ACS ENGINEERING Au
AN OPEN ACCESS JOURNAL OF THE AMERICAN CHEMICAL SOCIETY

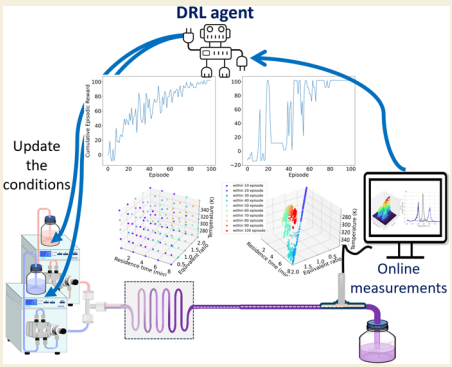
pubs.acs.org/engineeringau
Article

Deep Reinforcement Learning-Based Self-Optimization of Flow Chemistry

Ashish Yewale, Yihui Yang, Neda Nazemifard, Charles D. Papageorgiou, Chris D. Rielly, and Brahim Benyahia*

Cite This: ACS Eng. Au 2025, 5, 247−266
Read Online

ABSTRACT: The development of effective synthetic pathways is critical in many industrial sectors. The growing adoption of flow chemistry has opened new opportunities for more cost-effective and environmentally friendly manufacturing technologies. However, the development of effective flow chemistry processes is still hampered by labor- and experiment-intensive methodologies and poor or suboptimal performance. In this context, integrating advanced machine learning strategies into chemical process optimization can significantly reduce experimental burdens and enhance overall efficiency. This paper demonstrates the capabilities of deep reinforcement learning (DRL) as an effective self-optimization strategy for imine synthesis in flow, a key building block in many compounds such as pharmaceuticals and heterocyclic products. A deep deterministic policy gradient (DDPG) agent was designed to iteratively interact with the environment, the flow reactor, and learn how to deliver optimal operating conditions. A mathematical model of the reactor was developed based on new experimental data to train the agent and evaluate alternative self-optimization strategies. To optimize the DDPG agent's training performance, different hyperparameter tuning methods were investigated and compared, including trial-and-error and Bayesian optimization. Most importantly, a novel adaptive dynamic hyperparameter tuning was implemented to further enhance the training performance and optimization outcome of the agent. The performance of the proposed DRL strategy was compared against state-of-the-art gradient-free methods, namely SnobFit and Nelder−Mead. Finally, the outcomes of the different self-optimization strategies were tested experimentally. It was shown that the proposed DDPG agent has superior performance compared to its self-optimization counterparts. It offered better tracking of the global solution and reduced the number of required experiments by approximately 50 and 75% compared to Nelder−Mead and SnobFit, respectively. These findings hold significant promise for the chemical engineering community, offering a robust, efficient, and sustainable approach to optimizing flow chemistry processes and paving the way for broader integration of data-driven methods in process design and operation.



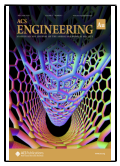
KEYWORDS: *flow chemistry, self-optimization, deep reinforcement learning, deep deterministic policy gradient, adaptive hyperparameter tuning, Bayesian optimization*

1. INTRODUCTION

Chemical reactions are essential steps to synthesis a variety of products, intermediates, additives, etc., relevant to several key industries such as the pharmaceutical, agrochemical, and petrochemical sectors. The total production costs are largely determined by the performance of the synthetic steps and inherent purification technologies, besides product quality and safety, environmental performance, and overall economic viability. The performance of a reaction is commonly captured by yield and selectivity which are influenced by a complex interplay between various factors such as catalysts, solvents, substrate concentrations, temperature, and reaction technologies which must be optimized to deliver consistent product quality. Traditional optimization approaches are labor-intensive, time-consuming, and very costly, often relying on "one variable at a time" (OVAT) experimentation or human intuition. The development of effective synthetic pathways for active pharmaceutical ingredients in pharma is a painstaking and very costly process.[1,2] Poor reaction pathways may impact not only costs due to poor yield but also safety of the drug. The presence of impurities or side products makes the purification and isolation steps more challenging[3,4] and may in many cases jeopardize the clinical trials and, if not addressed, may result in product withdrawals, leading to significant financial and reputational damage.[5−7]

To address these challenges, it is important to design more effective synthetic strategies and technologies and develop

Received: January 9, 2025
Revised: April 26, 2025
Accepted: April 29, 2025
Published: May 13, 2025



smarter and more systematic optimization strategies to avoid pitfalls and limitations of the traditional optimization methods and human intuitions. Continuous manufacturing and flow chemistry are increasingly adopted in the pharmaceutical industry to deliver more cost-effective, eco-friendly, and resilient alternatives to the traditional batch processing.[8,9] Most importantly, the interplay between flow chemistry, advanced real-time process analytical technologies, and self-optimization can open a new era for effective development, design, and operation based on plug-and-play strategies. To deliver these objectives, it is critical to develop the next generation algorithms and methodologies specifically to address this new class of real-time optimization problems. Moreover, Industry 5.0, leveraging advancements in artificial intelligence, big data analytics, and autonomous robotics, is emerging as a transformative approach to enhancing efficiency and minimizing risks in chemical reactions.[10−13] Recently, several machine learning (ML) methods have been successfully implemented to analyze more effectively both historical and real-time experimental data, delivering unique insights and allowing more reliable decision-making and effective design and operation strategies.[14,15]

