Check for updates

ASME Journal of Computing and Information Science in Engineering
Online journal at:
https://asmedigitalcollection.asme.org/computingengineering

Nuria Nievas[1]
Eurecat,
Centre Tecnològic de Catalunya,
Unit of Applied Artificial Intelligence,
Lleida 25003, Spain
e-mail: nuria.nievas@eurecat.org

Leonardo Espinosa-Leal
Graduate School and Research,
Arcada University of Applied Sciences,
Helsinki 00560, Finland
e-mail: leonardo.espinosaleal@arcada.fi

Adela Pagès-Bernaus
Department of Economy and Business,
University of Lleida,
Lleida 25001, Spain;
Department of Animal Sciences,
AGROTECNIO-CERCA Center,
Lleida 25198, Spain
e-mail: adela.pages@udl.cat

Albert Abio
Eurecat,
Centre Tecnològic de Catalunya,
Unit of Applied Artificial Intelligence,
Barcelona 08005, Spain
e-mail: albert.abio@eurecat.org

Lluís Echeverria
Eurecat,
Centre Tecnològic de Catalunya,
Unit of Applied Artificial Intelligence,
Lleida 25003, Spain
e-mail: lluis.echeverria@eurecat.org

Francesc Bonada
Eurecat,
Centre Tecnològic de Catalunya,
Unit of Applied Artificial Intelligence,
Cerdanyola del Vallès 08290, Spain
e-mail: francesc.bonada@eurecat.org

Offline Reinforcement Learning for Adaptive Control in Manufacturing Processes: A Press Hardening Case Study

This paper explores the application of offline reinforcement learning in batch manufacturing, with a specific focus on press hardening processes. Offline reinforcement learning presents a viable alternative to traditional control and reinforcement learning methods, which often rely on impractical real-world interactions or complex simulations and iterative adjustments to bridge the gap between simulated and real-world environments. We demonstrate how offline reinforcement learning can improve control policies by leveraging existing data, thereby streamlining the training pipeline and reducing reliance on high-fidelity simulators. Our study evaluates the impact of varying data exploration rates by creating five datasets with exploration rates ranging from $\epsilon = 0$ to $\epsilon = 0.8$. Using the conservative Q-learning algorithm, we train and assess policies against both a dynamic baseline and a static industry-standard policy. The results indicate that while offline reinforcement learning effectively refines behavior policies and enhances supervised learning methods, its effectiveness is heavily dependent on the quality and exploratory nature of the initial behavior policy. [DOI: 10.1115/1.4066999]

Keywords: artificial intelligence, machine learning for engineering applications, manufacturing automation

1 Introduction

Controlling batch manufacturing processes poses significant challenges in optimizing performance while minimizing costs [1]. Despite technological advancements, many industries still rely on traditional control methods that use fixed parameters based on empirical knowledge to maintain quality and meet operational constraints. These static approaches often fall short in dynamic environments, where precise, real-time adjustments are crucial. Additionally, fixed parameters are generally inadequate for handling unforeseen events, unexpected scenarios, or system failures, leading to suboptimal performance.

Cycle-by-cycle adaptive control strategies, which adjust parameters in real-time, provide a promising approach by considering how each cycle affects future operations. These methods aim to optimize short-term performance while anticipating future impacts. They range from simple rule-based approaches and dynamic programming, which explore the state space thoroughly, to advanced artificial intelligence techniques. For discrete or low-dimensional problems, advanced techniques may not be necessary to achieve near-optimal control. However, for continuous variables and high-dimensional spaces, sophisticated algorithms are often required to manage the complexity effectively.

Despite their widespread use in dynamic industrial control, proportional-integral-derivative (PID) controllers are reactive,

[1]Corresponding author.

Manuscript received February 15, 2024; final manuscript received September 24, 2024; published online November 12, 2024. Assoc. Editor: Yan Wang.

Copyright © 2024 by ASME; reuse license CC-BY 4.0

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/jcise_25_1_011004.pdf by guest on 13 December 2024

addressing current errors without considering future states [2]. This can result in suboptimal performance, especially in processes that require foresight into future disturbances. Alternative methods, such as metaheuristics [3] and model predictive control [4], use predictive models to optimize control actions over extended periods. However, these methods often demand complex process models and substantial computational resources, making them impractical for real-time applications. The extensive computations and potential model inaccuracies limit their effectiveness in scenarios requiring precise long-term forecasting.

Reinforcement learning (RL) emerges as a dynamic control solution that not only addresses the multifaceted challenges encountered across various industries—such as uncertainty, operational variability, and high-dimensional problem spaces—but also reduces the computational demands typical of other methodologies, thanks to its pretraining on relevant data while effectively accounting for future consequences. Unlike traditional supervised learning (SL) paradigms, which treat data as independent, RL recognizes that data dependencies are shaped by past interactions [5]. In RL, current actions influence future states, so potential actions are evaluated based on their expected outcomes. This evaluation relies on a utility or reward function, which guides the learning process by quantifying the expected rewards for each action-state pair.

In the classical RL framework, an agent interacts with an environment, gathering data through its actions and refining its understanding through iterative trial and error [6]. Recent advancements have demonstrated the effectiveness of online RL solutions in various industrial applications [7–9]. However, real-world physical environments impose significant constraints, such as safety considerations and limited exploration opportunities, which are crucial for preventing undesirable outcomes and mitigating negative consequences. Additionally, the training speed is limited by the actual physical time required, complicating the learning process. Therefore, key challenges in applying RL to industrial systems include accelerating the learning process and conducting exploration within safety constraints [10].

To address these challenges and facilitate safe exploration of the solution space, process simulators have become a prevalent strategy in RL research [11–13]. These simulators range from established software tools to data-driven surrogate models. However, even well-calibrated simulation models inevitably introduce discrepancies, leading to performance issues in RL models. This discrepancy, known as the simulation-to-reality gap [14], remains a significant challenge. Bridging this gap is crucial because simulators, while fast and potentially accurate, may not fully capture the complexities of real-world environments. Various calibration techniques have been proposed to mitigate this issue, such as re-training RL models in real-world settings [15–17]. However, these methods are often resource-intensive, requiring precise simulators, extensive RL training, and subsequent fine-tuning with real-world data.

To enhance sample efficiency and reduce the need for extensive interactions with real-world environments, improve the performance and generalization capabilities of data-driven models, and accelerate the overall RL process, recent research has investigated the integration of physics-informed principles and domain knowledge into data-driven surrogate models [18,19].

In addition to physics-informed models, another promising approach to overcoming the limitations of simulation-based training and simplifying the training process is offline RL [20]. In offline RL, a fixed dataset of experiences—comprising states, actions, and rewards collected from prior interactions with the environment—is used for training. This dataset, obtained via a behavior policy, eliminates the need for continuous interaction during the training phase. For offline RL to surpass the behavior policy, the dataset must be diverse and exploratory, covering a wide range of states and actions to effectively capture the potential rewards of different actions [21]. Although recent studies have explored offline RL applications in industrial and manufacturing contexts [22–25], the field is still in its early stages and holds substantial potential for further development.

This paper explores the feasibility of using offline RL to streamline the training process for control models, aiming to simplify traditional methods by requiring only a single training phase. Unlike SL, which replicates behavior policies, offline RL refines these policies based on reward signals. This study evaluates the performance of offline RL with datasets generated from various behavior policies to assess whether it offers a more efficient and practical alternative to conventional RL approaches and to identify the conditions necessary for its successful implementation. Specifically, we investigate the application of offline RL in a press hardening case study to achieve adaptive control in a dynamic and stochastic batch manufacturing process. The contributions of this paper are as follows:

(1) Demonstration of adaptive control in a dynamic and stochastic batch manufacturing process using offline RL, with improvements compared to an SL approach.
(2) Comparison and analysis of the dataset requirements, specifically the behavior policy used for data collection, necessary for successful offline RL training.

While our research illustrates these concepts through a case study in the press hardening industry, our primary aim is to analyze the broader potential of RL. Our approach is designed to scale effectively, handling complex state and action spaces to address challenging objectives across various real-world manufacturing scenarios. The paper is structured as follows: Sec. 2 reviews offline RL, contrasting it with online RL and detailing key algorithms. Section 3 describes the case study used for analysis. Section 4 covers the methodology, including behavior policies and the offline RL algorithm. Section 5 presents the benchmarking results. Section 6 concludes with a summary of findings and recommendations for future research.

2 Offline Reinforcement Learning

An RL problem is formalized as a Markov decision process defined by a tuple of 5 [6]: a set of states and actions ($\mathcal{S}$, $\mathcal{A}$), a state transition probability distribution $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, which defines the probability of transitioning from one state to another given an action, a reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, which assigns a scalar reward to each state-action pair, and a discount factor $\gamma \in [0, 1]$, which balances the importance of immediate rewards versus future rewards. The agent's ultimate goal is to learn a function, called policy π, that maps states to actions, maximizing the expected cumulative future reward.

