linear function approximation, both methods yield identical results, but it is unclear how they compare with nonlinear methods [3].

Currently, there are two distinct methods for implementing Model-Based Offline RL, both sharing the fundamental idea of keeping the policy close to the behavior policy.

Model-based Offline Policy Optimization (MOPO) [10] modifies the reward function by adding an intrinsic penalty $\lambda u(s,a)$ to the extrinsic reward $r(s,a)$. Here, $u(s,a)$ represents the estimated error, using the maximum standard deviation of the learned model rather than the mean to be more conservative. λ is a user-defined penalty coefficient, resulting in the new reward function $\widetilde{r}(s,a) = r(s,a) - \lambda u(s,a)$. The penalty increases with the uncertainty of the state, guiding the policy back into the model state space and preventing extreme deviations.

Model-Based Offline Reinforcement Learning (MOReL) [11] employs a similar approach by reducing rewards in uncertain states. However, unlike MOPO's softer penalization, MOReL uses a hard cutoff with an "absorbing state" when the deviations from the model exceed a certain threshold.

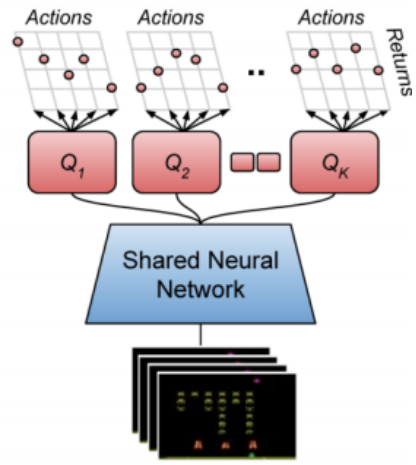
By managing the reward function and ensuring actions stay within a well-modeled space, these methods mitigate the risks of distributional shift, maintaining accurate Q-value estimations and stable policy performance.

D. Uncertainty-Based Methods

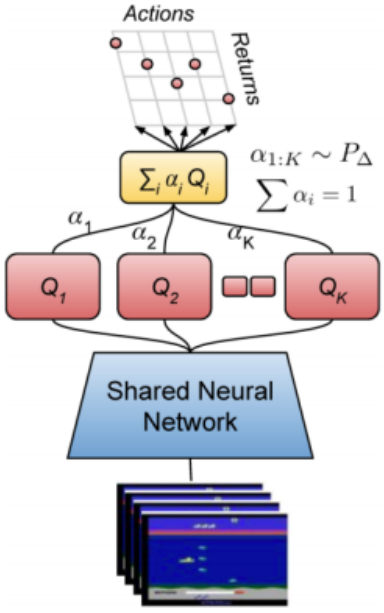
The three previously discussed classes share the common strategy of penalizing behavior that involves unpredictable actions by constraining the policy, either directly or indirectly through the Q-function or the reward function. Uncertainty-Based methods, on the other hand, aim to identify where the Q-function is inaccurate. This allows the policy to be less conservative in areas where the Q-function is known to be accurate, thereby retaining the "correct" high Q-values. The following two ensemble methods illustrate how these accurate Q-values can be obtained. Ensemble methods in machine learning combine different algorithms to achieve better results, and increasing the number of models typically improves performance.

Ensemble-DQN [12] is a straightforward extension of DQN. Instead of estimating the Q-value with a single Q-function as in DQN, multiple Q-values are estimated and then averaged. By incorporating multiple Q-values, better results are achieved compared to using a single Q-value [13].

Random Ensemble Mixture (REM) [12]applies this existing approach to offline environments. In REM, K different Q-values are combined into a single Q-function through a convex combination. REM combines various estimations of Q-values and uses this combination for robust training.



(a) Ensemble-DQN: Multiple Q-functions are estimated using a neural network. [12]



(b) Random Ensemble Mixture: Combination of multiple Q-values results in a new Q-function. [12]

Fig. 4: Ensemble methods to improve Q-value estimation in Uncertainty-Based methods.

These ensemble techniques allow Uncertainty-Based methods to better balance exploration and exploitation, ensuring

more accurate Q-value estimations while mitigating the effects of distributional shift.

