APEM journal

Advances in Production Engineering & Management
Volume 20 | Number 1 | March 2025 | pp 5–17
https://doi.org/10.14743/apem2025.1.523

ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Reinforcement learning for robot manipulation tasks in human-robot collaboration using the CQL/SAC algorithms

Husaković, A.[a], Banjanović-Mehmedović, L.[b], Gurdić-Ribić, A.[c], Prljača, N.[b], Karabegović, I.[d]

[a]Eacon doo, Zenica, Bosnia and Herzegovina
[b]University of Tuzla, Faculty of Electrical Engineering, Tuzla, Bosnia and Herzegovina
[c]Foundation for Innovation, Technology and Transfer of Knowledge, Tuzla, Bosnia and Herzegovina
[d]Academy of Sciences and Arts of Bosnia and Herzegovina, Sarajevo, Bosnia and Herzegovina

ABSTRACT

The integration of human-robot collaboration (HRC) into industrial and service environments demands efficient and adaptive robotic systems capable of executing diverse tasks, including pick-and-place operations. This paper investigates the application of Soft Actor-Critic (SAC) and Conservative Q-Learning (CQL)—two deep reinforcement learning (DRL) algorithms—for the learning and optimization of pick-and-place actions within HRC scenarios. By leveraging SAC's capability to balance exploration and exploitation, the robot autonomously learns to perform pick-and-place tasks while adapting to dynamic environments and human interactions. Moreover, the integration of CQL ensures more stable learning by mitigating Q-value overestimation, which proves particularly advantageous in offline and suboptimal data scenarios. The combined use of CQL and SAC enhances policy robustness, facilitating safer and more efficient decision-making in continually evolving environments. The proposed framework combines simulation-based training with transfer learning techniques, enabling seamless deployment in real-world environments. The critical challenge of trajectory completion is addressed through a meticulously designed reward function that promotes efficiency, precision, and safety. Experimental validation demonstrates a 100 % success rate in simulation and an 80 % success rate on real hardware, confirming the practical viability of the proposed model. This work underscores the pivotal role of DRL in enhancing the functionality of collaborative robotic systems, illustrating its applicability across a range of industrial environments.

ARTICLE INFO

Keywords:
Human-robot collaboration;
Robot learning;
Deep reinforcement learning;
Soft actor-critic algorithm (SAC);
Conservative Q-learning (CQL);
Robot manipulation tasks

**Corresponding author:*
lejla.banjanovic-mehmedovic@fet.ba
(Banjanović-Mehmedović, L.)

Article history:
Received 10 February 2025
Revised 2 March 2025
Accepted 7 March 2025

1. Introduction

Human-robot collaboration (HRC), driven by Industry 4.0 and 5.0, has become a cornerstone of modern industrial and service robotics, where robots and humans work together to achieve common goals. The human worker is tasked with solving social challenges and making decisions, while the cobot dictates the speed and acceleration of the process [1]. These collaborative environments require robots that are not only efficient and precise but also adaptable to dynamic scenarios involving human interaction [2]. HRC systems, within the framework of Industry 4.0, contribute to improving workflow, accelerating processes, enhancing product quality, reducing energy consumption and CO2 emissions. In the context of Industry 5.0, HRC is worker-centered, focusing on creating a positive and conflict-free working environment [3].

In the evolution of smart robotic manufacturing, the ability to adapt to uncertainties, changing environments, and dynamic tasks has become a critical necessity, often described as the emergence of *program-free robots* or *learning-enabled robots* [4]. Robot learning integrates vari-

ous machine learning techniques into robotics [5], focusing on enabling robots to learn and perform actions based on environmental inputs. This paradigm addresses challenges such as optimize policies for real-time decision-making, managing the exploration-exploitation trade-off, incorporate feedback to adapt to evolving tasks and conditions, and ensuring robust transfer of learning from simulations to real-world applications. These factors make robot learning a unique and complex area within machine learning and robotics.

Robotic manipulation is a fundamental challenge in robotics, involving tasks such as grasping, object handling, and assembly. These tasks are inherently complex due to high-dimensional state and action spaces, environmental uncertainties, and the dynamic interactions between the robot, objects, and human collaborators. Incorporating human-cobot collaboration adds another layer of complexity, as robots must effectively and safely interact with humans in shared workspaces. The safety distance between humans and machines, based on real-time changes in various scenarios, ensures smoother and safer collaboration [6]. The paper [7] emphasizes the importance of evaluating collaborative workplaces in both simulation and real-world environments, showing that such evaluations can enhance overall industrial system capacity This requires a high level of adaptability, precision, and real-time decision-making to ensure seamless task sharing, coordination, and safety in dynamic industrial or service environments [8-10].

Learning for robot manipulation is essential for enabling robots to adapt to complex, unstructured environments and perform tasks that require interacting with diverse objects [11]. Reinforcement learning (RL) has proven effective in training robots to perform intricate manipulation and grasping tasks, assembly and disassembly tasks, which is vital for advancing human-robot collaboration. By equipping robots with the ability to adapt and respond dynamically to their environment and human partners, RL enhances their role in shared workspaces across manufacturing, warehousing, healthcare, and service robotics applications [12].

Applying DRL in continual, dynamic environments is crucial for enabling robots to adapt to real-world uncertainties, optimize decision-making over time, and improve efficiency in tasks such as robotic manipulation and autonomous navigation [8]. Conservative Q-Learning (CQL) and Soft Actor-Critic (SAC) are both off-policy reinforcement learning algorithms tailored for continuous control, with SAC optimizing a stochastic policy using entropy regularization to encourage exploration, while CQL focuses on learning conservative Q-values to mitigate overestimation and improve robustness in offline learning [13]. While SAC excels in dynamic environments with abundant data, CQL is particularly useful in settings with limited or suboptimal datasets by constraining the learned Q-values to avoid risky actions [14].

Hierarchical Reinforcement Learning (HRL) is an AI approach that structures decision-making by breaking complex tasks into manageable sub-tasks, enhancing learning efficiency [15]. A Hierarchical Reinforcement Learning (HRL) approach enhances human-robot interactions by breaking complex tasks into smaller sub-tasks, allowing the robot to independently optimize each one while contributing to the overall goal. HRL improves learning efficiency by enabling the robot to solve simpler problems faster, adapts better to human inputs like gestures or commands, and offers scalability as new sub-tasks can be added without disrupting the system. It also decomposes complex tasks, such as assembly or assistance in dynamic environments, into manageable components, improving overall performance and flexibility.