Additionally, high-throughput experimental platforms for automated reaction optimization are increasingly adopted to maximize experimental screening.[16,17] These methods and technologies may offer enhanced and systematic exploration of the reaction recipes and conditions (e.g., reagents, solvents, catalysts). However, they are often restricted to batch processes and can be extremely costly due to, due their intensive experimentation nature, particularly when implemented based on trial-and-error, instead of rigorous optimization strategies. The combination of flow chemistry and advanced optimization and control algorithms is opening new avenues for more effective real-time monitoring and adjustment of the reaction conditions. Over the past few years, researchers have implemented several traditional optimization methods such as Nelder−Mead simplex algorithm,[18,19] stable noisy optimization by branch and fit (SnobFit) algorithm[20−22] and mixed integer nonlinear programming (MINLP).[23] Additionally, Deep Neutral Networks (DNNs) were used to optimize residence time, temperature, and catalyst loading in the Pd-catalyzed Suzuki-Miyaura reaction, predicting yield and identifying optimal reaction conditions by exploring the available reaction space.[24] However, these approaches can be data extensive and may require large sets of expensive experiments.[25] Among several other ML methods, Bayesian optimization has gained increased popularity.[26−28] The method uses kernel density estimators to explore the decision variable space and, by balancing exploitation and exploration, optimally identifies the next experimentation/sampling locations. Shields et al.[28] implemented experimental design based on Bayesian optimization featuring several strategies, such as density functional theory, cheminformatic, and binary one-hot encoded, aimed to find the optimal reaction conditions and functional groups of reactants. The authors proposed a new encoding of the reactions by concatenating molecular descriptors for each chemical component and continuous variables (temperature, reaction time, and concentration). Schweidtmann et al.[29] developed a new package, Thomson Sampling Efficient Multi-Objective (TS-EMO), to support initial sampling methods and Bayesian optimization with Gaussian processing surrogate models. The tool was used to find optimal reaction conditions for two different organic reactions, maximizing space-time yield and minimizing the E-factor. The applicability of Bayesian optimization is extended to different areas such as multistep reactions,[30,31] multi-objective problems,[32,33] and multitask Bayesian optimizations.[34,35] Furthermore, Bayesian optimization supported with Gryffin can operate with both continuous and discrete variables.[36,37] Overall, data-driven approaches may help minimize human bias and accelerate the optimization and development processes, enhancing precision and efficiency in synthetic route development. Despite these advantages, ML models, in particular, may face challenges when addressing new or unknown conditions due to their reliance on labeled data and pattern recognition techniques. These models also require substantial amounts of data for effective training, and interpreting their results can be complex. Alternatively, reaction optimization can be effectively addressed using reinforcement learning (RL). As a subset of machine learning, RL focuses on optimizing action policies to maximize or minimize rewards based on continuous interactions with the environment. Deep reinforcement learning (DRL), which employs DNNs to estimate unknown functions of state and action spaces (such as policy or value functions), has proven powerful in many applications. By deriving estimations from a small sample of observations, DRL addresses the challenges of high-dimensional state spaces and solves increasingly complex problems.[38] DRL's impactful applications span numerous fields such as vision-based navigation and manipulation in robotics,[39,40] autonomous navigation in self-driving cars,[41,42] and optimization of energy distribution in smart grids and buildings.[43−45] DRL also revealed pivotal in the control of crystallization processes for pharmaceuticals,[46,47] and optimization of the production lines in the manufacturing industries.[48,49] Most importantly, DRL has shown capabilities to effectively manage complex chemical processes by dynamically adjusting conditions in real-time. For instance, DRL was effectively applied to reaction control in polymerization processes by integrating a Long Short-Term Memory (LSTM) network to represent surrogate modeling and proximal policy optimization for decision making.[50,51] In addition, Zhou and coauthors[52] used a recurrent neural network as a policy function for decision making to optimize chemical reactions. Due to the time-consuming nature of evaluating these reactions, their approach involved initially training the model on simulated reaction data using a mixture of Gaussian density functions. This probabilistic approach models a data set as a weighted combination of multiple Gaussian distributions, each characterized by its own mean and covariance. It enhanced performance was achieved by continuously approximating the response surface of the chemical reaction systems, capturing their complex behavior with more reliable models. This pretraining strategy was designed to approximate complex reaction decision spaces and effectively handle multiple local minima. To improve exploration and avoid local optima traps, the authors introduced a randomized exploration strategy. This strategy uses a stochastic recurrent neural network to generate random decision choices, enhancing the algorithm's ability to explore different experimental conditions more effectively. Their method achieved optimal reaction conditions in just 40 episodes, delivering a significant improvement over the traditional methods such as covariance matrix adaptation, evolution strategy, and OVAT approaches. More recently, Neumann and Palkovits[53] demonstrated the capability of DRL

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

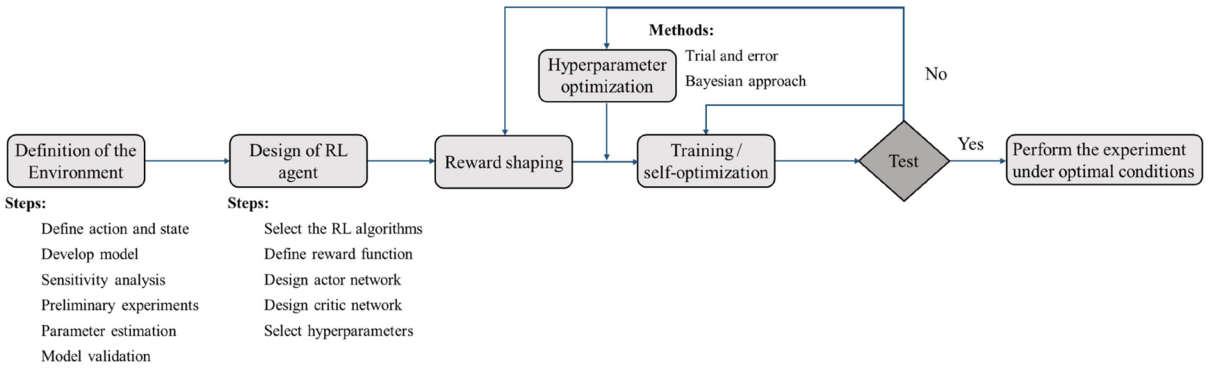


Figure 1. Workflow of the proposed deep reinforcement learning-based self-optimization method.

in optimizing methane oxidation in a plug flow reactor, highlighting the potential of self-optimization in flow chemistry. However, their approach relies on fixed hyperparameters and predefined target conditions, which can be particularly limiting in chemical reaction optimization. In these systems, even slight variations in reaction parameters can significantly alter outcomes, and a rigid framework may fail to explore the full parameter space.[54] Additionally, as seen in continuous control tasks,[55] the sample inefficiency and high computational cost associated with simulating numerous reaction iterations pose substantial challenges. Moreover, analogous issues have been observed in related fields such as de novo drug design[56] and in automated chemical synthesis,[57] where balancing exploration with computational expense is critical. Future work could benefit from integrating adaptive hyperparameter tuning and more robust exploration strategies to better capture the dynamic behavior of chemical systems, thereby improving the discovery of true optimal reaction conditions.