An RL model can learn from scratch without any specific initial knowledge of the domain or predefined logic defining any initial policy. However, it is crucial to define the action space, which refers to the set of possible actions the agent can take; the state space, which comprises the environmental variables necessary for decision-making, and the reward function, which guides the RL agent during its learning process. Value functions and Q-functions are functions that provide estimates of the expected cumulative future reward associated with being in a state s_t or taking an action a_t from a given state s_t. In this learning paradigm, complex modeling isn't required; instead, the RL agent learns through trial and error, interacting with its environment to understand the consequences of its actions. The environment can be accessible (either physically or through some digital representation) or have limited information.

Offline RL learns by utilizing a fixed dataset containing states, actions, and rewards. This dataset is collected prior to training and is limited in scope. The difference between the distribution of data collected before training and the distribution of states and actions learned from this data is known as the distribution shift [26]. A significant challenge in offline RL occurs when the training dataset lacks transitions that highlight regions of high reward. In such cases, identifying these rewarding regions becomes exceedingly difficult. Additionally, if the agent erroneously assumes that certain regions should yield high rewards (overestimation) without this information being supported by the data, correcting this misconception becomes a complex task. In contrast, online RL involves continuous

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024

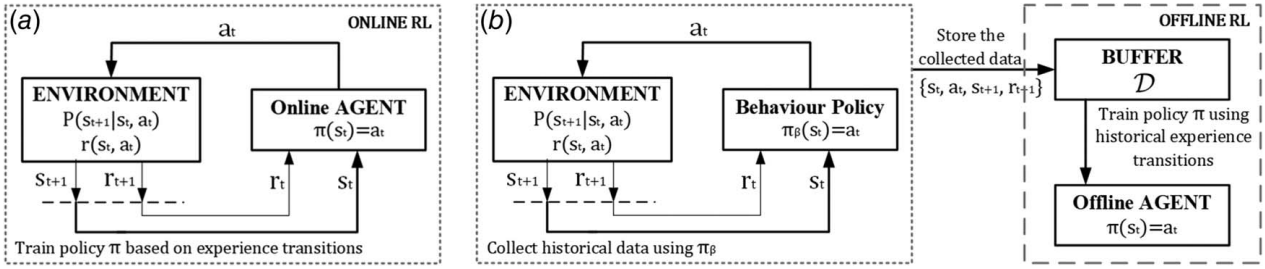

Fig. 1 Schematic representation of online and offline reinforcement learning methodologies: (*a*) online reinforcement learning scheme and (*b*) offline reinforcement learning scheme

interaction with the real environment, allowing erroneous beliefs to be corrected during the learning process. The schemes for both online and offline RL are illustrated in Fig. 1.

Several types of algorithms have been proposed to address the challenges of distribution shift and overestimation in offline RL. These algorithms are designed to limit visits to out-of-distribution (OOD) states or penalize such visits, introducing a certain degree of uncertainty or regularization terms when dealing with states that deviate significantly from the distribution of the predefined dataset. There are various types of algorithms based on the method used to address the distribution shift problem [21]: some focus on constraining the policy from deviating too far from the behavior policy, others introduce uncertainty into the Q-functions or reward function, some incorporate regularizers into the value function, and there are those that restrict training to the data provided in the dataset.

We present a taxonomy of offline RL methods classified based on whether they learn a model of the environment (model-based methods) or learn directly from data (model-free methods). Furthermore, within each of these types of methods, we have categorized algorithms based on the type of mechanism they employ to address the issues of distribution shift and overestimation. The simplified taxonomy is presented in Fig. 2. Detailed explanations of these methods are provided in the Appendix.

3 Use Case Description

To demonstrate the practical application and benefits of offline RL in manufacturing environments, we deploy it within the industrial press hardening process. The press hardening process is a metal forming technique employed in the manufacturing of lightweight parts with intricate shapes, offering superior mechanical properties, sustainability, and impact protection. This method involves the simultaneous shaping and transformation of metallurgical features of a metal sheet. Initially possessing poor mechanical characteristics, the sheet undergoes phase changes when heated to high temperatures (austenitization temperature), allowing it to be formed into complex shapes and then quenched to obtain improved mechanical properties [27]. During forming, the heated sheet, which initially has low strength, is pressed into the desired shape, while quenching involves rapid cooling to transform the sheet into strong and stable martensite [28].

Our study primarily focuses on controlling a batch press hardening manufacturing process. This process involves adjusting the production time for each part in the batch based on the initial temperature of the metal sheet after heating and the temperature of the die.

In this manufacturing scenario, sheets undergo a controlled process starting with insertion into a furnace for the crucial heating phase. Furnace temperatures are precisely maintained between 900°C and 930°C to ensure thorough austenitization, a fundamental step for subsequent processing stages. Upon heating, sheets are carefully transferred to a hydraulic press die, during which variable cooling introduces temperature fluctuations ranging from 680°C to 880°C upon arrival at the die.

At the press die, sheets undergo simultaneous forming and quenching operations aimed at achieving precise shaping and metallurgical transformations. This process simplifies operations by eliminating plastic deformation and friction.

The success of each cycle is evaluated based on the final part's temperature immediately after forming, with an efficiency benchmark set at 160°C to ensure effective heat exchange. The flat geometry of the part used in our study justifies our focus on evaluating temperature at a single point. Furthermore, acquiring temperature distribution in real-world manufacturing environments typically requires expensive tools such as thermographic cameras. Real-

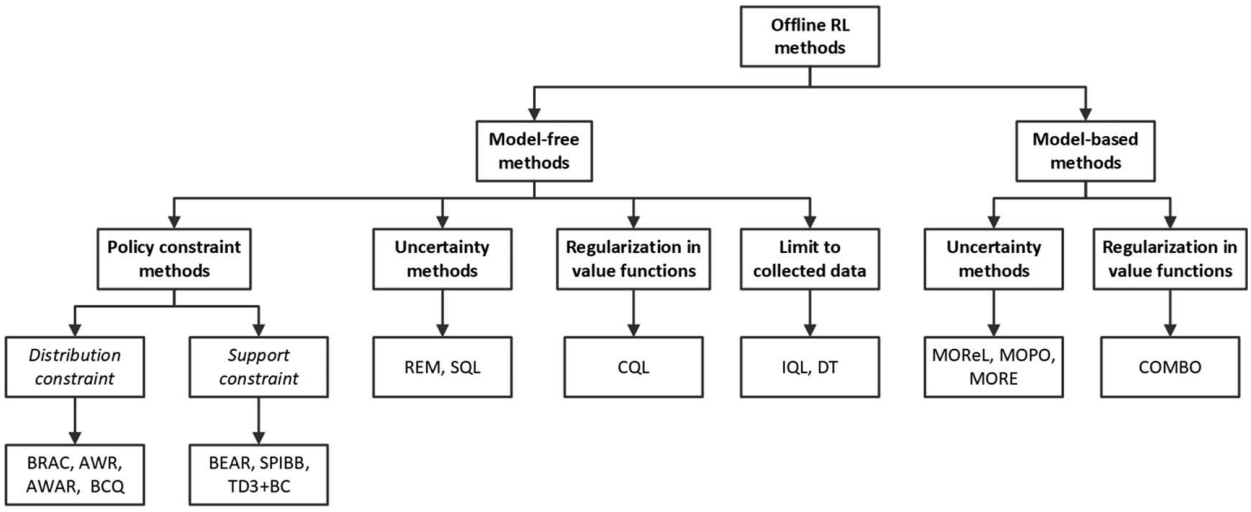

Fig. 2 Taxonomy of offline RL methods

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024

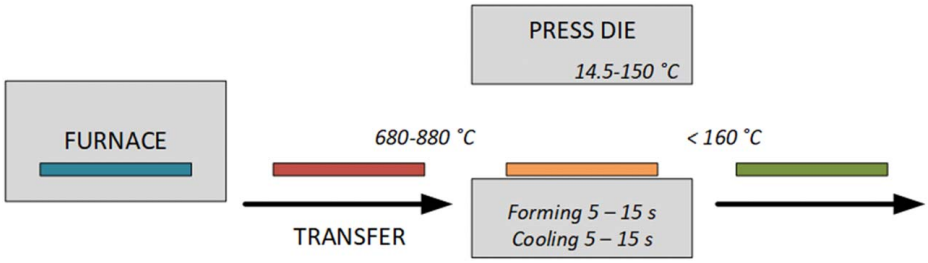

Fig. 3 Diagram illustrating the manufacturing press hardening process, depicting defined parameters from the case study: sheet and die temperature ranges, forming and cooling time ranges, and quality threshold

world processes often monitor only a few key points, and in our case, we utilized data from a specific sensing point in the actual pilot plant used to define our case study [29]. Ultimately, the ability to assess quality using a single variable depends on the complexity of the simulation model and the feasibility of acquiring comprehensive system data in the plant. Ideally, the martensite fraction should determine the quality of the hot stamping process, but this metric closely correlates with the cooling rate [30]. Temperature serves as a reliable indicator that the desired cooling rate has been achieved, closely approaching a martensite fraction of nearly 1, and is readily measurable in real-time scenarios.