This paper focuses on applying the CQL/SAC algorithm to develop a framework for learning and executing two sub-tasks, pick-and-place actions in human-robot collaborative settings. The proposed approach addresses several critical challenges, including trajectory optimization, and autonomously and safe interaction with human collaborators. By integrating reinforcement learning into the HRC domain, this paper aims to improve the robot's ability to adapt to varying conditions while maintaining task efficiency and safety.

The paper is structured as follows. Section 2 reviews related work on robot manipulation, with focus on pick-and-place operations and reinforcement learning. Section 3 presents the proposed deep reinforcement methodology, detailing the CQL/SAC-based learning framework. Section 4 discusses the system description, including the collaborative robotics platform and reward function design. Results are presented in Section 5, demonstrating the effectiveness of the CQL/SAC algorithm in optimizing pick-and-place actions. Finally, Section 6 presents the conclusion of this study and discusses potential directions for future research.

2. Related work

The precise object grasping is essential for robots, especially in industrial settings. Most learning-driven grasping tasks rely on vision signals, requiring models with strong representational capabilities. Using deep convolutional neural networks and a guided policy search method, the study presented in paper [16] enables robots to learn policies that map raw visual inputs directly to motor commands. The approach is validated on real-world tasks requiring vision-control coordination, such as assembling a bottle cap.

Reinforcement Learning (RL) enables robots to learn manipulation and collaboration skills through interaction with their environment and human partners. By optimizing a reward function, robots can develop policies to perform complex tasks and adapt to human inputs. The deep reinforcement learning and deep neural networks like CNNs are commonly used, showcasing the power of robot learning-based grasping. Mahler *et al*. used CNN and DQN for pick-and-transport tasks with an ABB Yumi robot [12]. Mohammed *et al*. utilized DQN and Q-learning with RGBD input for target position generation on UR robots [17].

Recent research has demonstrated the effectiveness of DRL in various task-specific applications [18]. Liu *et al*. [19] designed a robotic pick-and-place system, emphasizing reward shaping to address complex challenges. Rewards were based on the Euclidean distance between an object's current and target configurations, with additional rewards for task completion, significantly improving convergence compared to linear reward methods. The study also demonstrated that the employed learning technique outperformed both PPO and Actor-Critic (A3C) by directly generating manipulation signals, thereby effectively managing high task complexity. A simulated pick-and-place task with a simple block, using DDPG enhanced with hindsight experience replay (HER), is presented in [20].

In [21], a flexible framework was proposed that integrates motion planning with RL for continuous robot control in cluttered environments. By combining model-free RL with a sampling-based motion planner, this approach minimizes dependency on task-specific knowledge and enables the RL policy to determine when to plan or execute direct actions through reward maximization. Additionally, [22] introduced a framework that leverages demonstrations, unsupervised learning, and RL to efficiently learn complex tasks using only image input.

SAC algorithm, has demonstrated broad applicability in tasks like door opening and block stacking by breaking the task into fundamental steps (e.g., reaching, grasping, turning, and pulling for door opening) and training each step individually [23]. The advantage of SAC in path planning lies in entropy maximization, which encourages the robot to explore its surroundings, while the addition of HER (hindsight experience replay) allows the use of past experiences for improved learning and environmental adaptation [24]. In paper [25], a novel framework for sequentially learning vision-based robotic manipulation tasks in offline reinforcement learning settings was presented. Their approach leverages offline data and visual inputs to train adaptable policies for complex, sequential manipulation scenarios, demonstrating improved performance and generalization in robotic applications.

Multi-Agent Deep Reinforcement Learning (MADRL) involves multiple agents learning and interacting in the same environment, either through collaboration or competition. Each agent learns to optimize its own policy while considering the actions of others. It is widely used in robotics and industrial automation for tasks such as coordinated robot control, resource allocation, and adaptive decision-making in complex systems. The paper [26] employs a multi-agent deep reinforcement learning (DRL) approach, which is a significant advancement in the field of production scheduling. This method incorporates an attention mechanism within an advantage actor-critic framework, which is further complemented by a global reward function.

3. Deep reinforcement learning

Deep Reinforcement Learning (DRL) has emerged as a transformative approach, providing adaptive and intelligent solutions for optimizing robotics and production processes in complex industrial environments. It enhances efficiency by enabling autonomous systems to learn optimal strategies for navigation, manipulation, and workflow automation [27].

Finding the optimal policy in reinforcement learning is influenced by whether the method is model-based or model-free. Model-based approaches estimate transition probabilities $p(s'|s,a)$ using prior knowledge or state-space searches, while model-free methods learn directly from rewards without modelling system dynamics. Model-free techniques are common in robotics due to the complexity of modelling continuous state spaces.

RL requires balancing exploration (random actions to discover optimal strategies) and exploitation (choosing actions to maximize rewards). This trade-off is managed using a parameter ε, which controls the likelihood of exploring non-optimal actions. On-policy learning adjusts the current policy by mixing optimal and random actions, while off-policy learning uses a target policy for optimization and a behaviour policy for exploration.

Within the RL/DRL approaches, we distinguish several key methods: value-based, policy-based, and actor-critic methods [28]. Value-based methods are generally more suitable for problems with discrete action spaces**.** For continuous action spaces, policy-based or actor-critic methods are typically preferred.

Value function learning leverages the value function to assess the quality of a given state when the agent follows a specific policy. Consequently, a robotic agent can refine its policy by exploring the action space to identify actions that maximize the estimated value function. Key Value methods are Q-learning, SARSA, Deep Q-Networks (DQN), Double Q-learning [29]. Q-learning, uses the maximum Q-value for the next state to update Q-values. It can lead to overestimation in some cases. Deep Q-Network (DQN) employs deep neural networks to estimate Q-values, enabling efficient decision-making in environments with complex and high-dimensional state spaces. It stabilizes training with experience replay and target networks. Double DQN mitigates overestimation bias by utilizing two separate Q-networks: one to determine the best action and another to evaluate its corresponding Q-value, leading to more accurate value estimations. SARSA is an on-policy method, meaning it updates the Q-values based on the actions that the agent actually takes while following its policy, as opposed to Q-learning, which is an off-policy method and updates the Q-values based on the maximum possible future reward.

Policy gradient directly optimizes the policy in a model-free manner by maximizing the accumulated reward. Policy-based algorithms, compared to value-based methods, typically provide better convergence and can learn stochastic policies. Examples of policy-based methods in Reinforcement Learning (RL) include Proximal Policy Optimization (PPO), Deterministic Policy Gradient (DPG), and Trust Region Policy Optimization (TRPO). They focus on directly optimizing the policy to improve decision-making, rather than learning a value function.