Addressing these limitations could enhance DRL's application in reaction optimization, including incorporating more complex scenarios and benchmarking against traditional methods.

In this paper, we propose addressing the challenges discussed earlier by introducing DRL, allowing more effective optimization of the reaction conditions of continuous chemical reactions. DRL combines the strengths of deep learning and reinforcement learning, allowing the development of strategies that can handle high-dimensional state and action spaces, which is critical in the complex environment of continuous action spaces. Deep Deterministic Policy Gradient (DDPG), a model-free, off-policy actor-critic method, is well-suited for such environments, as it allows agents to navigate an infinite number of possible actions efficiently.[58,59] The model-free nature of DDPG facilitates the self-optimization of reaction conditions by enabling autonomous decision-making, where RL agents select actions that maximize expected rewards without human inputs. The proposed DDPG agent uses deep learning capabilities to extract meaningful features from complex data and consolidates it by the reinforcement learning power to learn optimal policies through interaction with the environment. It simplifies the learning process and enhances sample efficiency by focusing on learning a deterministic policy that directly maps states to actions, rather than learning a distribution over actions (stochastic approach) that requires more exploration and data. The performance of the RL agent is further enhanced by optimizing hyperparameters and set up of the episodes and actions. The proposed DRL self-optimization strategy is validated using a case study relevant to flow chemistry, offering a continuous, adaptive, and scalable approach that enhances efficiency, resource utilization, and cost-effectiveness. The imine synthesis, which exhibits side reactions, is used to show the capabilities and performance of the proposed approach.

Additionally, the DDPG agent's performance is compared against the state-of-the-art gradient free optimization techniques, highlighting the effectiveness of DDPG in achieving superior optimization outcomes.

2. METHOD

The proposed general methodology to address the self-optimization problem is outlined in the workflow presented in Figure 1. The process begins with the definition of the environment, including decision variables, states, and outputs, which sets the foundation for the subsequent steps. An RL agent is then designed to interact with the environment, followed by the formulation of a reward function to steer the agent toward achieving the desired behaviors. The agent undergoes a training phase involving the hyperparameter optimization to identify and fine-tune key hyperparameters. After training, the agent is tested to evaluate its performance and robustness. Finally, the experiment is conducted under optimal conditions to validate the effectiveness of the trained agent.

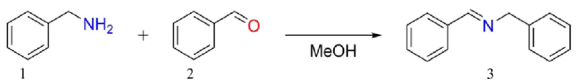
2.1. Environment

The environment represents the system of interest which can be influenced by a set of manipulated or decision variables which are captured by the set of actions initiated by the agent at a given time. The response of the system to a set of actions is captured by the states and outputs which allow the evaluation of instantaneous and cumulative rewards. In this study, the environment is represented by a mathematical model which allows more cost-effective training before validation or deployment in the real system. The imine synthesis in a tubular reactor (continuous) is considered as a case study to validate the proposed approach. The Imines and their derivatives are considered as key intermediates for the synthesis of nitrogen heterocycles and versatile pharmacophores.[60] In addition, they are used in several important classes of compounds with herbicidal, anticancer, and antimycotic activities.[61] The imine synthesis considered in this study is a condensation reaction between benzylamine **1** (ReagentPlus, ≥99%, Sigma-Aldrich, United Kingdom) and benzaldehyde **2** (ReagentPlus, 99%, Sigma-Aldrich, United Kingdom) in the presence of methanol (for synthesis, Fisher Scientific, United

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

Kingdom), to produce imine *N*-benzylidenebenzylamine **3** as shown in Scheme 1.

Scheme 1. Imine Synthesis[a]



[a]Benzaldehyde **1**, benzylamine·**2**, and *N*-benzylidenebenzylamine **3**.

2.1.1. Mathematical Model. The mathematical model of the imine synthesis system in flow is used as the environment for the RL-based self-optimization. The mathematical model was constructed using the following key assumptions:

- The reaction is homogeneous.
- No axial dispersion (plug flow reactor).
- The feeding temperatures (T_f) and initial reactor temperature (T_0) are equal.
- The overall heat transfer coefficient is constant over the length of reactor and during the reaction.
- The density of the reaction mixture (ρ) is constant.
- Chemical reactions are the only processes that can produce heat, and the cooling system is the only way for heat removal (no heat loss).

The residence time (Res_t) of the flow reactor can be expressed as a function of the total volumetric flow rate (Q) and the reactor volume (V) by eq 1:

$$\mathrm{Res}_t = \frac{V}{Q} \tag{1}$$

The concentration of each reactant is determined by its individual flow rate relative to the total feed rate (Q) and concentration of the stock solution of the reactant. The total flow rate (Q) is the sum of the individual flow rates of all reactants.

The concentration of each reactant (c_{jf}) in the feed stream is described as

$$c_{jf} = \frac{Q_j}{Q} \times C_{sj} \tag{2}$$

where Q_j is the individual flow rate and C_{sj} is the concentration of the stock solution of the reactant, j, respectively.