Following forming, the die opens to extract the formed part, signaling the end of the cycle. The die then enters a cooling phase where water cooling channels dissipate heat from the die surface to prepare for subsequent cycles. The initial die temperature, initially set between 14.5°C and 23.5°C, adjusts during batch execution and typically peaks at 150°C in this operational setup.

Throughout the process, sheets are processed sequentially until the entire batch is completed. The manufacturing process considered in this study, along with the ranges of the variables under consideration, is depicted in Fig. 3. In the figure the part is shown at various temperature stages: the initial temperature before entering the furnace (not measured), the sheet temperature after heating, the temperature during the forming and quenching phase while the sheet is in the die (where it decreases), and the final part temperature, which must be below 160°C.

Our objective is to optimize the production time for each individual part, referred to as a cycle, to enhance the overall efficiency of the batch. However, this goal involves addressing two inherently opposing objectives:

(1) **Minimizing the total production time**: This involves optimizing the individual cycle times during batch production, where multiple parts are produced sequentially. We dynamically adjust two key parameters: the closed die time, during which heat exchange between the sheet and the die shapes the part and determines its final temperature, and the resting die time, when the die cools down for the subsequent cycle.

(2) **Minimizing the ratio of defective parts**: This aims to reduce the number of faulty parts produced during batch production. To minimize the defect rate in batch production, it is crucial to ensure the production of satisfactory parts. Maintaining the cooling rate above acceptable thresholds establishes a direct correlation between the hardness of the components and their final temperature [29]. Consequently, a part is deemed defective if the temperature exceeds a specified threshold after the forming and quenching step.

To achieve our objectives, we establish two key control variables: forming (or closing) time and cooling (or resting) time. We explore different values for forming time (5, 10, and 15 s) and cooling time (5, 10, and 15 s) to assess their respective impacts. The combination of these control variables defines the actions within the action space of the control problem.

The state space is characterized by the initial temperatures of both the sheet and die for each cycle. Each state transition within the problem formulation represents a cycle across the entire batch. The initial sheet temperature is influenced by stochastic factors and is not directly controllable. The initial die temperature for each new cycle is determined by the preceding cycle and the chosen cooling time for the die, ranging between 14.5°C and 150°C. At the beginning of the batch production, the die temperature is low, but it rapidly increases with the initial cycles.

The reward function generates a scalar signal after each cycle and consists of two penalties. The first penalty is associated with the selection of the cycle time, determined by the sum of the forming and cooling times. Its magnitude ranges from −30 to −10, depending on the combination of the three defined forming times and the three available cooling times. The second penalty, set at −10, 000, is imposed only if the part's temperature exceeds a defined threshold of 160°C, indicating the production of a defective part. Mathematically, the reward function $r(s_t, a_t)$ is formulated as shown in Eq. (1)

$$r(s_t, a_t) = -F_t - C_t - 10{,}000 \cdot \Delta T(s_t, a_t) \tag{1}$$

where

- F_t denotes the forming time associated with the action a_t in state s_t.
- C_t represents the cooling time associated with the action a_t in state s_t.
- $\Delta T(s_t, a_t)$ is an indicator function, defined as follows in Eq. (2):

$$\Delta T(s_t, a_t) = \begin{cases} 1 & \text{if } T_p > 160°\text{C} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where T_p is the final temperature of the part after the cycle.

Table 1 outlines the action space, state space, and reward function specified for the control problem addressed in this study. Additionally, the discount factor (γ) is set to 0.99.

Table 1 Summary of action space, state space, and reward function

Component	Description	Values
Action	Discrete cycle time adjustments (s)	$\{-5, -10, -15\} \times \{-5, -10, -15\}$
State	Initial sheet and die temperatures (°C)	$[680, 880] \times [14.5, 150]$
Reward	Time optimization and quality assessment	$\{-10, -15, -20, -25, -30\} \times \{0, -10, 000\}$

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024

4 Methodology

This study describes a methodology for generating behavior policies and applying an offline RL algorithm. Due to the limitations of accessing a real industrial plant, we utilized a simulator specifically designed for batch production in press hardening to conduct our analysis. This simulator closely mimics real-world conditions, enabling data collection across different policies and the subsequent validation of the trained agents. Importantly, while the simulator is used here to test the RL methodology, our approach is intended for real-world application without reliance on simulators for decision-making.

The simulator employs two surrogate models, as described in Ref. [31], to predict die and part temperatures after each cycle. The die temperature is incorporated into the state for the next cycle, while the part temperature is used to evaluate the part quality and detect defects. These models have been experimentally validated at the STaMP pilot plant at Eurecat Manresa [32]. Initial sheet temperatures are modeled using a uniform distribution ranging from 680°C to 880°C, covering a wide range of potential conditions. In practical applications, explicit modeling of the initial sheet temperature is unnecessary, as it would naturally be influenced by the stochastic nature of the environment.

4.1 Behavior Policies for Data Collection. To establish a dynamic expert policy as a baseline for learning, we initially trained an online RL approach using the deep Q-learning (DQN) algorithm [33]. DQN enhances the classical Q-learning algorithm [34] by utilizing artificial neural networks to approximate the Q-value function, thus effectively handling large state-action spaces more efficiently than traditional tabular methods. For this study, we employed the DQN implementation from the Stable Baselines3 library [35] in PYTHON. The parameters included a learning rate of 0.0003, a total of 2,000,000 interactions with the environment, a batch size of 64, and a neural network architecture consisting of two hidden layers, each with 64 units.

The exploration–exploitation tradeoff was managed using an epsilon-greedy decay strategy. Initially, the exploration rate (ε) was set to 1.0, indicating a high probability of taking random actions. This rate gradually decayed to a final value of 0.02 over the first 50% of the training duration. The soft update coefficient (τ) of 0.2 governed the rate at which the target network's parameters were updated toward the main network's parameters.

Figure 4 shows the progression of accumulated rewards over an episode during the training process, highlighting the trend of increasing and converging reward values. This accumulated reward over an episode is also referred to as the return. Evaluations were conducted at intervals of 100,000 interaction steps with the environment. Each data point represents the average return obtained from 100 episodes, with each episode consisting of a batch of 15 parts.

This dynamic expert policy served as a baseline for data collection for the offline RL training. The goal was not to achieve an optimal solution but to create a policy with room for improvement, allowing offline RL algorithms to refine it further. Consequently, efforts to optimize the policy were limited. Alternatively, predefined rules or domain-specific knowledge could also be utilized for this task. To explore the impact of different levels of exploration on offline RL performance, controlled randomness was systematically introduced into this dynamic policy. Starting with a robust policy, noise was added to simulate more exploratory behaviors. The randomness was quantified using an ε parameter, which determined the probability of selecting a random action instead of following the expert policy. Five behavior policies were defined with ε values of 0, 0.2, 0.4, 0.6, and 0.8. An $\varepsilon = 0$ indicated full exploitation with no exploration, assuming no physical or safety constraints in a real environment due to the initial policy's validation. Higher ε values introduced greater exploration, potentially leading to feasibility issues, with maximum randomness at 80% exploration and 20% exploitation.

A dataset was generated for each of the five behavior policies, which were then used to train offline RL policies. These trained policies, along with a dynamic expert policy, were compared against a static policy that used fixed timings regardless of environmental conditions, representing standard industry practice. The static policy was unsuitable as a behavior policy due to its lack of necessary exploration for effective offline RL training. However, it was used as an industry baseline to highlight the significant improvement of dynamic policies.

Figure 5 illustrates the pipeline for dataset collection and the evaluation of the offline RL approach. This framework aims to identify the most effective strategies. To capture a diverse range of actions and thoroughly explore the state space, controlled interventions in the manufacturing process are essential for data collection. Each policy is associated with a specific ε value, and all datasets are uniformly sized at 20,000 transitions. It is important to note that this study does not directly compare online and offline RL. However, the dynamic expert policy is trained using 2,000,000 interactions, whereas the offline RL models are trained on datasets of 20,000 transitions, which represents only 1% of the data collected for online training.

4.2 Offline Reinforcement Learning Algorithm. The algorithm employed for offline RL analysis in the press hardening case study is conservative Q-learning (CQL) [36]. The selection of CQL was driven by its demonstrated effectiveness in addressing overestimation biases in Q-value predictions, as well as its successful application across various domains [37,38]. CQL is particularly advantageous due to its versatility; it offers implementations for both continuous action spaces using actor-critic methods and discrete action spaces through Q-learning. This flexibility allows CQL to be effectively scaled and adapted to a wide range of problem formulations, enhancing its utility in diverse scenarios.