PPO is a policy-based method that improves the stability of policy updates using a clipped objective function. PPO is designed to handle large, complex environments and is widely used due to its efficiency and simplicity. TRPO is another policy-based method that ensures updates to the policy are within a *trust region* to prevent overly large changes that can destabilize learning. It uses a constrained optimization approach, making it more computationally expensive but stable and effective. DPG is a policy gradient method specifically designed for continuous action spaces. It learns a deterministic policy and utilizes a policy gradient approach to directly update the policy without needing a value function for actions.

Actor-Critic algorithms integrate two key components: an actor, responsible for determining actions, and a critic, which assesses the chosen actions' value. By optimizing both networks concurrently, these algorithms enhance stability and learning efficiency in reinforcement learning. Notable Actor-Critic approaches include Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C), Deep Deterministic Policy Gradient (DDPG), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC). The summary of Actor-critic algorithms is presented in Table 1.

The main challenges of applying reinforcement learning to manipulation tasks in robotics include the high dimensionality, the continuous action space, and the significant training time required [30]. Model-free approaches like PPO, DDPG and SAC are widely used in robot manipulation. These algorithms excel in learning continuous control policies for complex robotic arms and manipulators. In our research, we used CQL/SAC algorithm.

Table 1 Characteristics of Actor-critic algorithms

Algorithm	Type	Key Features	Action Space
A2C	Synchronous	Advantage function to reduce variance	Discrete and continuous
A3C	Asynchronous	Multiple parallel agents for stability and efficiency	Discrete and continuous
DDPG	Off-Policy	Deterministic policy, used for continuous action spaces	Continuous
TD3	Off_policy	Double Q-learning and target network smoothing to reduce bias	Continuous
SAC	Off-Policy	Includes entropy regularization for exploration	Continuous

3.1 CQL/SAC algorithm

SAC is a DRL algorithm built on the Actor-Critic framework and operates as an off-policy method. It is designed to overcome the stability and efficiency limitations present in earlier approaches. SAC utilizes the maximum entropy reinforcement learning paradigm, where the actor seeks to maximize both the expected reward and exploration by maximizing entropy [31].

By employing off-policy updates, SAC allows for faster learning and improved sample efficiency through experience replay, unlike on-policy methods like Proximal Policy Optimization (PPO), which require fresh data for each gradient update. In contrast, off-policy methods reuse past experiences, significantly enhancing learning efficiency. Unlike PPO and DDPG, SAC employs twin Q-networks alongside a separate Actor network, with entropy tuning mechanisms to improve both stability and convergence.

The integration of CQL promotes more stable learning by addressing the issue of Q-value overestimation, which is especially useful when dealing with offline or suboptimal data scenarios [13]. The advantage of combining Conservative Q-Learning (CQL) with Soft Actor-Critic (SAC) lies in the synergy between robust exploration and conservative value estimation. SAC excels at learning effective policies through its entropy-regularized framework, which promotes balanced exploration and exploitation. However, SAC can sometimes overestimate Q-values, especially in data-limited or offline scenarios. By introducing a conservative penalty, CQL mitigates this issue, resulting in improved training stability and policy robustness.

Compared to PPO and Twin Delayed Deep Deterministic Policy Gradient (TD3), the CQL/SAC model offers advantages in both training time and efficiency. While SAC requires more training time than PPO, it achieves more stable learning, and it converges faster than TD3 due to its entropy-based exploration [8], [32, 33]. Although CQL slightly increases training time, it significantly enhances learning stability [13]. Overall, CQL/SAC is more sample-efficient than PPO and better suited for real-world deployment than TD3, particularly in high-dimensional tasks and offline learning scenarios.

This CQL/SAC hybrid method leads to safer and more efficient decision-making, particularly in dynamic, continuous control tasks where real-world uncertainties and limited data are common challenges [25]. RL objective for SAC algorithm is:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[(r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot\,|s_t))] \tag{1}$$

where $\mathcal{H}$ is the entropy:

$$\mathcal{H}\pi(\cdot\,|s_t) = \mathbb{E}[-\log(f_\pi(\cdot\,|s_t))] \tag{2}$$

with expectation operator $\mathbb{E}$, which denotes the averaging over all state-action pairs sampled from the trajectory distribution. A temperature parameter α is controlling the balance between exploration (higher entropy) and exploitation (higher reward). SAC can tune α automatically during training. Ultimately, this objective function aims to optimize the cumulative expected reward while simultaneously promoting exploration by maximizing policy entropy.

SAC follows an actor-critic architecture, where the actor explicitly models the policy, while the critic is solely responsible for guiding the actor's improvement. The critic's role is confined to the training phase, without directly influencing action selection during execution.

Two separate critic networks Q1 and Q2 are trained to minimize the Bellman error using the target Q value:

$$\hat{Q}_{\hat{\theta}_1,\hat{\theta}_2}(s_{t+1},a_{t+1}) = r_t + \gamma \mathbb{E}_{(s_{t+1}\sim D, a_{t+1}\sim\pi_\phi(.|s_{t+1}))}[\hat{Q}_{min} - \alpha \log(\pi_\phi((a_{t+1}|s_{t+1}))] \quad (3)$$

where the actor network $\pi_\theta(a|s)$ is trained to maximize the expected return while encouraging exploration using the entropy regularization term.

The target Q value incorporates the minimum of the two Q-values (to mitigate overestimation bias) and the entropy of the policy:

$$\hat{Q}_{min}(s_{t+1},a_{t+1}) = \min[\hat{Q}_{\hat{\theta}_1}(s_{t+1},a_{t+1}), \hat{Q}_{\hat{\theta}_2}(s_{t+1},a_{t+1})] \quad (4)$$

For each critic network, the Q-loss is computed as the mean squared error between the predicted Q-value and the target Q value:

$$J_Q(\theta_i) = \frac{1}{2}\mathbb{E}_{(s_t,a_t\sim D)}[\left(\hat{Q}_{\hat{\theta}_1,\hat{\theta}_2}(s_{t+1},a_{t+1}) - \hat{Q}_{\theta_1}(s_t,a_t)\right)^2] \quad (5)$$

The target networks are slowly updated copies of the critic networks. The target network does not have its own loss function; it serves to provide stable target values for training the critic network.