VII. Benchmarks

The use of large datasets instead of online data collection holds significant potential. It is crucial to measure the effectiveness of various methods accurately, and benchmarks serve this purpose. However, existing online RL benchmarks are not well-suited for offline methods. Older benchmarks for Offline RL have typically focused on scenarios where datasets were generated by a previously trained policy. This approach tries to optimize the performance of the pre-defined policy rather than the current one. In real-world applications like autonomous driving, datasets are generated by numerous diverse policies, not by a single RL policy. This makes it unlikely that enough states and actions are covered to develop a robust new strategy, making it challenging to predict the performance of a new algorithm accurately.

A. D4RL

To address this issue, the "Datasets for Deep Data-Driven Reinforcement Learning" (D4RL) benchmark was introduced [4]. The goal of D4RL is to provide generalized tasks that are transferable to real-world scenarios, allowing new algorithms to be tested effectively. D4RL includes over 40 different tasks across seven domains.

1) Evaluation: To evaluate the performance of a method, D4RL introduces a scale from 0 to 100. This normalization allows for the comparison of different tasks. The performance score of an algorithm is calculated as follows:

$$\text{score} = \frac{\text{algorithm score} - \text{random score}}{\text{expert score} - \text{random score}}$$

A score of 0 corresponds to the average score achieved by an algorithm performing random actions (based on 100 trials). A score of 100 corresponds to the score of the best-performing algorithm for that task.

2) Challenges: One challenge in D4RL is the distribution of the previously collected data. Since it is impossible to cover all potential action spaces with pre-collected data, an algorithm might overfit to the available data. Additionally, the data might have been collected using a non-representative policy π_β, which can happen in real-world data generation. For example, human actions might contain unobservable patterns, such as decisions based on memory or experience. If the strategy used to generate the data cannot be determined (because it does not follow a clear policy), this adds another source of error.

Problems can also arise when data is generated without a specific task or goal. These datasets might come from videos of autonomous driving or from RL algorithms with different objectives. While these datasets can contain useful information, the solutions derived from them are often suboptimal. The "stitching" method (Figure 5) combines various suboptimal solutions into a better one. For instance, if a dataset contains a path from A to B and another from B to C, stitching can be used to find an optimal path from A to C.

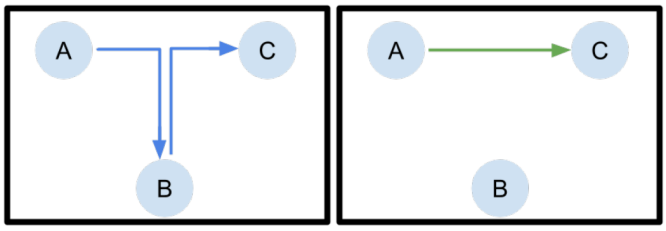


Fig. 5: Stitching: The shortest path from A to C is determined by combining the paths from A to B and B to C. [4]

Ultimately, simulated tasks are still simulations and not real-world tests. Simulation packages such as MuJoCo [14], Flow [15], and CARLA [16] have demonstrated a broad range of tasks for online algorithms, but there remains a significant gap between these simulations and real-world applications [4].

B. RL Unplugged

RL Unplugged [17] is another benchmark designed to evaluate and compare Offline RL algorithms. The contribution of RL Unplugged can be divided into tasks for different environmental states, corresponding datasets, and clear evaluation guidelines. This benchmark includes data from various domains, such as Atari games and the DM Control Suite [18]. RL Unplugged is continuously being developed, with new tasks and datasets being added to bridge the gap to real-world applications. While D4RL and RL Unplugged share many similarities, their method evaluation processes differ.

The evaluation process in RL Unplugged involves two main steps:

1) Online Evaluation: Initially, the algorithm is tested online. Although online interaction is not allowed during the execution of the algorithm, online evaluation helps in testing various hyperparameters to determine which ones yield the highest rewards. This step serves as an upper bound for the algorithm's performance, indicating its optimal potential in an online setting. However, this performance should be interpreted cautiously, as it only reflects the best possible online performance, which may not be achievable using offline methods.