CQL utilizes a data buffer D containing past transitions (s, a, r, s'). Training is conducted over multiple iterations. In each

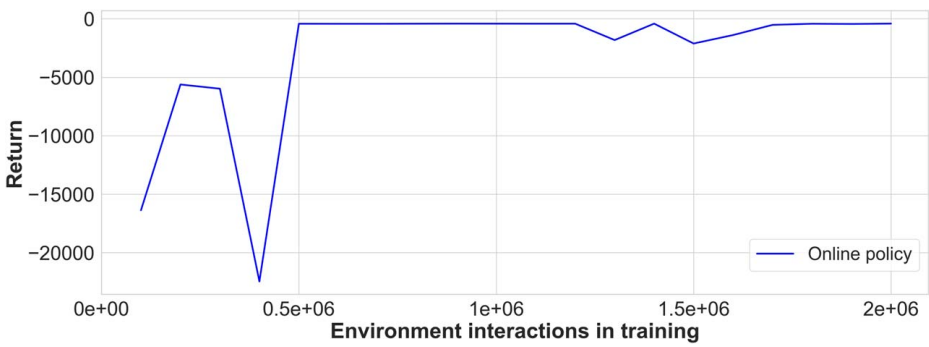

Fig. 4 Progression of returns during online training, evaluated every 100,000 interaction steps with the environment

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/jcise_25_1_011004.pdf by guest on 13 December 2024

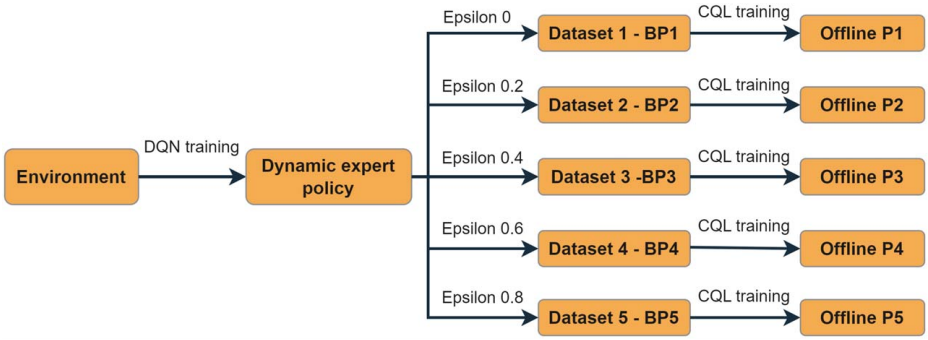

Fig. 5 Pipeline illustrating the methodology for implementing offline RL in batch press hardening production. This figure details the process of collecting datasets using various exploratory behavior policies and outlines the connection between each dataset and the corresponding trained offline RL policy. For example, "Offline P1" refers to offline policy 1, trained with dataset 1 collected using behavior policy 1, and so forth.

iteration, a sample is drawn from the buffer to update the Q-function using the gradient of the loss function. Despite using only historical data from the buffer, CQL indirectly learns about actions that are not present in the data by exploring the action space during training. This is achieved by regularizing the Q-function to minimize overestimation of values for unseen actions, employing a penalty based on KL-divergence and a conservative estimation against the reference behavior policy. Consequently, the algorithm gradually adjusts its Q-value estimation to accommodate actions that may arise in future environments, thereby enhancing its ability to generalize beyond the specific data stored in the buffer.

Fine-tuning the compensation factor α is critical for achieving a balance between the regularization imposed by the conservative penalty and the optimization of the Bellman error. Importantly, CQL can be implemented both as an actor-critic algorithm and as a Q-learning method. In the actor-critic paradigm, training involves updating both the policy and the Q-function, whereas in the Q-learning setting, only the Q-function undergoes training.

We selected the d3RLpy library for offline RL training for several reasons. It provides robust implementations of both discrete and continuous CQL, which contributes to its versatility. Designed specifically for offline reinforcement learning, d3RLpy encompasses a comprehensive range of algorithms and is supported by extensive documentation, detailed tutorials, and practical examples that greatly enhance both understanding and usability. Moreover, the library benefits from ongoing development, as demonstrated by recent updates and a growing user community. The discrete CQL implementation in d3RLpy adheres to the loss function outlined in Eq. (3), incorporating the loss from a DoubleDQN-based RL algorithm [39]. It is important to note that the original CQL paper [36] referenced the DQN algorithm [33].

$$\mathcal{L}_{\mathrm{CQL}}(\theta) = \alpha \mathbb{E}_{s\sim D}\left[\log \sum_{a'} \exp\left(Q_\theta(s, a')\right) - \mathbb{E}_{a\sim D}[Q_\theta(s, a)]\right] + \mathcal{L}_{\mathrm{DoubleDQN}}(\theta) \tag{3}$$

The hyperparameters for CQL training were optimized through iterative experimentation. We set the batch size to 32, the learning rate to 0.0001, and the target network update interval to 200 iterations to ensure stable learning. The regularization parameter, alpha, was configured to two to effectively balance the conservative loss with the standard Q-learning loss. The training process was conducted over a total of 600,000 iterations (Q-values updates). These hyperparameter settings were uniformly applied across all CQL training runs for different datasets generated from the behavior policies previously described.

Although direct interactions with the environment are not possible during offline training, evaluations have been performed to assess progress throughout the training process. Figure 6 illustrates the progression of returns across evaluations conducted every 1000 training iterations for each offline policy, each of which was trained with a distinct dataset. Each data point represents the average return calculated from 100 episodes, with each episode consisting of a batch 15 parts. The return trajectories are smoothed in the figure to improve clarity and facilitate trend analysis.

The graph reveals that offline policy 1 (P1) shows a steady increase in returns and eventually converges over the training period. In contrast, offline policies 2 (P2) and 3 (P3), which incorporate greater exploration within the datasets, converge more

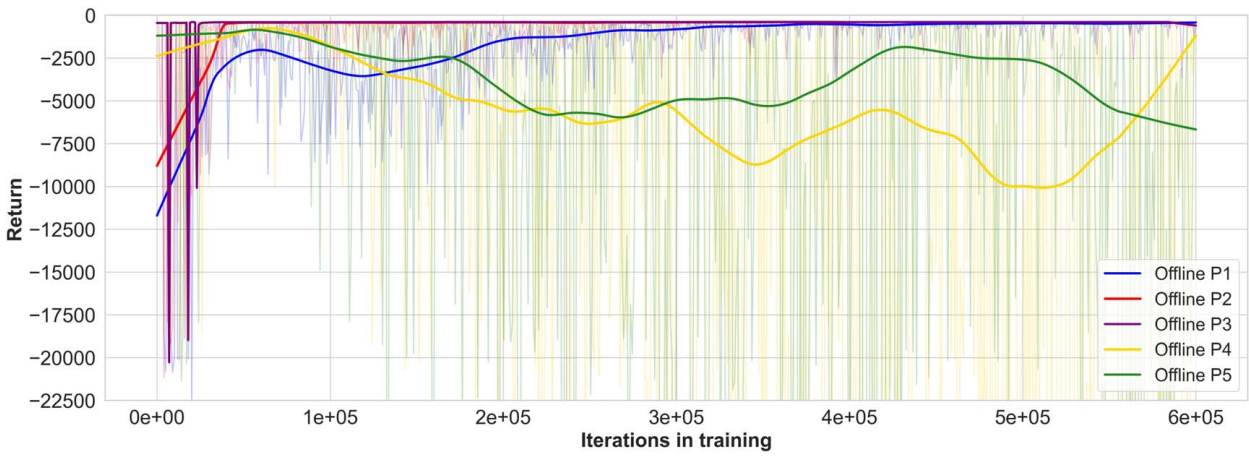

Fig. 6 Progression of returns during the training of multiple offline policies, with evaluations conducted every 1000 iterations

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024

Table 2 Performance comparison of policies relative to the fixed-cycle time baseline

Policy	Cycle time (s)	% Time improvement	Defective parts	Reward	% Reward improvement
Static	75,000	—	0	−75,000	—
Dynamic	68,090	9.21%	2	−88,090	−17.45%
Offline P1	67,800	9.6%	0	−67,800	9.6%
Offline P2	67,690	9.75%	0	−67,690	9.75%
Offline P3	67,495	10.01%	0	−67,495	10.01%
Offline P4	68,045	9.27%	0	−68,045	9.27%
Offline P5	72,735	3.02%	0	−72,735	3.02%

rapidly. However, the return curves for offline policies 4 (P4) and 5 (P5) deviate from the expected behavior, indicating less stable training and failing to reach the return levels achieved by the other policies. This instability is attributed to the more exploratory and random nature of the datasets used for these policies, which complicates the learning process in offline RL.

5 Results

This section provides a detailed analysis of offline RL policies, evaluating their performance relative to two baseline approaches: the dynamic expert policy and the static policy, which represents standard industry practice. The comparison with the static policy serves primarily to highlight the substantial improvements offered by dynamic solutions. The analysis is structured around two main objectives:

(1) First, to determine whether offline RL can advance data-driven policies beyond the capabilities of traditional supervised learning. This is achieved by comparing P1—trained on a dataset derived from behavior policy 1 (BP1) without additional exploration—with its corresponding behavior policy, which serves as the dynamic baseline. The objective of an SL approach is to replicate the performance metrics of the dynamic baseline, effectively matching its results. In contrast, the goal of RL is to surpass the performance metrics of the dynamic baseline.
(2) Second, to investigate the impact of data exploration on learning effectiveness. This involves assessing the performance of various offline policies (from P1 to P5), each trained with different levels of exploration, and identifying the optimal level of exploration required for effective offline RL training.