The policy-loss for actor-network is defined as:

$$J_\pi(\phi) = \mathbb{E}_{(s_t\sim D, a_t\sim\pi_\phi(.|s_t))}[\alpha \log(\pi_\phi(a_t|\, s_t)) - \min[(Q_{\theta_1}(s_t,a_t^\pi), Q_{\theta_2}(s_t,a_t^\pi)]] \quad (6)$$

A schematic view of SAC algorithm is presented in Fig. 1.

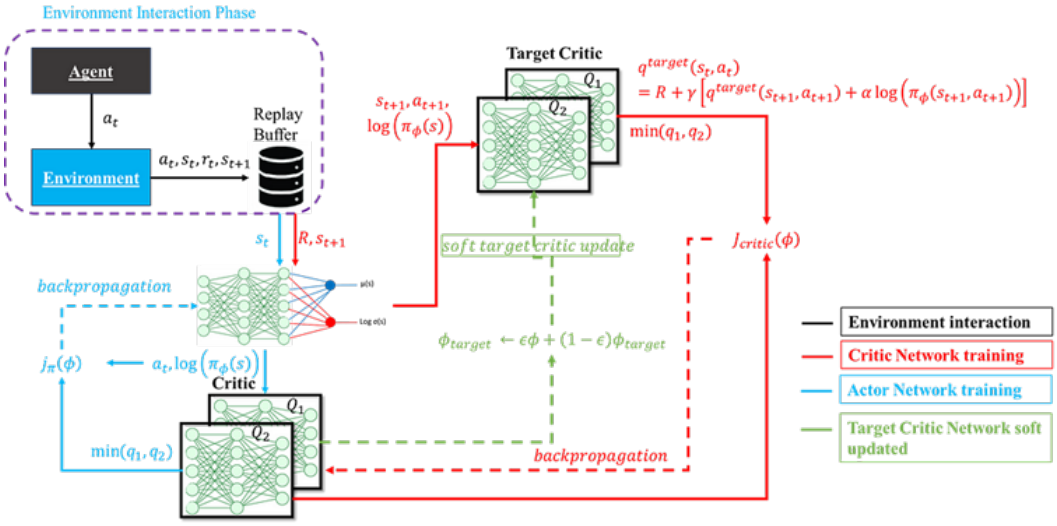

Fig. 1 A schematic view of the SAC algorithm [31]

In offline reinforcement learning (offline RL), the non-Lagrange version of the Conservative Q-Learning (CQL) method is utilized, as proposed in [13]. This approach is beneficial because it seamlessly integrates with existing continuous RL algorithms like Soft Actor-Critic (SAC) by introducing a regularization loss. By incorporating the CQL loss term into Eq. 5, the total Q-loss is expressed as:

$$J_Q^{total}(\theta_i) = J_Q(\theta_i) + \alpha_{cql}\mathbb{E}_{(s_t\sim D)}[log \sum_{a_t} exp(Q_{\theta_i}(s_t,a_t) - \mathbb{E}_{(a_t\sim D)}[Q_{\theta_i}(s_t,a_t)]) \quad (7)$$

where α_{cql} determines the degree to which the CQL loss is applied to the Q-loss. This penalizes actions that significantly deviate from the existing dataset, ensuring that the learned policy remains conservative in terms of exploration.

SAC algorithm is highly effective in continuous action spaces and complex tasks that require real-time adaptation, which is critical for industrial applications [9]. Due to these advantages, SAC can significantly contribute to the automation and optimization of pick-and-place operations in industry, making robotics more efficient and adaptable in dynamic production environments [8].

4. System description

The pick-and-place task requires a robotic arm to detect, grasp, and relocate an object to a target position efficiently. Key challenges include accurate perception using sensors (e.g., cameras, depth sensors), motion planning to compute smooth and collision-free trajectories for object grasping and placement, grasp stability to prevent object slippage, and dynamic adaptation to object variations.

This study focuses on enhancing the performance of RL agents in robotic pick-and-place tasks. To initiate this process, the robot interprets human gestures to trigger the intended pick-and-place actions. The block diagram of the system is presented in Fig. 2.

The cobot (myCobot320 by Elephant Robotics) is trained to perform pick-and-place tasks separately. The pick range is 10 cm × 15 cm, and the place range is identical but mirrored along the y-axis. The object's initial position is within the pick range, while the place position is randomly assigned within the place range. The trained policies are then combined to execute tasks sequentially in both simulation (PyBullet) and real-world scenarios. For real-world object detection, an Astra Pro 2 camera uses HSV color space, with tests conducted on green objects of 2 × 2.5 cm and 5 × 5 cm. Additionally, human collaboration is incorporated, allowing the robot to pause and resume operation based on hand state recognition using Google's MediaPipe library.

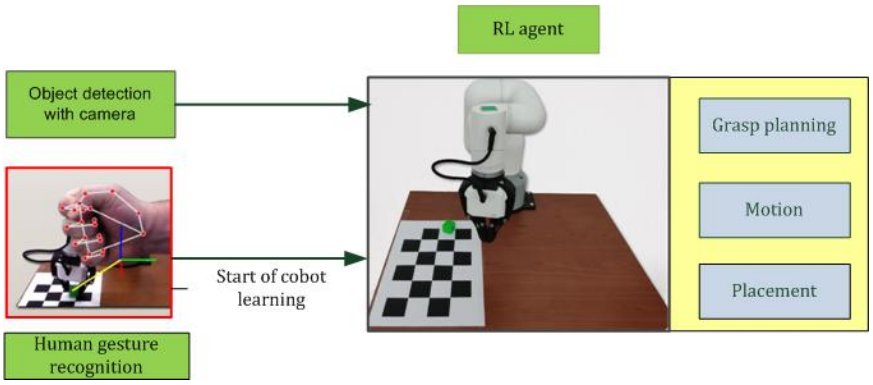

Fig. 2 A schematic view of Pick-and-place task in human-robot collaboration setting

4.1 Simulated and real-world platform

Robotic systems, with their high-dimensional and continuous action spaces, require extensive training for agents to develop optimal policies. However, online training can be expensive due to factors like the need for human oversight, potential risks of robot wear and damage, and limited access to training robots, making simulation-based training a practical alternative. Despite its benefits, simulation often suffers from low accuracy when applied to real-world tasks. Transferring policies from simulation to real-world robotics faces several challenges due to the sim-to-real gap, including [30, 34]:

- Model Inaccuracies: Simulations often oversimplify physics, neglecting factors like friction, sensor noise, and joint flexibility.
- Perception Discrepancies: Simulated sensors lack real-world noise and distortions, leading to poor generalization in tasks like vision-based navigation.
- Actuation Latency: Real actuators introduce delays and non-linearities not present in simulations.
- Data Distribution Shift: Policies trained in simulation may not generalize well to real-world variations.
- Safety and Robustness: Errors in real-world execution can damage hardware, and policies must handle unpredictable human interactions.
- Transfer Learning: Simulation-trained policies often need fine-tuning in the real world, which can be costly and inefficient.
- Lack of Real-World Data: Collecting real-world data for training RL policies is resource-intensive.