The mass balance for the reactants and product in the tubular reactor are shown in the partial differential equation below, which depicts the spatial-temporal changes in the concentration of the different species.

$$\frac{\partial c_i}{\partial t} = -v_z \frac{\partial c_i}{\partial z} \pm r_i \tag{3}$$

where c_i represents the concentration of the reactant or product i, v_z is the velocity in the z direction, and r is rate of reaction.

The general form of the rate law (reaction rate) for a bimolecular irreversible reaction is given by

$$r = kc_1c_2 \tag{4}$$

The reaction rate constant (k) follows an Arrhenius law, expressed as

$$k = k_{\mathrm{ref}} \exp\left(\frac{-E_a}{R} \times \left(\frac{1}{T_f} - \frac{1}{T_{\mathrm{ref}}}\right)\right) \tag{5}$$

Here k_{ref} represents the reference reaction rate coefficient, E_a is the activation energy, R is the universal gas constant, and T_f and T_{ref} represent the feed temperature and reference temperature, respectively,

The energy equation accounts for the heat of reaction, diffusive flux, and heat exchange between the reaction side and the coolant. The energy balance equation is given by eq 6.

$$\rho C_p \frac{\partial T}{\partial t} = -v_z \rho C_p \frac{\partial T}{\partial z} \pm r \times \Delta H_{\mathrm{react}} + \mathrm{UA}(T_c - T) \tag{6}$$

$$C_p = \sum_i x_i C_{pi} \tag{7}$$

where C_p in the above reaction refers to the average specific heat capacity and $C_{p,i}$ specific heat capacity of i^{th} species, x_i is the mole fraction of each species, T and T_c represent temperature inside the tubular reactor and of the coolant, respectively. UA is the overall heat transfer coefficient, and $\Delta H_{\mathrm{react}}$ is the reaction enthalpy. All relevant parameter values can be found in Table 1.

Table 1. Model Parameters, Physical Properties, and Initial Conditions

variables (unit)	values
length, L (m)	0.15
volume, V (m^3)	1.32×10^{-6}
UA (W/K)	100
$\Delta H_{\mathrm{react}}$ (kJ/mol)	160
$C_{p,1}$ (J/mol·K)	172
$C_{p,2}$ (J/mol·K)	207.2
$C_{p,3}$ (J/mol·K)	165.7
$C_{p,\mathrm{solvent}}$ (J/mol·K)	81.2
ρ (kg/m^3)	800
benzylamine stock concentration, C_{s1} (mol L^{-1})	4
benzaldehyde stock concentration, C_{s2} (mol L^{-1})	4
gas constant, R (J K^{-1} mol^{-1})	8.314
residence time bounds, Res_t (min)	[0.5, 8]
equivalent ratio bounds, $\mathrm{ER}_{1/2}$	[0.1, 2]
temperature bounds, T (K)	[278, 330]

2.1.2. Experimental Setup and Analytical Technologies. The experimental setup was designed to estimate kinetic parameters and validate the reaction mathematical model using inline spectroscopic measurements. The reaction was conducted in a microreactor system composed of coiled 1/16-in. stainless steel tubes. The reactor setup allowed for adjustable residence times between 0.5 and 8 min, ensuring near plug flow behavior throughout the experiments. Starting materials were delivered with high precision using an AZURA P 4.1s compact pump (Knauer, Germany). A detailed diagram of the setup is shown in Figure 2.

Real-time reaction monitoring was achieved using an inline Raman spectrometer (Raman Rxn2 analyzer, Kaiser Optical Systems, Inc.). Raman spectra were recorded in the 500−1900 cm^{-1} range with a spectral resolution of 4 cm^{-1}. Spectra were acquired at 30-s intervals, with each spectrum accumulated over a 5-s duration. Key Raman peaks were identified at 1596 cm^{-1} (ring C═C stretching), 1639 cm^{-1} (C═N stretching), and 1684 cm^{-1} (C═O stretching). The recorded spectra revealed significant intensity changes during the reaction course. The C═O stretching peak at 1684 cm^{-1} decreased steadily and disappeared completely, indicating full conversion

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

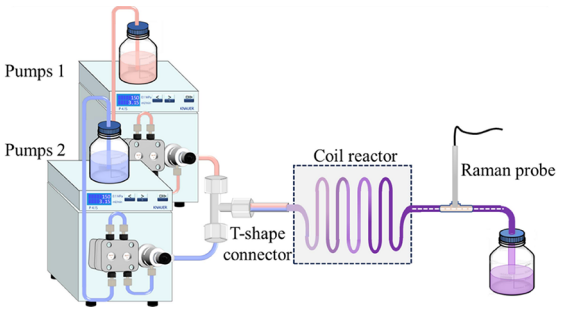


Figure 2. Reaction experimental set up for the proposed imine synthesis reaction in flow.

of the starting material. In contrast, the C═N stretching peak and the ring C═C stretching peak increased in intensity, signifying the formation of the imine product and the associated π-electron conjugation enhancement. These inline measurements provide valuable time-resolved data to support kinetic parameter estimation and validation the proposed reaction model.

2.2. Reinforcement Learning Agent

Reinforcement learning (RL) agents are designed to interact with their environment and autonomously choose the optimal behavior. The agent's selection of a set of specific actions (i.e., reaction conditions) to be implemented in the environment is informed by the rewards received based on the previous actions and current circumstances captured by the states of the environment. Consequently, the agent receives a new reward or penalty which after combination with the prior knowledge informs the new sets of actions to be taken. Each sequential step involves interactions between the agent and the environment through state, action, and rewards (r_t).[62] Subsequently, throughout a sequence of recurrent episodes, the agent is instructed to maximize cumulative rewards (G_t)

$$G_t = \sum_{t=0}^{\infty} \gamma^t r_t \tag{8}$$

where γ is the discount factor and signifies the importance of the future reward.

The primary objective of an RL algorithm is to identify and implement the best sequence of actions over an episode to maximize the accumulated reward. Thus, agents attempt to determine the optimal policy to maximize the expected return given by a state-value function (V-function) or state-action pair value function (Q-function).