The performance of each policy was assessed through 50 randomly generated tests, with initial sheet temperatures uniformly distributed between 680°C and 880°C, simulating batches of 50 parts. In some tests, random perturbations introduced extreme temperatures (680°C or 880°C) by varying the sequence of sheet temperatures.

5.1 Performance Benchmarking Analysis. The performance analysis and benchmarking of policies center on two key metrics: total cycle time and the number of defective parts. Table 2 provides a summary of the numerical results, comparing each policy against the static baseline. This comparison is quantified as the percentage improvement over the static approach in both cycle time and reward. The reward metric is computed by aggregating total times and rewards across all evaluated episodes, where the reward function integrates both objectives–minimizing cycle time and reducing defect rates. To determine the most effective policy, we focus on the "% Reward Improvement" column, which reflects enhancements in batch processing efficiency while accounting for penalties related to defects.

Analysis of the reward metrics reveals that all offline RL policies outperform the dynamic policy, which incurs a negative percentage improvement due to penalties for defective parts. This highlights the effectiveness of offline RL in enhancing behavior policies.

Improvements in cycle times for offline policies compared to the dynamic policy are modest. Specifically, P5 does not match the performance of the dynamic policy, while P1 shows improvements in both cycle time and the number of non-defective parts. Figure 7 illustrates the impact of varying exploration levels on the accumulated batch production times across the 50 tests in our offline policy training case study. The plot compares the performance of offline policies against static and dynamic baselines, excluding defective parts. The x-axis displays ε values ranging from 0 to 0.8, while the y-axis represents the total accumulated time in seconds. The cycle times for the offline policies are depicted by a solid line marked with points. The lower dashed line indicates the accumulated batch production time for the dynamic policy, while the upper dashed line represents the accumulated batch production time for the static policy.

The plot reveals that precise exploration of the data is crucial for the effectiveness of offline RL agents. While some offline policies outperform the dynamic policy, others, particularly P5, result in longer cycle times, indicating less effective performance. Adequate exploration, as demonstrated by P2 and P3, enhances the algorithm's ability to refine both the initial policy and P1. However, excessive exploration can degrade performance, as observed with P5. Based on these results, we draw the following key insights:

(1) **Improvement through offline training:** Training with collected data can enhance the behavior policy, leading to better results than SL alone, as demonstrated by P1.
(2) **Impact of data exploration:** Moderate exploration during data collection (with ε values between 0.2 and 0.4), when feasible in a real-world environment, can improve learning by more effectively covering the state space, as seen with P2 and P3 compared to P1.
(3) **Risks of excessive exploration:** Excessive exploration destabilizes learning, as shown in Fig. 6, and can degrade solution quality, resulting in worse performance than SL, as observed with P5.

This study establishes offline RL as a viable alternative to online RL for enforcing safety constraints, expediting learning, and enhancing supervised learning methodologies. Nonetheless, it is important to recognize that safety risks are not completely mitigated during the deployment phase of models trained using offline RL.

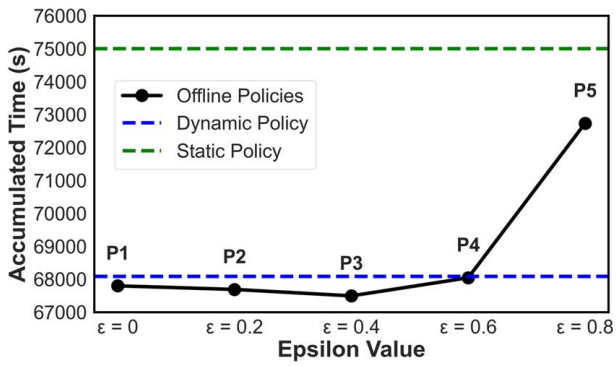

Fig. 7 Accumulated time for different ε values in offline policies

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024

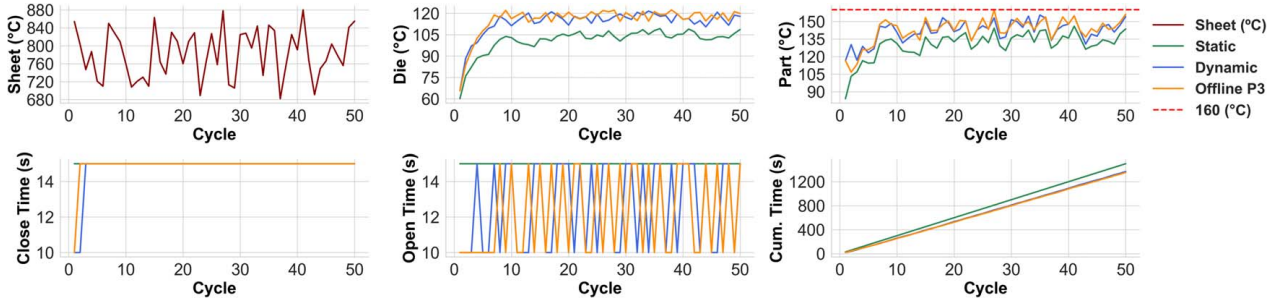

Fig. 8 Comparison of static and dynamic baseline policies with P3 ($\epsilon = 0.4$) in example scenario 1. Subplots show initial sheet temperature, die temperature, final part temperature, closing time, opening time, and total cycle time.

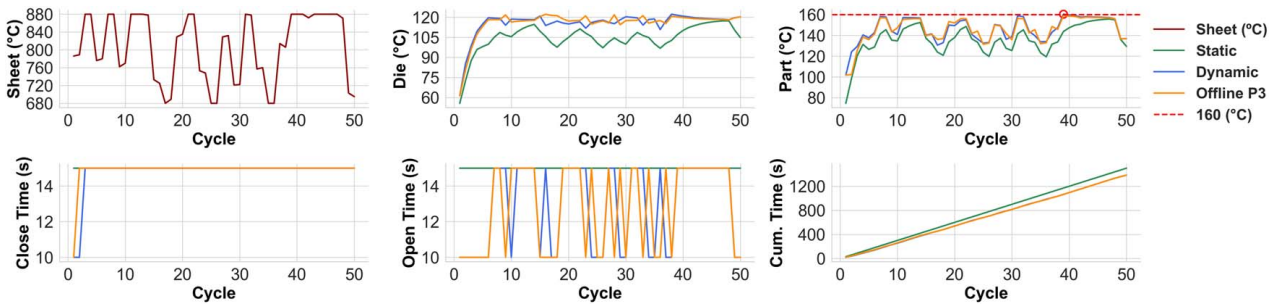

Fig. 9 Comparison of static and dynamic baseline policies with P3 ($\epsilon = 0.4$) in example scenario 2. Subplots show initial sheet temperature, die temperature, final part temperature, closing time, opening time, and total cycle time.

Therefore, validation of models prior to deployment, along with rigorous, controlled testing during deployment, is essential to ensure robust and reliable performance.

Furthermore, our analysis suggests that achieving significant performance improvements with offline RL requires a sufficiently robust behavior policy and a substantial amount of exploratory data. More research should focus on identifying which offline RL algorithms are most effective for various problem domains and exploring the impact of different neural network architectures and hyperparameters that were not covered in this study. It is also important to clarify that this study examines the application of offline RL using data collected from a policy trained via online RL. It does not provide a direct comparative analysis of online versus offline RL performance but rather demonstrates offline RL as a viable and effective alternative.

5.2 Policy Comparisons in Specific Scenarios. We present a detailed graphical analysis of two scenarios to evaluate the learned dynamic policies. These scenarios were selected to compare: (i) cases where offline RL outperforms the dynamic policy in terms of time, and (ii) cases where the time difference is minimal, but offline RL successfully completes all satisfactory parts, whereas the dynamic policy fails to complete one, thereby excelling in the quality objective. The scenarios are illustrated in Figs. 8 and 9. These figures compare key execution variables across three policies: the static baseline, the dynamic baseline, and P3. P3 was trained with a dataset at an ϵ value of 0.4 and demonstrated superior performance in previous benchmarking.

Each figure contains six plots illustrating the performance of different policies over a batch of 50 parts. The first three plots focus on temperature variables: the initial sheet temperature (as defined by each test episode), the variation in die temperature throughout the batch (affected by the policies' actions and initial sheet temperatures), and the final part temperature (which must remain below 160°C to ensure quality).

The figures reveal that, in both scenarios, the die temperature and final part temperature for dynamic policies are higher than those for the static policy. This is because the dynamic policies reduce cycle times, impacting these temperature variables. In Fig. 9, where initial sheet temperatures show more extreme variations, the dynamic baseline policy results in one part exceeding the maximum allowable temperature, an issue that P3 effectively avoids.

The remaining three plots address time-related variables: closing time and opening time, which are key process parameters, and the total cycle time, which combines both closing and opening times. This final plot highlights significant differences between the dynamic policies and the static baseline. However, the distinction in total cycle time between the baseline dynamic policy and P3 is relatively minor.