A major challenge in RL for robotics lies in bridging the Sim2Real gap, as transferring policies learned in simulators like Mujoco, PyBullet, or Gazebo, to real-world environments remains difficult. Addressing this challenge requires not only improving the robustness of RL algorithms but also developing more accurate and reliable simulators [35]. The development of standardized benchmarks and open-source environments, such as OpenAI Gym, Robosuite, and Isaac Gym, has accelerated research in RL for robotic manipulation. These advancements would provide suitable training environments for RL, enabling the creation of policies enabling them to perform effectively on physical robots.

This paper explores solutions to these challenges, focusing on leveraging RL to enable robots to autonomously learn and execute pick-and place tasks in both simulated (PyBullet environment), presented in Fig. 3 and real-world environments using myCobot320 by Elephant Robotics, presented in Fig. 4.

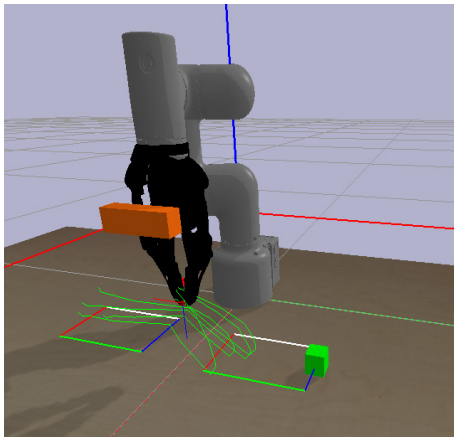

Fig. 3 Simulated environment (PyBullet)

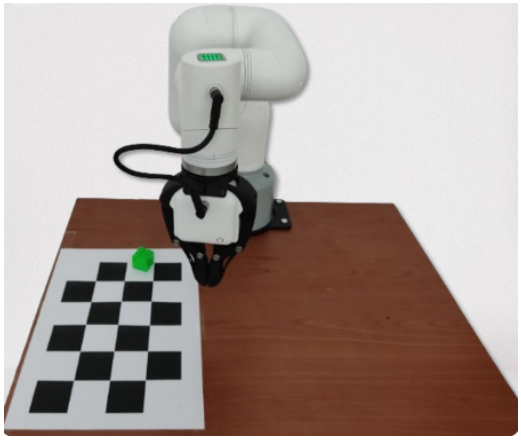

Fig. 4 Hardware settings for real environment

4.2 Human gesture recognition

Human collaboration is applied by stopping and resuming the work of real robot by recognizing the state of the hand by using Google's MediaPipe library. MediaPipe is an open-source framework developed by Google for building cross-platform, real-time perception pipelines, particularly in computer vision and machine learning applications. It utilizes graph-based processing, where modular components (calculators) efficiently handle tasks such as object detection, pose estimation, and gesture recognition.

Google's MediaPipe library can be utilized for gesture recognition by leveraging its pretrained models for hand tracking. It detects key points on the hand, such as the position of each finger and the palm, allowing for precise gesture recognition in real-time. By integrating MediaPipe into robotic systems, gestures can be used as inputs to control the robot's actions, enabling intuitive human-robot interaction.

MediaPipe utilizes GPU acceleration and model optimization strategies to enable high-performance inference on edge devices, such as smartphones and embedded systems. Its architecture is designed for fast prototyping and seamless deployment of AI-powered applications across different platforms, while minimizing computational demands.

In the event of failure (due to unexpected human interventions beyond simple gesture-based commands), the real robot will perform unusual actions based on observations that are currently unseen. Typically, in such situations, we stop the execution.

4.3 RL agent

In our study, the RL agent leverages key components to learn effective object manipulation within a human-robot collaboration setting. By observing the state, taking actions, and receiving rewards, the agent refines its policy to enhance task performance, such as successful object picking and placement.

The state encapsulates essential details about the environment and the robot's condition, including the end-effector's position and orientation, the real object's position relative to the robot's gripper, the target placement position, the computed desired object position based on the end-effector and real object positions, and the gripper's state (open/closed), resulting in a state size 22 in total.

The agent operates with continuous actions, such as executing small translations and rotations of the end-effector, adjusting joint angles for articulated control, and managing the gripper's state (open/close) to grasp or release objects, resulting in a total size of 7 actions.

The reward function is designed to encourage efficient task completion while discouraging suboptimal behaviours. Positive rewards are given for actions such as successfully grasping the object, moving it closer to the target, and correctly placing it in the target area. Negative rewards (penalties) are applied for actions like collisions with obstacles, dropping the object before reaching the target, and unnecessary movements that delay task completion.

In the context of the Soft Actor-Critic (SAC) framework and Conservative Q-learning (CQL) on top of the framework, the policy network plays a crucial role in determining the actions the robot should take based on the current state [8]. We use two Multi-Layer Perceptron (MLP), one for policy (actor) and the other for Q-value (critic, Q1 and Q2). The structure of both networks is the same except for the input/output layer. The structure of MLP consists of 3 fully connected hidden layers with size of 256 and batch size of 32. The policy network takes the state (observations) as input of size 22 and gives action as output of size 7, representing the actions the robot should take. The Q-value network takes the states and actions as input of size 29 and gives Q-value of size 1.

Overall, this neural network architecture within the CQL/SAC algorithm enables the robot to learn a policy that maps states to actions in a continuous space, continuously optimizing its performance over time through reinforcement learning.

Hyperparameters include the state dimension for the policy (actor) set to 22 and the action dimension set to 7. The total number of training epochs is 300 for the place task and 500 for the pick task. Each epoch consists of 1000 steps, where a step represents a single interaction where the agent takes an action, receives a reward, and moves to the next state.

Other hyperparameters include a maximum of 40 steps per episode during evaluation and a training batch size of 32. Evaluation occurs after each epoch (set to 1), using 16 episodes to compute evaluation metrics. An episode starts from an initial state and ends upon reaching a goal, a time limit, or failure.