Traditional methods for estimating value functions or policies in reinforcement learning (RL) typically rely on tabular approaches or linear function approximations. These methods work well for simple environments with discrete and small state-action spaces, as seen in techniques like Q-learning or temporal difference (TD) learning.[63] However, when faced with complex or high-dimensional problems, such as those encountered in chemical reactions, these traditional approaches become impractical. The challenge arises from the difficulty of manual feature engineering, which involves selecting and designing relevant features from raw data. This process can be time-consuming, error-prone, and may lead to instability or divergence in the learning process, further complicating the model's ability to generalize to new situations.

To address these limitations, deep learning methods were integrated with RL, leveraging deep neural networks (DNNs) to more effectively approximate value functions, policies, and models within the RL framework. DRL offers several key advantages over traditional RL.[64,65] It excels in handling high-dimensional state and action spaces, thanks to the capacity of deep neural networks to represent intricate functions. DRL also automates feature learning from raw data, reducing the need for manual feature engineering and improving adaptability to complex environments. This capability enhances scalability and generalization, allowing DRL to perform well in diverse and previously unseen scenarios. Furthermore, DRL enables end-to-end learning, integrating perception and decision-making into a unified process, which improves overall performance and flexibility in tackling sophisticated problems.[59]

A deep deterministic policy gradient (DDPG), which is one of the family of DRL algorithm, is proposed to address the challenges associated with environments exhibiting continuous action spaces and high-dimensional and complex actions and achieve a nuanced exploration-exploitation trade-off.[58] The proposed DDPG integrates value-based techniques like Q-learning with policy-based approaches such as policy gradient methods to deliver a robust framework for concurrently learning complicated behaviors. This integration allows DDPG to leverage the stability and robustness of value-based methods while benefiting from the flexibility and adaptability of policy-based approaches, enhancing learning efficiency in complex and continuous action environments.

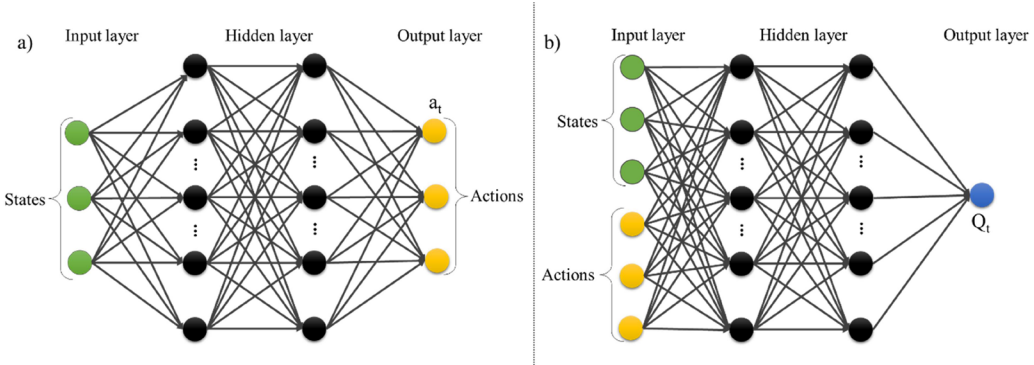


Figure 3. Deep neural network architectures of the proposed Actor (a) and Critic (b) models in the DDPG agent, each featuring two hidden layers with 64 nodes.

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

2.2.1. DDPG Framework and Network Design. The DDPG algorithm employs a sophisticated framework consisting of two key models: the actor and the critic. The proposed actor network architecture consists of three tiers: an input layer that receives the state as input, an output layer that generates the action, and hidden layers, which are two layers with 64 nodes each that record intricate interactions within the input. The critic network follows a similar structure but takes both the state and the selected action as input. After the input layer and several hidden layers, the concatenation layer and output layer are utilized to produce a single Q-value. This actor-critic network design offers a compromise between the RL agent's ability to capture complex policy behaviors and the necessity for computational efficiency, thereby enabling stable training and practical deployment in reaction optimization tasks.[53,58] While more complex network configurations could potentially enhance performance, they would come with increased computational costs, longer training times, and a higher risk of overfitting, which may not justify the performance gains in reaction optimization tasks where computational resources are a key limitation.[58] Then, nonlinearity is introduced to both networks through nonlinear transformation or activation functions such as logistic, SoftMax, tanh or rectified linear unit (Relu). These functions allow neural networks to represent complex relationships between inputs and outputs, enabling them to approximate any continuous function and make them more efficient than linear transformations.[66] The choice of ReLU activation function is primarily motivated by its computational efficiency, gradient stability, and established success in reinforcement learning tasks like DDPG.

After computations, error derivatives are computed backward and gradients backpropagated toward the input layer, allowing weights to be updated to optimize a loss function. A generalized architecture of the actor and critic network for DDPG agent is presented in Figure 3.

These actor and critic networks collectively guide the behavior of the RL agent in continuous action spaces. The actor is a policy network that receives the state as input and produces a precise continuous action, rather than a probability distribution of actions like DQN.[64,65] Whereas the critic network evaluates the quality of actions chosen by the actor by estimating the state-action value function. Both networks have target networks, which are time-delayed copies of the actor and critic networks. They stabilize the learning process by acting as targets during updates.[63]

The actor network $\mu(s \mid \theta^{\mu})$ is responsible for defining the current policy by deterministically mapping the current state of the environment to a specific action (reaction conditions variations). According to Lillicrap et al.,[58] in off-policy algorithms such as DDPG, the exploration process can be treated separately from the learning process. Hence, the exploration policy with the addition of noise sampled from a noise process is given as follows:

$$a_t(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N} \tag{9}$$

Here, s_t and a_t denotes the reactions conditions and their variations at time t, respectively. Noise $(\mathcal{N})$ is introduced to the action to encourage exploration of the action space in the early stages of training. This is essential for balancing the exploration−exploitation trade-off, a fundamental aspect of RL. Exploration refers to the process of trying new actions or strategies to gather more information about the environment or solution space, enabling the agent to discover potentially better solutions that may not have been previously considered. In contrast, exploitation involves leveraging known information to make decisions that are expected to yield high rewards or optimal outcomes based on prior experiences or learning.[67]

After every iteration of the agent interaction with the environment, the network parameter (θ) is updated to optimize the probability that "good" behaviors will be sampled later. These transitions (s_t, a_t, R_t, s_{t+1}) are stored in the experience replay buffer. A minibatch of transitions (N) is sampled from the buffer for updating the actor and critic network.