A critical aspect of the analysis is the behavior of the learned actions. To ensure that the final part temperature remains below the required 160°C, sufficient time must be allocated for the closing phase, resulting in a relatively consistent closing time across all policies. In contrast, the opening time exhibits greater variability, which is closely linked to the die temperature. This variability is necessary to provide an adequate rest period for the die, ensuring the smooth continuation of the batch process while also contributing to cycle time reduction.

Figure 8 reveals significant variability in opening times among the dynamic policies, with P3 achieving a lower total accumulated cycle time compared to the dynamic baseline. Figure 9 shows that, while the total accumulated times for both the dynamic baseline policy and P3 converge, P3 consistently avoids defective parts. This indicates a notable improvement in quality control. Although the gains from the offline RL approach are modest, the performance of P3 highlights its potential for refining data-driven strategies.

6 Conclusions

In this paper, we evaluate the application of offline RL in batch manufacturing, particularly in press hardening processes. Online RL methods are often impractical in such contexts due to safety constraints and physical limitations. Traditional RL approaches rely on virtual simulations, which require creating detailed simulation models, training RL agents against these models, and calibrating to bridge the simulation-to-reality gap. In contrast, supervised learning only replicates the behavior policy without improvements.

We propose that offline RL presents a significant advantage by leveraging existing data to refine and enhance the policy. This

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024

approach simplifies the RL pipeline for physical environments, eliminating the need for complex simulation models and iterative calibrations. Therefore, offline RL offers a more practical and effective solution for improving control policies in real-world applications.

Our study highlights the critical tradeoff between exploration during data collection and the physical constraints of the system. Achieving an optimal balance between exploration and safety is essential for maximizing the effectiveness of offline RL. We generated five datasets, each containing 20,000 interactions from a simulated environment using a DQN-trained agent. These datasets varied in exploration rates, from $\varepsilon = 0$ to $\varepsilon = 0.8$. It is important to note that the simulation environment was used solely for data generation, simulating what would otherwise be obtained from a real-world setting. We applied the CQL algorithm to train policies on each dataset and compared their performance with the initial dynamic policy and a static policy representing industry standards.

Our findings indicate that offline training has the potential to effectively improve the behavior policy. However, the quality of the learned policy heavily depends on the quality of the behavior policy used for data collection. The most effective policies in our study were achieved with datasets featuring exploration rates between 20% and 40%.

Looking ahead, offline RL shows significant potential for a wide range of applications. While our study focuses on the press hardening industry, we believe that offline RL has broader applicability in physical environments. Future research should investigate its effectiveness and scalability across various manufacturing processes to establish its general applicability and robustness.

Acknowledgment

This work was partly supported by the Industrial Doctorate Plan of the Secretariat of Universities and Research of the Department of Business and Knowledge of the Generalitat de Catalunya (AGAUR 2019 DI 87). Albert Abio is fellow of Eurecat's "Vicente López" Ph.D. grant program.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

Appendix: Detailed Explanation of Offline Reinforcement Learning methods

In this appendix, we present an introduction and brief description of the different types of offline RL methods, focusing on the most relevant methods in each category as outlined in the taxonomy described in Fig. 2. We also survey the main libraries that implement these methods, providing a preliminary guide on the available methods and relevant libraries.

Policy Constraint Methods. Policy constraint methods take a conservative approach, imposing limitations on the learned policy through an added penalty to the standard objective. These methods can be classified into two main types: distribution constraint and support constraint methods [21]. Distribution constraint methods enforce alignment with the behavior policy by constraining its distribution to exhibit a similar density, controlling divergence. Alternatively, support constraint methods restrict the learned policy to the subset of actions highly likely in the behavior policy, regardless of density. These methods emphasize specific actions by assigning them greater weight.

Distribution constraint methods include the behavior regularized actor critic (BRAC) [40], batch-constrained Q-learning (BCQ) [41], advantage-weighted regression [42], and advantage weighted actor critic [43]. For instance, the BCQ algorithm was employed for an energy management system in connected power-split hybrid electric vehicles to enhance energy efficiency and adaptability under complex driving conditions, outperforming a deep deterministic policy gradient (DDPG) algorithm. Examples of support constraint methods are the bootstrapping error accumulation reduction (BEAR) [26], the safe policy improvement with baseline bootstrapping [44], and the twin delayed DDPG with behavioral cloning [45] algorithms.

Policy constraint methods typically require an estimation of the behavior policy, which is often challenging and tends to result in excessively conservative approaches [21]. In contrast, methods involving uncertainty, incorporating regularization in the value function, or restricting training to collected data do not necessitate estimating the behavior policy.

Uncertainty Methods. To minimize errors, acquiring data across the entire state space remains pivotal. However, scenarios featuring high-risk attributes can pose challenges for safe testing in real-world environments, often resulting in a scarcity of data related to these extreme scenarios within the training dataset. In such instances, using more conservative Q-functions and/or reward functions may be necessary. These functions serve to penalize states where the agent demonstrates heightened uncertainty while exploring these regions. Yet, accurately estimating this uncertainty presents a challenge.

In model-free RL, uncertainty methods aim to gauge the uncertainty surrounding potential Q-functions derived from collected data, particularly for actions lying outside the distribution. Noteworthy examples encompass the random ensemble mixture [46] and stable Q-learning (SQL) [25] methodologies. Notably, the SQL approach, which relies on ensemble Q functions, was found to be applicable in a manufacturing setting for flatness control in the strip rolling industry, leveraging data gleaned from an operational steel mill.

A prevalent strategy in model-based offline RL involves leveraging ensemble model disagreement to estimate uncertainty. This methodology consists of training multiple simulation models and contrasting their predictions. When these predictions align, they are deemed more dependable and less uncertain compared to conflicting forecasts. Consequently, a penalty-based on the variance among the predictions of the different models, which quantifies the dispersion of these predictions from their mean—is integrated into the reward function during the RL agent's exploratory phase. However, determining the appropriate penalty level can pose a challenge, potentially leading to inadvertent penalization of accurate predictions and fostering an overly conservative stance that stifles exploration. Exemplars of such algorithms featuring conservative reward functions include model-based offline reinforcement learning (MOReL) [47] and model-based offline policy optimization (MOPO) [48].

The algorithm model-based Offline RL with restrictive exploration (MORE), introduced by Zhan et al. [22], employs a restrictive exploration scheme that considers both the reliability of predictions and the potential for OOD samples. This algorithm was specifically developed to optimize combustion in thermal power-generating units. Initially, a simulator was trained using historical data, and this simulator was then used to train the MORE agent. The effectiveness of the MORE agent was evaluated in real-world settings and validated using the D4RL benchmarks, covering three locomotion tasks and two types of datasets derived from partially trained Soft Actor-Critic (SAC) policies. Across these benchmarks, MORE demonstrated superior performance compared to behavior cloning (BC), BEAR, BRAC, BCQ, model-based policy optimization [49], and MOPO.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024

Table 3 Comparison of offline RL libraries comparison for PYTHON

Features	Tianshou	TorchRL	RLlib.	Pearl	d3RLpy	CORL	OfflineRL-Kit
# algorithms	+20	+12	+9	+16	+16	+11	+9
Model-free	✓	✓	✓	✓	✓	✓	✓
Model-based	✓	✓	✓	✗	✗	✗	✓
Offline-oriented	✗	✗	✗	✗	✓	✓	✓
# offline algorithms	+5	+3	+3	+2	+9	+11	+9
API documentation	Comp.	Comp.	Comp.	Suff.	Comp.	Suff.	Suff.
Tutorials & examples	Comp.	Comp.	Comp.	Suff.	Comp.	Lim.	Suff.
Data collection	✗	✗	✗	✓	✓	✗	✗
OPE	✗	✗	✓	✗	✓	✗	✗
Backend	PT	PT	PT/TF	PT	PT	PT	PT

Notably, the MORE model was successfully deployed in four large coal-fired power plants in China.

Value Function Regularization Methods. Value function regularization methods offer effective mitigation of overestimation in OOD scenarios. Conservative Q-learning (CQL) [36], which employs a conservative evaluation strategy to estimate Q-values. This is achieved by incorporating a penalty into the loss function, grounded on the disparity between the behavior and target policies, with the aim of alleviating overestimation.

Another approach in this domain is conservative offline model-based policy optimization (COMBO) [50]. Unlike some model-based methods relying on multiple models, COMBO relies on a single model. In COMBO, the reward function remains unaltered, with emphasis placed on a conservative evaluation of the Q-values to optimize a lower bound on expected performance. A penalty is introduced to diminish the Q-values when evaluated in situations deemed less reliable or less akin to the training data. The policy is then updated using this conservative estimate of Q. Both the original offline data and the data generated by the simulation model are leveraged for training purposes.