CQL and SAC-specific hyperparameters include automatic entropy tuning (set to true), CQL alpha (1), CQL temperature (1), discount factor (0.99), target smoothing coefficient tau (0.005), and entropy regularization alpha (0.2). The number of random samples for CQL loss estimation is 10, with CQL version 3. Learning rates are 1e-4 for the policy network and automatic entropy tuning, and 3e-4 for the Q-value network.

4.4 Evaluation metrics

Evaluation metrics in robot manipulation are essential for assessing the performance, efficiency, and reliability of robotic systems in handling various tasks. These metrics help quantify how well a robot interacts with objects, executes movements, and completes assigned tasks under different conditions. Some key evaluation metrics include Trial Success Rate (TSR), Task Completion Time, Grasp Success Rate, Path Efficiency, etc. [36].

In this study, we used TSR and average cumulative reward over multiple episodes, which represents the percentage of multi-step tasks completed with 100 % success [37].

5. Experimental design, results and discussion

To evaluate CQL/SAC's performance in high-dimensional continuous control tasks, we train a cobot on a pick-and-place problem requiring accurate object detection, grasping, transportation, and precise placement. Figs. 5 and 6 show end-effector positions and object positions for four trajectories and ten trajectories, where green color represents successful trajectory/object placement and red color is for failed trajectory/object placement.

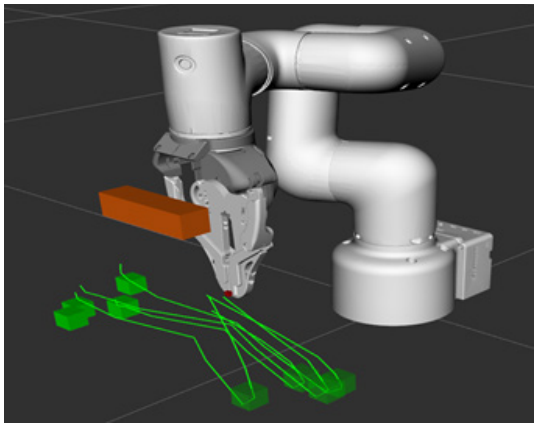

Fig. 5 End-effector positions for four trajectories

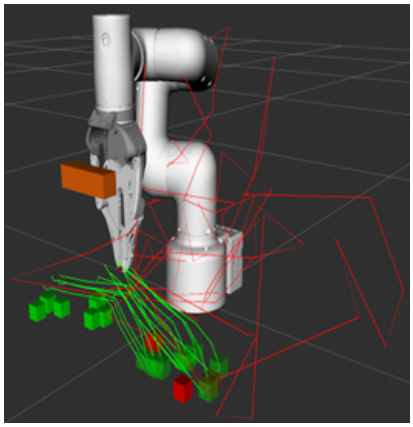

Fig. 6 End-effector positions for ten trajectories

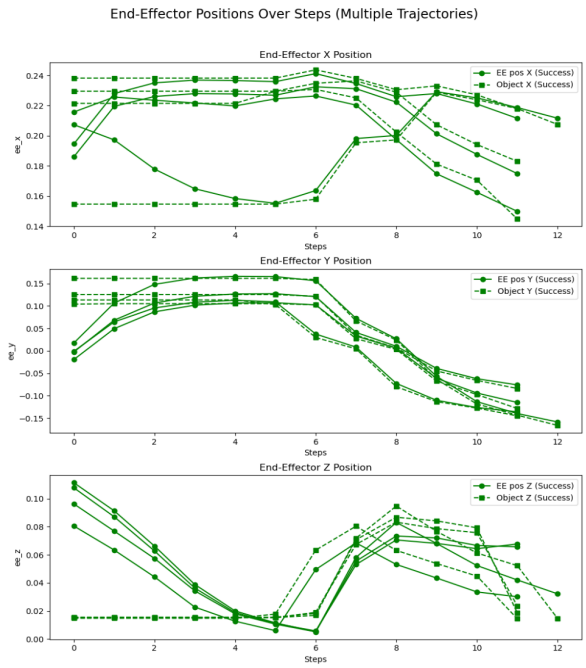

Fig. 7 Four successful trajectories, End-effector/object positions

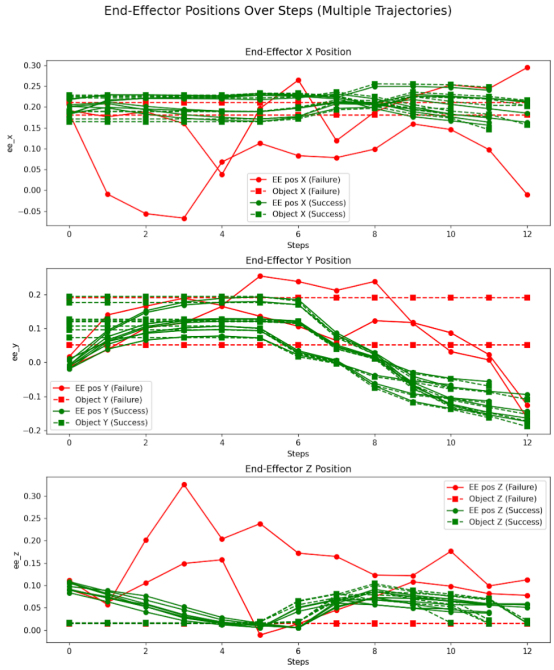

Fig. 8 Ten mixed (successful and failed) trajectories, End-effector/object positions

Fig. 7 and Fig. 8 presents trajectories of end-effector (dots) positions and object coordinates (squares) as functions of steps for 4 and 10 trajectories, respectively.

Fig. 9 and Fig. 10 illustrate the best performance metrics of picking and the placing tasks for evaluation and training. Neural networks with such trained weights are used for the real robot for pick and place tasks respectively. Pick task is trained for 500k steps and place task for 300k steps. From Fig. 9 can be seen that pick task gets 100 % success rate after 280k episodes, and mean rewards during training and evaluation remain stable. The similar results can be concluded from Fig. 10 for place task where after around 100k steps (1 epoch) we get the best performance regarding evaluation success rate and trained/evaluated reward remains similar.

Figs. 11 and 12 presents comparisons between different runs of pick and place tasks respectively. From Fig. 11 we started with 100k steps (run called `pick_0`) for training for pick task. We can see that it didn't get a successful eval success rate (obtained success rate is 0), even though other parameters were looking promising. The same happened for other runs (`pick_{1,2,3}`) respectively, where we can also see that RL is not deterministic. After expanding the training time from 100k to 500k we obtained the `pick_best` run.