To maximize cumulative rewards, the DDPG algorithm uses policy gradients to optimize policies by iteratively adjusting the parameters (θ^{μ}). The gradient for the actor network is computed using the deterministic policy gradient theorem, which is given as

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s_i} \tag{10}$$

In essence, the steps in eq 10 involve first computing the gradient of the action with respect to the actor's parameters and then computing the gradient of the Q-value against the action. This dual-gradient strategy allows the actor network to be trained in a way that maximizes the expected Q-value, hence refining the actions it selects to achieve improved performance and enhancing the policy.[58,68]

The critic network $Q(s, a \mid \theta^Q)$ evaluates the quality of the actions suggested by the Actor network and provides feedback by estimating the values (Q-value or reward) of the actions chosen by the actor.[64,65] The evaluation of the Q-value is very critical for evaluating and refining the policy. To demonstrate the influence of various reaction conditions in this work, the Q-value function (Q_{θ}) is presented as follows:

$$Q_{\theta}(s_t, a_t) = \mathbb{E}_{\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \tag{11}$$

where E_{θ} denotes the expected value based on the policy, γ is the discount factor, and $r(s_t, a_t)$ is the instantaneous reward associated with the actions a_t resulting in states s_t. This formulation captures the long-term value of the state-action pair under the policy parametrized by θ. To practically compute and update these Q-values, the Bellman equation used, which provides the recursive relationship for the Q-value:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \max Q(s_{t+1}, a_t) \tag{12}$$

The Bellman operator minimizes the TD error by reducing the difference between the Q function's value before and after the update, estimating the expected value for the future state, s_{t+1}. This error is usually calculated with the distinct target policy (actor) and value (critic) networks, each with unique parameters $(\theta^{\mu'}, \theta^{Q'})$, to stabilize learning. Using the two-norm of this error, the critic loss is expressed as

$$L(Q^{\theta}) = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \tag{13}$$

where y_i represents the target Q-value for the next state, computed using the target networks, which is defined as

$$y_i = r_i(s_t, a_t) + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) \tag{14}$$

Minimizing the mean-squared loss between the target Q-value and the main Q-value $(Q(s_i, a_i \mid \theta^Q)$, as shown in eq 13,

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

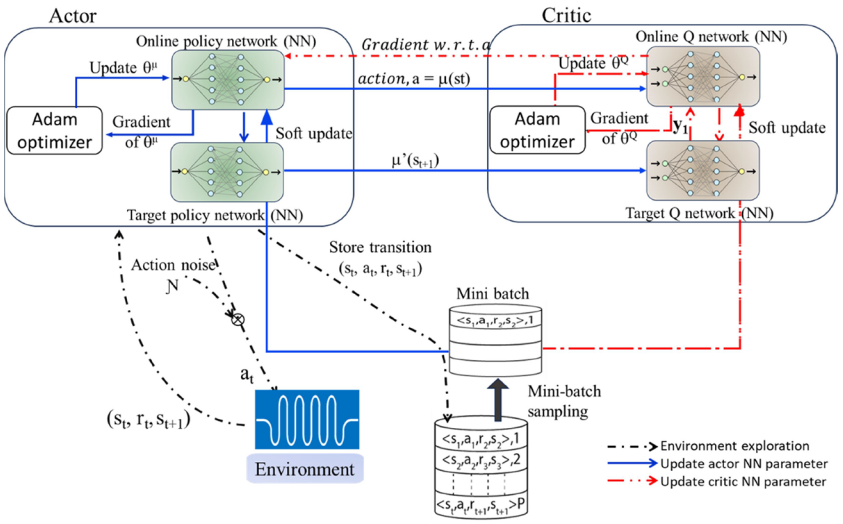


Figure 4. Framework of the DDPG RL agent.

helps in optimizing the critic network approximation. This process ensures that the critic network accurately evaluates the quality of the actions, which, in turn, helps improve the actor network's policy updates.

The target network of actor $\mu'(s \mid \theta^{\mu'})$ and critic $Q'(s, a \mid \theta^{Q'})$, are designed to gradually follow the learning of the main networks. This gradual update significantly improves the stability of the learning process by preventing large, abrupt changes that could hinder it. The target network weights are updated using "soft updates," where a fraction of the main network weights is incorporated with a smoothing factor (τ)[58,69]:

$$\theta^{Q'} \leftarrow \tau\theta^{Q'} + (1-\tau)\theta^{Q} \tag{15}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu'} + (1-\tau)\theta^{\mu} \tag{16}$$

The policy gradient of the actor network depends only on a proficient critic network. This suggests that any improvements made to the critic network will result in an instantaneous improvement in the quality of the actor network updates. Finally, the deployment of the DDPG algorithm is completed by training a policy network using a deterministic policy gradient (eq 10) and a Q-learning network by minimizing the error between the actor and target actor (eq 13). This combination of deep learning and an off-policy actor-critic network builds a powerful framework for effective training in high-dimensional state and action spaces, as shown in Figure 4.

The agent can be conceptualized as an optimization algorithm that aims to learn how to solve a set of objective functions. The state of the agent includes the current experimental conditions in the optimization process and any relevant features like gradients, previous conditions, and objective values. The action taken by the agent is the adjustment made to the current reaction conditions during the optimization process. The policy is the method used to determine which action to take based on the current state and the history of gradients, experimental conditions, and objective values. Learning the policy is learning the best way to update the experimental conditions. The RL algorithm ensures the agent maintains flexibility in solving the objective function without becoming overly specialized.