Limit Training on Collected Data Methods. Methods that restrict training to collected data are designed to circumvent queries about unseen actions in the collected data. These methods effectively address the problem of overestimation prevalent in previous algorithms. However, they introduce a challenge related to underestimation. Implicit Q-learning [51] overcomes this issue by incorporating expected loss, where negative errors receive a distinct coefficient compared to positive errors, thus imposing a heavier penalty on negative errors.

Recent advancements in leveraging transformers for modeling trajectories have facilitated the capture of intricate dependencies within offline data. In the model-free domain, the decision transformer (DT) [52] analyzes trajectories composed of sequences of actions and rewards-to-go. Here, rewards-to-go encompass all future rewards within the trajectory at each time-step. Consequently, instead of selecting actions to maximize cumulative future rewards, the model selects actions conditioned on the desired cumulative future reward. Notably, DT demonstrates commendable performance compared to other state-of-the-art offline RL algorithms, particularly excelling in environments with sparse rewards. Wang et al. [23] applied the DT algorithm to address cloud manufacturing scheduling challenges outperforming several baseline methods, including online Deep Reinforcement Learning (DRL) algorithms and priority dispatching rules, across various scenarios, showcasing its efficacy in real-world applications.

Offline Reinforcement Learning Frameworks. There are numerous libraries dedicated to implementing RL algorithms for PYTHON. Some exclusively focus on online RL algorithms, including Stable Baselines3 [35], TensorForce [53], ReAgent [54], MushroomRL [55], RL_Coach [56], and CleanRL [57]. Conversely, there are libraries that cover both online and offline RL algorithms, such as Tianshou[2] [58], TorchRL[3] [59], RLlib[4] [60], and Pearl[5] [61]. Additionally, specific libraries cater exclusively to offline RL, notable among them being d3RLpy[6] [62], CORL[7] [63], and OfflineRL-Kit[8] [64]. Table 3 compares libraries implementing offline RL algorithms and those actively maintained with recent and continuous updates.

We assess the documentation quality and the availability of tutorials and examples in each library with the goal of enhancing user accessibility and comprehension. "Complete" (Comp.) indicates comprehensive documentation and examples, "Sufficient" (Suff.) denotes an adequate level of documentation, and "Limited" (Lim.) signifies a constrained amount of tutorials or examples. The table provides information on the backend framework used by each library, which is either TensorFlow (TF) or PyTorch (PT).

While offline training data often comes from pre-existing datasets, we emphasize the importance of libraries that facilitate the creation of diverse datasets from environments when an environment is available. This versatility enables users to generate datasets with varying expertise levels, a crucial factor for the subsequent training of RL agents. To this end, we explore whether the libraries are equipped with data collection functionalities within the environment.

Given the complexity of evaluating offline RL agents without access to a virtual environment, we also investigate whether the libraries incorporate off-policy evaluation (OPE) methods to assess the performance of policies without deploying them. Finally, Table 3 lists the number of algorithms implemented (# algorithms and # offline algorithms), whether the library has model-free or model-based implementations, and whether the library is oriented toward offline RL. The "+" symbol is included in the algorithm count to account for instances where certain libraries feature multiple versions of the same algorithm, with minimal differentiation, thus not being considered as distinct algorithms.

References

[1] Jiang, Y., Yin, S., Dong, J., and Kaynak, O., 2020, "A Review on Soft Sensors for Monitoring, Control, and Optimization of Industrial Processes," IEEE Sens. J., **21**(11), pp. 12868–12881.

[2] Ang, K. H., Chong, G., and Li, Y., 2005, "PID Control System Analysis, Design, and Technology," IEEE Trans. Control Syst. Technol., **13**(4), pp. 559–576.

[3] Ameer, M., and Dahane, M., 2023, "Reconfigurability Improvement in Industry 4.0: A Hybrid Genetic Algorithm-Based Heuristic Approach for a Co-Generation of Setup and Process Plans in a Reconfigurable Environment," J. Intell. Manuf., **34**(3), pp. 1445–1467.

[4] Schwenzer, M., Ay, M., Bergs, T., and Abel, D., 2021, "Review on Model Predictive Control: An Engineering Perspective," Int. J. Adv. Manuf. Technol., **117**(5), pp. 1327–1349.

[5] Espinosa-Leal, L., Chapman, A., and Westerlund, M., 2020, "Autonomous Industrial Management Via Reinforcement Learning," J. Intell. Fuzzy Syst., **39**(6), pp. 8427–8439.

[6] Sutton, R. S., and Barto, A. G., 2018, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.

[2]GitHub Repository: https://github.com/thu-ml/tianshou
[3]GitHub Repository: https://github.com/pytorch/rl
[4]GitHub Repository: https://github.com/ray-project/ray/tree/master/rllib
[5]GitHub Repository: https://github.com/facebookresearch/Pearl
[6]GitHub Repository: https://github.com/takuseno/d3rlpy
[7]GitHub Repository: https://github.com/tinkoff-ai/CORL
[8]GitHub Repository: https://github.com/yihaosun1124/OfflineRL-Kit

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024

[7] Shi, F., Cao, H., Zhang, X., and Chen, X., 2020, "A Reinforced K-Nearest Neighbors Method With Application to Chatter Identification in High-Speed Milling," IEEE Trans. Ind. Electron., **67**(12), pp. 10844–10855.

[8] Masinelli, G., Le-Quang, T., Zanoli, S., Wasmer, K., and Shevchik, S. A., 2020, "Adaptive Laser Welding Control: A Reinforcement Learning Approach," IEEE Access, **8**, p. 103803–103814.

[9] Quang, T. L., Meylan, B., Masinelli, G., Saeidi, F., Shevchik, S. A., Farahani, F. V., and Wasmer, K., 2022, "Smart Closed-Loop Control of Laser Welding Using Reinforcement Learning," Proc. CIRP, **111**, pp. 479–483.

[10] Li, C., Zheng, P., Yin, Y., Wang, B., and Wang, L., 2023, "Deep Reinforcement Learning in Smart Manufacturing: A Review and Prospects," CIRP J. Manuf. Sci. Technol., **40**, pp. 75–101.

[11] Li, X., Luo, Q., Wang, L., Zhang, R., and Gao, F., 2022, "Off-Policy Reinforcement Learning-Based Novel Model-Free Minmax Fault-Tolerant Tracking Control for Industrial Processes," J. Process Control, **115**, pp. 145–156.

[12] Han, X., Mu, C., Yan, J., and Niu, Z., 2023, "An Autonomous Control Technology Based on Deep Reinforcement Learning for Optimal Active Power Dispatch," Int. J. Electr. Power Energy Syst., **145**, p. 108686.

[13] Ma, Z., and Pan, T., 2024, "DRL-DEWMA: A Composite Framework for Run-to-Run Control in the Semiconductor Manufacturing Process," Neural Comput. Appl., **36**(3), pp. 1429–1447.

[14] Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T., 2021, "Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks and Analysis," Mach. Learn., **110**(9), pp. 2419–2468.

[15] Beltran-Hernandez, C., Petit, D., Ramirez-Alpizar, I., and Harada, K., 2020, "Variable Compliance Control for Robotic Peg-in-Hole Assembly: A Deep-Reinforcement-Learning Approach," Appl. Sci., **10**(19), pp. 1–17.

[16] Ahn, K.-H., Na, M., and Song, J.-B., 2023, "Robotic Assembly Strategy Via Reinforcement Learning Based on Force and Visual Information," Rob. Auton. Syst., **164**, p. 104399.

[17] Nguyen, H., Kozuno, T., Beltran-Hernandez, C. C., and Hamaya, M., 2024, "Symmetry-Aware Reinforcement Learning for Robotic Assembly Under Partial Observability With a Soft Wrist," arXiv preprint arXiv:2402.18002.

[18] Liu, X.-Y., and Wang, J.-X., 2021, "Physics-Informed DYNA-Style Model-Based Deep Reinforcement Learning for Dynamic Control," Proc. R. Soc. A, **477**(2255), p. 20210618.

[19] Ramesh, A., and Ravindran, B., 2023, "Physics-Informed Model-Based Reinforcement Learning," Proceedings of The 5th Annual Learning for Dynamics and Control Conference, Philadelphia, PA, June 14–16, PMLR, pp. 26–37

[20] Nievas, N., Pagès-bernaus, A., Bonada, F., and Echeverria, L., 2024, "Reinforcement Learning for Autonomous Process Control in Industry 4. 0: Advantages and Challenges," Appl. Artif. Intell., **38**(1), pp. 0–53.

[21] Levine, S., Kumar, A., Tucker, G., and Fu, J., 2020, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," pp. 1–43.

[22] Zhan, X., Xu, H., Zhang, Y., Zhu, X., Yin, H., and Zheng, Y., 2022, "DeepThermal: Combustion Optimization for Thermal Power Generating Units Using Offline Reinforcement Learning," Proceedings of the 36th AAAI Conference on Artificial Intelligence, Virtual, Feb. 22–Mar. 1, AAAI 2022, Vol. 36, pp. 4680–4688.

[23] Wang, X., Zhang, L., Liu, Y., and Zhao, C., 2023, "Logistics-Involved Task Scheduling in Cloud Manufacturing With Offline Deep Reinforcement Learning," J. Ind. Inf. Integr., **34**, p. 100471.