Fig. 9 The best performances during training and evaluation for pick task

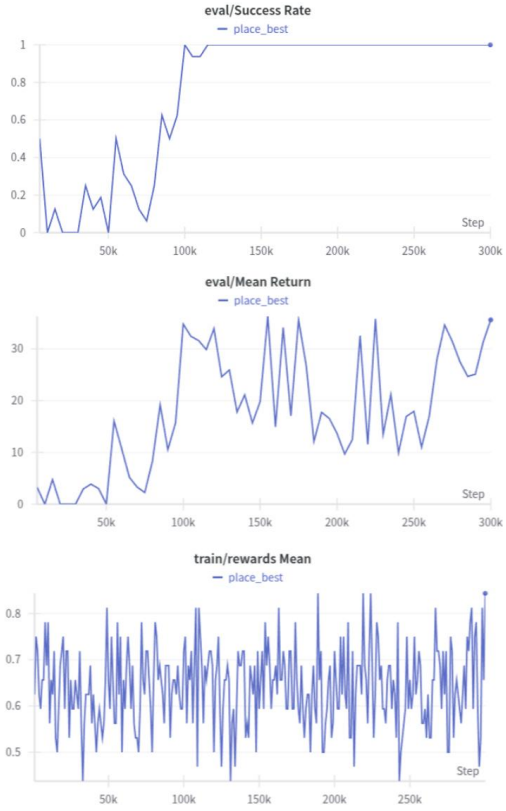

Fig. 10 The best performances during training and evaluation for place task

Fig. 11 The performance comparison between different runs for pick task

Fig. 12 The performance comparison between different runs for place task

Similarly, for place task, we varied the total number of steps for different runs, from 130k (`place_0` run), 250k steps (`place_1` run) and 300k `place_best` run, which can be seen on Fig. 11. We observed that although the highest success rate was achieved during evaluation, stability throughout the evaluation steps was not maintained. Therefore, we extended the training time to 300k episodes.

The success rate achieved was 100% in simulation and 80% on the real robot, evaluated over 100 episodes.

6. Conclusion

In this research, we demonstrated the effectiveness of combining CQL/SAC for robotic manipulation in human-robot collaboration settings. By leveraging the combination of CQL and SAC, the robot successfully learned optimal policies for executing intricate tasks, such as object picking and placement. CQL ensures more conservative value estimation, preventing suboptimal actions that could arise due to overestimation, while SAC facilitates efficient exploration and adaptation in uncertain environments. The results show that this hybrid approach significantly enhances the performance and reliability of robotic manipulation, enabling robots to effectively collaborate with humans in shared environments. We assessed this approach through cobot manipulation experiments, demonstrating the transferability of the learned policy from simulation to real-world settings without additional training. This was validated through real robot experiments, confirming that the integration of CQL with SAC enables safer and more reliable policy deployment in human-robot collaborative scenarios.

Currently, the framework is designed for pick-and-place tasks, but future work will focus on expanding its capabilities to a wider range of robotic manipulation tasks. Additionally, future research should explore multi-object detection, enabling the system to handle various shapes in a more dynamic industrial setting. Further advancements will also integrate diverse interaction modalities, including voice commands and text-based collaboration powered by Generative AI, such as Large Language Models, to enhance human-robot communication and adaptability.

Acknowledgement

This research is supported in scope of projects “Research and Development of Collaborative Intelligence in Service Robots for Industrial Applications”, funded by Federal Ministry of Education and Science, Bosnia and Herzegovina and “Smart factory enabled by artificial intelligence at the edges of a distributed network - a prerequisite for Industry 5.0”, funded by Ministry of education and science of TK, Bosnia and Herzegovina.

References

[1] Javernik, A., Buchmeister, B., Ojsteršek, R. (2022). Impact of Cobot parameters on the worker productivity: Optimization challenge, *Advances in Production Engineering & Management,* Vol. 17, No 4, 494-504, doi: 10.14743/apem2022.4.451.

[2] Banjanovic-Mehmedovic, L., Karabegovic, I., Jahic, J., Omercic, M. (2021). Optimal path planning of a disinfection mobile robot against COVID-19 in a ROS-based research platform, *Advances in Production Engineering & Management*, Vol. 16, No. 4, 405-417, doi: 10.14743/apem2021.4.409.

[3] Li, Y.C., Wang, X. (2024). Human-robot collaboration assembly line balancing considering cross-station tasks and the carbon emissions, *Advances in Production Engineering & Management,* Vol. 19, No 1, 31-45, doi: 10.14743/apem2024.1.491.

[4] Liu, Q., Liu, Z., Xiong, B., Xu, W., Liu, Y. (2021). Deep reinforcement learning-based safe interaction for industrial human-robot collaboration using intrinsic reward function, *Advanced Engineering Informatics,* Vol. 49, Article No. 101360, doi: 10.1016/j.aei.2021.101360.

[5] Peters, J., Lee, D.D., Kober, J., Nguyen-Tuong, D., Bagnell, J.A., Schaal, S. (2016). Robot learning, In: Siciliano, B., Khatib, O. (eds.), *Springer handbook of robotics,* Springer, Cham, Switzerland, 357-398, doi: 10.1007/978-3-319-32552-1_15.

[6] Xing, H.R. (2024). Optimizing human-machine systems in automated environments, *International Journal of Simulation Modelling*, Vol. 23, No. 4, 716-727, doi: 10.2507/IJSIMM23-4-CO19.

[7] Ojstersek, R., Javernik, A., Buchmeister, B. (2021). The impact of the collaborative workplace on the production system capacity: Simulation modelling vs. real-world application approach, *Advances in Production Engineering & Management*, Vol. 16, No. 4, 431-442, doi: 10.14743/apem2021.4.411.

[8] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, *ArXiv,* doi: 10.48550/arXiv.1801.01290.