2) Offline Policy Evaluation (OPE): After identifying the optimal hyperparameters, they are optimized offline. This step ensures that the algorithm's performance is assessed both online and offline. A new Offline RL algorithm aims to achieve perfect performance across all tasks, indicating its suitability for real-world applications [17].

RL Unplugged continues to evolve, providing new challenges and datasets to ensure comprehensive evaluation and to facilitate the development of robust Offline RL algorithms that can be effectively applied in real-world scenarios.

VIII. Conclusion

Offline RL presents a highly effective approach for utilizing reinforcement learning in tasks where interaction with the environment is infeasible due to safety or cost concerns.

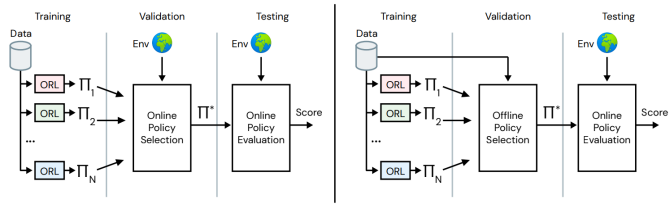


Fig. 6: Evaluation of RL-Unplugged. 1. Step: Online Testing; 2. Step: Offline Testing. [17]

Additionally, the vast amounts of data naturally generated in many tasks can serve as a solid foundation for developing effective decision-making strategies. However, the lack of active exploration introduces several challenges, and while current state-of-the-art methods offer promising solutions, they are not yet fully equipped to handle complex real-world scenarios.

To mitigate errors arising when a method operates outside the known dataset, various strategies have been proposed. These include constraining the policy to remain close to the original behavior policy (Policy Constraint), directly addressing the problem of overestimated Q-values by actively reducing them (Value-Regularization), estimating model uncertainty and modifying the reward function accordingly (Model-Based), and estimating the uncertainty of Q-values to avoid excessive conservatism in less uncertain areas (Uncertainty-Based). These approaches have already shown promising results in relatively simple tasks such as Atari games.

In the broader field of machine learning, rapid progress is driven not only by improved methodologies but also by the increasing size and diversity of datasets. For real-world tasks, the quality and comprehensiveness of datasets often play a crucial role, sometimes more so than the specific algorithm used. As datasets continue to grow in size and quality, they will provide a robust foundation for Offline RL, enabling its application to more complex real-world problems.

Looking ahead, we can expect continued advancements in both the datasets and the methods used in Offline RL. As more data becomes available across various domains, Offline RL stands on a promising foundation, with the potential to tackle increasingly complex tasks in real-world settings effectively.

References

[1] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

[2] Oriol Vinyals, Igor Babuschkin, Wojciech Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John Agapiou, Max Jaderberg, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019.

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

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

[5] Omer Gottesman, Fredrik Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, Jiayu Yao, Isaac Lage, Christopher Mosch, Li Wei H. Lehman, Matthieu Komorowski, Aldo Faisal, Leo Anthony Celi, David Sontag, and Finale Doshi-Velez. Evaluating reinforcement learning algorithms in observational health settings, 2018.

[6] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction second edition: 2014, 2015. 2014.

[7] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019.

[8] Caude Sammut. *Behavioral Cloning*, pages 93–97. Springer US, Boston, MA, 2010.

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

[10] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization, 2020.

[11] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning, 2020.

[12] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *CoRR*, abs/1907.04543, 2019.

[13] Oron Anschel, Nir Baram, and Nahum Shimkin. Deep reinforcement learning with averaged target DQN. *CoRR*, abs/1611.01929, 2016.

[14] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[15] Eugene Vinitsky, Aboudy Kreidieh, Luc Le Flem, Nishant Kheterpal, Kathy Jang, Cathy Wu, Fangyu Wu, Richard Liaw, Eric Liang, and Alexandre M. Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 399–409. PMLR, 29–31 Oct 2018.

[16] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: an open urban driving simulator. *CoRR*, abs/1711.03938, 2017.

[17] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad

Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando de Freitas. Rl unplugged: Benchmarks for offline reinforcement learning, 2020.

[18] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018.