[24] He, H., Niu, Z., Wang, Y., Huang, R., and Shou, Y., 2023, "Energy Management Optimization for Connected Hybrid Electric Vehicle Using Offline Reinforcement Learning," J. Energy Storage, **72**, p. 108517.

[25] Deng, J., Sierla, S., Sun, J., and Vyatkin, V., 2023, "Offline Reinforcement Learning for Industrial Process Control: A Case Study From Steel Industry," Inf. Sci., **632**, pp. 221–231.

[26] Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S., 2019, "Stabilizing Off-Policy Q-Learning Via Bootstrapping Error Reduction," Advances in Neural Information Processing Systems, NeurIPS 2019, Vancouver, BC, Canada, Dec. 8–14, Vol. 32, pp 11761–11771.

[27] Merklein, M., Lechler, J., and Stoehr, T., 2008, "Characterization of Tribological and Thermal Properties of Metallic Coatings for Hot Stamping Boron-Manganese Steels," Proceedings of the Seventh International Conference on Coatings in Manufacturing Engineering, Chalkidiki, Greece, Oct. 1–3, pp. 1–3.

[28] Åkerström, P., 2006, "Modelling and Simulation of Hot Stamping," Doctoral dissertation, Luleå Tekniska Universitet, Luleå, Sweden.

[29] Garcia-Llamas, E., Pujante, J., Torres, P., and Bonada, F., 2021, "A Thermography-Based Online Control Method for Press Hardening," IOP Conf. Ser.: Mater. Sci. Eng., **1157**(1), p. 012010.

[30] Nishibata, T., and Kojima, N., 2010, "Effect of Quenching Rate on Hardness and Microstructure of Hot-Stamped Steel," TETSU to Hagane-J. Iron Steel Inst. Jpn, **96**(6), pp. A378–A385.

[31] Abio, A., Bonada, F., Pujante, J., Grané, M., Nievas, N., Lange, D., and Pujol, O., 2022, "Machine Learning-Based Surrogate Model for Press Hardening Process of 22MnB5 Sheet Steel Simulation in Industry 4.0," Materials, **15**(10), p. 3647.

[32] Pujante, J., García-Llamas, E., and Casellas, D., 2019, "Study of Wear in Press Hardening Using a Pilot Facility," Proceedings of the 7th International Conference Hot Sheet Metal Forming of High-Performance Steel, Lulea, Sweden, June 2–5, pp. 151–158.

[33] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., et al., 2015, "Human-Level Control Through Deep Reinforcement Learning," Nature, **518**(7540), pp. 529–533.

[34] Watkins, C. J. C. H., and Dayan, P., 1992, "Q-Learning," Mach. Learn., **8**(3–4), pp. 279–292.

[35] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N., 2021, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," J. Mach. Learn. Res., **22**(268), pp. 1–8.

[36] Kumar, A., Zhou, A., Tucker, G., and Levine, S., 2020, "Conservative Q-Learning for Offline Reinforcement Learning," Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Vitrual, Dec. 6–12, pp. 1–13.

[37] Gao, C., Xu, K., Zhou, K., Li, L., Wang, X., Yuan, B., and Zhao, P., 2022, "Value Penalized Q-Learning for Recommender Systems," Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11–15, pp. 2008–2012.

[38] Bayramoğlu, Ö. Z., Erzin, E., Sezgin, T. M., and Yemez, Y., 2021, "Engagement Rewarded Actor-Critic With Conservative Q-Learning for Speech-Driven Laughter Backchannel Generation," The 23rd ACM International Conference on Multimodal Interaction (ICMI 2021), Montreal, Canada, Oct. 18–22, pp. 613–618.

[39] Van Hasselt, H., Guez, A., and Silver, D., 2016, "Deep Reinforcement Learning With Double Q-Learning," Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, Feb. 12–17, Vol. 30, No. 1, pp. 2094–2100.

[40] Wu, Y., Tucker, G., and Nachum, O., 2019, "Behavior Regularized Offline Reinforcement Learning," pp. 1–25.

[41] Fujimoto, S., Meger, D., and Precup, D., 2019, "Off-Policy Deep Reinforcement Learning Without Exploration," 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, June 9–15, pp. 3599–3609.

[42] Peng, X. B., Kumar, A., Zhang, G., and Levine, S., 2019, "Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning".

[43] Nair, A., Gupta, A., Dalal, M., and Levine, S., 2020, "AWAC: Accelerating Online Reinforcement Learning With Offline Datasets".

[44] Laroche, R., Trichelair, P., and Des Combes, R. T., 2019, "Safe Policy Improvement With Baseline Bootstrapping," 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, June 9–15, pp. 6487–6520.

[45] Fujimoto, S., and Gu, S. S., 2021, "A Minimalist Approach to Offline Reinforcement Learning," 35th Annual Conference on Neural Information Processing Systems, NeurIPS 2021, Virtual, Dec. 6–14, Vol. 24, pp. 20132–20145.

[46] Agarwal, R., Schuurmans, D., and Norouzi, M., 2020, "An Optimistic Perspective on Offline Reinforcement Learning," 37th International Conference on Machine Learning, ICML 2020, Virtual, July 13–18, pp. 92–102.

[47] Kidambi, R., and Joachims, T., 2020, "MOReL : Model-Based Offline Reinforcement Learning".

[48] Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T., 2020, "MOPO : Model-based Offline Policy Optimization," pp. 1–14.

[49] Janner, M., Fu, J., Zhang, M., and Levine, S., 2019, "When to Trust Your Model: Model-Based Policy Optimization," Advances in Neural Information Processing Systems, NeurIPS 2019, Vancouver, BC, Canada, Dec. 8–14, Vol. 32.

[50] Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C., 2021, "COMBO: Conservative Offline Model-Based Policy Optimization," 35th Annual Conference on Neural Information Processing Systems, NeurIPS 2021, Virtual, Dec. 6–14, Vol. 35, pp. 28954–28967.

[51] Kostrikov, I., Nair, A., and Levine, S., 2022, "Offline Reinforcement Learning With Implicit Q-Learning," ICLR 2022 - 10th International Conference on Learning Representations, Virtual, Apr. 25–29, pp. 1–13.

[52] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I., 2021, "Decision Transformer: Reinforcement Learning Via Sequence Modeling Equal Contribution † Equal Advising," NeurIPS 2021, Virtual, Dec. 6–14, pp. 15084–15097.

[53] Kuhnle, A., Schaarschmidt, M., and Fricke, K., 2017, "Tensorforce: A Tensorflow Library For Applied Reinforcement Learning," Web Page.

[54] Gauci, J., Conti, E., Liang, Y., Virochsiri, K., Chen, Z., He, Y., Kaden, Z., Narayanan, V., and Ye, X., 2018, "Horizon: Facebook's Open Source Applied Reinforcement Learning Platform," arXiv preprint arXiv:1811.00260.

[55] D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J., 2021, "Mushroomrl: Simplifying Reinforcement Learning Research," J. Mach. Learn. Res., **22**(131), pp. 1–5.

[56] Caspi, I., Leibovich, G., Novik, G., and Endrawis, S., 2017, "Reinforcement Learning Coach," December.

[57] Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and Araújo, J. G., 2022, "CleanRL: High-Quality Single-File Implementations of Deep Reinforcement Learning Algorithms," J. Mach. Learn. Res., **23**(274), pp. 1–18.

[58] Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H., and Zhu, J., 2022, "Tianshou: A Highly Modularized Deep Reinforcement Learning Library," J. Mach. Learn. Res., **23**(267), pp. 1–6.

[59] Bou, A., Bettini, M., Dittert, S., Kumar, V., Sodhani, S., Yang, X., Fabritiis, G. D., and Moens, V., 2023, "TorchRL: A Data-Driven Decision-Making Library for PyTorch".

[60] Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I., 2018, "RLlib: Abstractions for Distributed Reinforcement Learning," 35th International Conference on Machine Learning, Stockholm, Sweden, July 10–15, pp. 3053–3062.

[61] Zhu, Z., de Salvo Braz, R., Bhandari, J., Jiang, D., Wan, Y., Efroni, Y., Wang, L., et al., 2023, "Pearl: A Production-Ready Reinforcement Learning AI Agent Library".

[62] Seno, T., and Imai, M., 2022, "d3rlpy: An Offline Deep Reinforcement Learning Library," J. Mach. Learn. Res., **23**(315), pp. 1–20.

[63] Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S., 2023, "CORL: Research-Oriented Deep Offline Reinforcement Learning Library," NeurIPS 2023, the 37th Annual Conference on Neural Information Processing Systems, New Orleans, LA, Dec. 10–16, pp. 30997–31020.

[64] Sun, Y., 2023, "OfflineRL-kit: An Elegant PyTorch Offline Reinforcement Learning Library". https://github.com/yihaosun1124/OfflineRL-Kit

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/1/011004/7401986/cise_25_1_011004.pdf by guest on 13 December 2024