[9] Gu, S., Holly, E., Lillicrap, T., Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, In: *Proceedings of 2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 3389-3396, doi: 10.1109/ICRA.2017.7989385.
[10] Karabegović, I., Banjanović-Mehmedović, L. (2021). *Service robots: Advances and applications*, Nova Science Publisher, New York, USA.
[11] Kroemer, O., Niekum, S., Konidaris, G. (2021). A review of robot learning for manipulation: Challenges, representations, and algorithms, *ArXiv,* doi: 10.48550/arXiv.1907.03146.
[12] Mahler, J., Matl, M. Satish, V., Danielczuk, M., DeRose, B., McKinley, S., Goldberg, K. (2019). Learning ambidextrous robot grasping policies, *Science Robotics,* Vol. 4, No. 26, Article Id. eaau4984, doi: 10.1126/scirobotics.aau4984.
[13] Kumar, A., Zhou, A., Tucker, G., Levine, S. (2020). Conservative Q-learning for offline reinforcement learning, *ArXiv,* doi: 10.48550/arXiv.2006.04779.
[14] Liu, Y., Wang, C., Zhao, C., Wu, H., Wei, Y. (2024). A soft actor-critic deep reinforcement-learning-based robot navigation method using LiDAR, *Remote Sensing*, Vol. 16, No. 12, Article No. 2072, doi: 10.3390/rs16122072.
[15] Kosaraju, D. (2024). Hierarchical reinforcement learning: Structuring decision-making for complex AI tasks, *International Journal of Science and Healthcare Research,* Vol. 7, No. 2, 485-491, doi: 10.52403/ijshr.20220468.
[16] Levine, S., Finn, C., Darrell, T., Abbeel, P. (2016). End-to-end training of deep visuomotor policies, *ArXiv,* doi: 10.48550/arXiv.1504.00702.
[17] Mohammed, M.Q., Chung, K.L., Chyi, C.S. (2020). Pick and place objects in a cluttered scene using deep reinforcement learning, *International Journal of Mechanical and Mechatronics Engineering,* Vol. 20, Vol. 4, 50–57.
[18] Han, D., Mulyana, B., Stankovic, V., Cheng, S. (2023). A survey on deep reinforcement learning algorithms for robotic manipulation, *Sensors*, Vol. 23. No. 7. Article No. 3762, doi: 10.3390/s23073762.
[19] Liu, D., Wang, Z., Lu, B., Cong, M., Yu, H., Zou, Q. (2020). A reinforcement learning-based framework for robot manipulation skill acquisition, *IEEE Access,* Vol. 8, 108429-108437, doi: 10.1109/ACCESS.2020.3001130.
[20] Al-Selwi, H.F., Aziz, A.A., Abas, F.S., Zyada, Z. (2021). Reinforcement learning for robotic applications with vision feedback, In: *Proceedings of the 2021 IEEE 17th International Colloquium on Signal Processing & Its Applications (CSPA)*, Langkawi, Malaysia, 81-85, doi: 10.1109/CSPA52141.2021.9377292.
[21] Yamada, J., Lee, Y., Salhotra, G., Pertsch, K., Pflueger, M., Sukhatme, G.S., Lim, J.J., Englert, P. (2020). Motion planner augmented reinforcement learning for robot manipulation in obstructed environments, *ArXiv*, doi: 10.48550/arXiv.2010.11940.
[22] Zhan, A., Zhao, P., Pinto, L., Abbeel, P., Laskin, M. (2020). A framework for efficient robotic manipulation, *ArXiv.*
[23] Kwon, G., Kim, B., Kwon, N.K. (2024). Reinforcement learning with task decomposition and task-specific reward system for automation of high-level tasks, *Biomimetics*, Vol. 9, No. 4, Article No. 196, doi: 10.3390/biomimetics9040196.
[24] Zhao, T., Wang, M., Zhao, Q., Zheng, X., Gao, H. (2023). A path-planning method based on improved soft actor-critic algorithm for mobile robots, *Biomimetics,* Vol. 8, No. 6, Article No. 481, doi: 10.3390/biomimetics8060481.
[25] Yadav, S.P., Nagar, R., Shah, S.V. (2024). Learning vision-based robotic manipulation tasks sequentially in offline reinforcement learning settings, *Robotica,* Vol. 42, No. 4, 1715-1730, doi:10.1017/S0263574724000389.
[26] Liu, A.Y., Yue, D.Z., Chen, J.L., Chen, H. (2024). Deep learning for intelligent production scheduling optimization, *International Journal of Simulation Modelling*, Vol. 23, No. 1, 172-183, doi: 10.2507/IJSIMM23-1-CO4.
[27] Wei, Z.H., Yan, L., Yan, X. (2024). Optimizing production with deep reinforcement learning, *International Journal of Simulation Modelling*, Vol. 23, No. 4, 692-703, doi: 10.2507/IJSIMM23-4-CO17.
[28] Wang, Y., Friderikos, V. (2020). A survey of deep learning for data caching in edge network, *Informatics,* Vol. 7. No. 4, Article No. 43, doi: 10.3390/informatics7040043.
[29] Liu, Z., Liu, Q., Xu, W., Wang, L., Zhou, Z. (2022). Robot learning towards smart robotic manufacturing: A review, *Robotics and Computer-Integrated Manufacturing,* Vol. 77, Article No. 102360, doi: 10.1016/j.rcim.2022.102360.
[30] Lobbezoo, A., Qian, Y., Kwon, H.-J. (2021). Reinforcement learning for pick and place operations in robotics: A survey, *Robotics,* Vol. 10, No. 3, Artice No. 105, doi: 10.3390/robotics10030105.
[31] Sharifi, N. Mathematical foundation underpinning reinforcement learning, from *https://ai.gopubby.com/mathematical-foundation-underpinning-reinforcement-learning-34b304890c33*, accessed January 3, 2025.
[32] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms, *ArXiv,* doi: 10.48550/arXiv.1707.06347.
[33] Fujimoto, S., Hoof, H., Meger, D. (2018). Addressing function approximation error in actor-critic methods, *ArXiv,* doi: 10.48550/arXiv.1802.09477.
[34] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world, In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* Vancouver, Canada,, 23-30, doi: 10.1109/IROS.2017.8202133.
[35] Zhu, W., Guo, X., Owaki, D., Kutsuzawa, K., Hayashibe, M. (2023). A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 34, No. 7, 3444-3459, doi: 10.1109/TNNLS.2021.3112718.
[36] Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,*The International Journal of Robotics Research,* Vol. 37, No. 4-5, 421-436, doi: 10.1177/0278364917710318.
[37] Hundt, A., Killeen, B., Greene, N., Wu, H., Kwon, H., Paxton, C., Hager, G.D. (2020). "Good Robot!": Efficient reinforcement learning for multi-step visual tasks with sim-to-real transfer, *IEEE Robotics and Automation Letters*, Vol. 5, No. 4, 6724-6731, doi: 10.1109/LRA.2020.3015448.