2.3. Reward Function

As discussed in the previous section, the rewards help the agent refine its actions and policies to converge on an optimal policy or solution. At every step of the episode, the agent receives a reward (r_t) which by accumulating over all steps within an episode, helps to evaluate the total or cumulative reward that needs to be maximized. Here, the objective is to maximize the production of *n*-benzylidenebenzylamine, which is measured by the normalized concentration of the product. To achieve this, a hybrid reward function is proposed, as described in eq 17:

$$c_{3,n}\mathrm{norm}_c = \frac{c_3 - \min(c_3)}{\mathrm{mix}(c_3) - \min(c_3)}$$

$$r_t = \begin{cases} \mathrm{norm}_c + 100 & \text{if, } c_3 > 1.6 \\ \mathrm{norm}_c & \text{else if, } 1.2 < c_3 \le 1.6 \\ \mathrm{norm}_c - 15 & \text{else, } c_3 \le 1.2 \end{cases} \tag{17}$$

Here, c_3 is the concentration of the product at the outlet of the reactor. Min (c_3) and max (c_3) are respectively minimum and maximum product concentrations experienced during training. Since the yield of c_3 directly reflects the conversion rate with 100% selectivity (no side reactions or byproducts), the reward calculation aligns with yield computation. This means the reward function, based on the normalized concentration of c_3, directly measures performance, simplifying the reinforcement learning process.

2.4. Hyperparameter Optimization

The training of RL agent involves both parameters and hyperparameters. Parameters, such as weights and coefficients, are internal values associated with DNN learned during training. In contrast, hyperparameters are external configuration settings of an agent that must be defined beforehand. Hyperparameter optimization is the process of systematically finding the optimal values for these external settings to maximize the performance. Hyperparameter settings play a critical role in shaping how the agent learns and adapts during training, influencing its performance and efficiency.[59,70,71] In DRL, careful optimization is essential for optimizing learning speed, stability, and overall effectiveness. Techniques such as

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

fANOVA have revealed that, depending on the environment, one or two hyperparameters may be significantly more critical than others,[72] highlighting the need for more systematic and targeted optimization approaches. Traditional methods, such as OVAT tuning or grid search, often prove inefficient and costly, focusing solely on final hyperparameter settings without fully quantifying their impact throughout the training process. While OVAT methods are frequently used due to their simplicity and ease of implementation, they often fail to explore the hyperparameter space comprehensively and can be resource intensive. The goal here is not just to find good hyperparameter settings but also to quantify the effect of hyperparameter tuning. To overcome these limitations, Bayesian optimization has emerged as a sophisticated Hyperparameter Optimization (HPO) method. It models the objective function probabilistically, enabling a more targeted and efficient exploration of the hyperparameter space. Unlike the OVAT approach, which evaluates hyperparameters independently, Bayesian optimization considers the interactions between multiple hyperparameters simultaneously, leading to more informed and effective tuning.

One of the most widely used tools for using Bayesian optimization is Optuna, an open source hyperparameter optimization framework. Optuna leverages the Tree Parzen estimator to manage tree-structured search spaces and conditional parameters while utilizing kernel density function estimators for efficient optimization.[73] The tool provides efficient search strategies, support for distributed computing, and automatic algorithm selection, making it a powerful tool for hyperparameter tuning. In this study, hyperparameters such as learning rates for actor and critic networks, noise exploration factors, discount factors, and smoothing factors were optimized. Performance metrics, including average reward, convergence time, and policy stability, were used to evaluate the effectiveness of each method. This work compares the performance of traditional OVAT tuning with Bayesian optimization, implemented using the Optuna framework, to identify which approach yields superior results in DRL tasks.

2.5. Training and Refinement

The training performance of any RL agent is strongly dependent on the exploration and exploitation trade-off. A RL agent is trained to find the optimal solution as fast as possible to minimize the experimental cost. However, selecting a solution too early without full exploration is not a good approach as it may leads to either unsuccessful training or suboptimal or local solutions. To achieve effective training, an agent must explore thoroughly the operating/decision space to identify optimal policies, in a cost-effective manner, and avoid suboptimal solutions. Several algorithms can be used to enhance exploration by adding randomness or noise to the actions, states, or directly to the weights of the actor network. When selecting actions, noisy policies add randomness directly to the decision-making process while ε-greedy policies probabilistically balance exploration for new actions and exploitation of known ones.

Noise based exploration was suggested by Lillicrap in their introductory paper for continuous action.[58] As given in eq 9, the noise ($\mathcal{N}$) can be tailored to suit the characteristics of the environment by adjusting its variance, exploring different noise distributions, and making it adaptive to the agent's performance and environment conditions. The proposed method employs the Ornstein−Uhlenbeck (OU) process, a stochastic process introduced by Uhlenbeck and Ornstein[74] and later used by Lillicrap et al.[58] to generate time-dependent temporally correlated noise. OU noise is calculated independently for each action, as follows:

$$\chi_{a_t} = \chi_{a_{t-1}} + \theta(\mu - \chi_{a_{t-1}})\cdot dt + \sigma\sqrt{dt}\cdot\mathcal{N}(0, I) \tag{18}$$

Here, χ_{a_t} is the current value of the noise, θ suggests the rate at which noise returns to the mean, μ represents the long-term mean, and σ is the volatility of the noise. The mean serves as a central tendency or baseline value for the noise over time $\mathcal{N}(0, I)$ represents a random variable sampled from a standard normal distribution The OU process introduces autocorrelation in the noise, which helps the agent explore the action space more effectively.

In ε-greedy, the exploration-exploitation trade-off is achieved by randomly selecting actions at the beginning of the training and more purposefully toward the end of training. The agent computes epsilon (ε), as follows:

$$\varepsilon = \varepsilon_{\min} + \frac{(1 - \varepsilon_{\min})}{e^{-\lambda - N_t}} \tag{19}$$

where $\varepsilon_{\min}$ is the lowest probability of choosing a random action at the end of the training. The parameter λ, which governs the rate of exponential decay, gives the rate of exploration decline over time. As the number of training iterations (N_t) increase, the exponential decay of ε indicates a decreasing likelihood of selecting random actions. This decay mechanism ensures that the agent proceeds from exploratory activity to the gradual exploitation of learned information, hence improving its decision-making capabilities. By using exponential decay to adjust ε intelligently, the agent begins with high probability of exploring new actions with limited knowledge and discover various possible strategies. As training progresses, ε decreases, leading the agent to exploit best known actions more frequently. This balance allows the agent to collect comprehensive information and enhance decision-making based on accumulated knowledge. Additionally, this improves the agent's learning trajectory, integrating new information and prior learning for more informed decisions in complex situations.

3. RESULTS AND DISCUSSION

In this section, the methodology and RL algorithms described in the previous section are implemented. The simulations were conducted in Python version 3.9. The mathematical model described earlier (eqs 3−6), representing the imine synthesis in flow, is used as the environment to train the proposed DRL agents and validate and test the proposed architectures and methods. The model is simulated using the initial conditions and variable values outlined in Table 1. The method of lines is employed to convert the partial differential equations into ordinary differential equations (ODEs), which can be solved numerically. In the current case, the resulting ODE system is solved in Python using the 'solve_ivp' function, which addresses initial value problems with stiffness. The solver employs a backward differentiation formulas method, a robust technique designed to efficiently solve stiff equations. The agent interacts with the reactor model (environment) continuously by changing the manipulated variables (actions) and collecting the states to evaluate the rewards. A complete list of Python packages used is provided in Section A of the Supporting Information (SI).

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

Prior to the self-optimization efforts, a parameter estimation procedure was conducted based on collected experimental data. This is followed by a comprehensive validation and testing of the DDPG and reaction network setting. The impact of hyperparameters on the training procedure and outcomes is demonstrated through systematic parametric analysis. At the end, the performance of the DRL approach is compared against the state-of-the-art techniques such as SnobFit and Nelder–Mead.

A DRL agent was designed based on the proposed actor and critic network architecture as discussed in the previous section. The DDPG agent was trained based on a maximum number of episodes of 100, each episode consisting of 10 decision/action steps. The replay buffer size is kept at 5000 to store training experience, which allows the algorithms to learn from a more diverse set of experiences.

3.1. Estimation of the Model Parameters

To build a reliable model able to capture or represent the RL environment, several model parameters must be identified based on a set of experimental data. Here the kinetic parameters, namely the activation energy (E_a) and the rate coefficient k_{ref}, which are critical to predict how the reaction behaves under various conditions and consequently allowing systematic optimization of the reaction process, are identified from the experimental data. Accurate determination of these parameters ensures that the mathematical model, used as a training environment for the DRL agent, aligns closely with real-world outcomes. The remaining model parameters (e.g., UA, ΔH_{react}) were fixed at nominal values obtained from the literature (Table 1).

The results shown in Figure 5 represent the concentration profiles over time obtained under four different temperatures

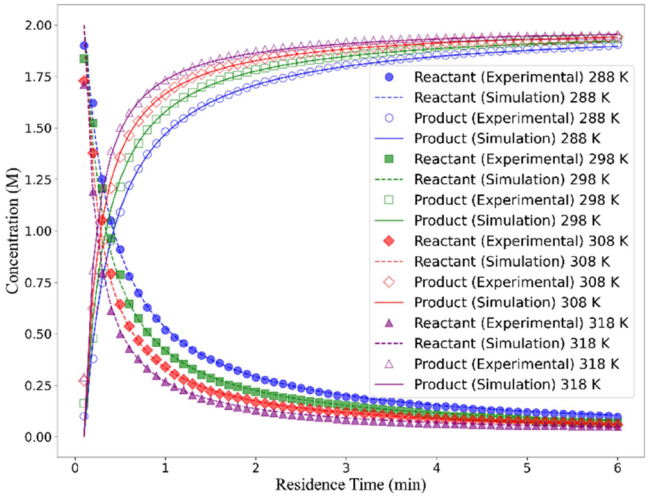


Figure 5. Model predictions vs experimental data (a) Concentrations of the product and reactant (benzaldehyde) at the reactor outlet, at different temperatures.

(288, 298, 313, and 318 K). The experimental data capturing the concentration profiles of the reactant (benzaldehyde) and product (imine) were used for the parameter estimation procedure based on the minimization of the sum of the squared errors. The optimal kinetic parameters are summarized in Table 2. As shown in Figure 5, the model demonstrates excellent prediction capabilities with respect to the four different experiments. Additional details are provided in Section B of the Supporting Information (SI).

Table 2. Optimal Estimates of the Kinetic Parameters of the Imine Synthesis

parameter	optimal value
activation energy, E_a (kJ mol^{-1})	21.57
reaction rate coefficient k_{ref} (L mol^{-1} min^{-1})	0.413

3.2. Tuning of the DRL Hyperparameters

In DRL, the performance of an agent is strongly dependent on the choice of the hyperparameters. In this section, we present and compare the results associated with the hyperparameter optimization using an advanced technique, namely Bayesian optimization via Optuna, against a tuning strategy based on the traditional trial-and-error methods. Both approaches were evaluated under identical conditions, including the same environments, network architectures, and performance metrics. Here, the emphasis is on five key hyperparameters, namely the learning rates for the actor and critic networks, discount factor, target network update rate, and noise addition, which significantly impact the agent's decision-making, stability, and gradient descent efficiency.

The learning rates of the actor and critic networks are indeed critical hyperparameters in DRL algorithms. The convergence performance of the proposed algorithms with various learning rates is illustrated in Figure 6. The figure

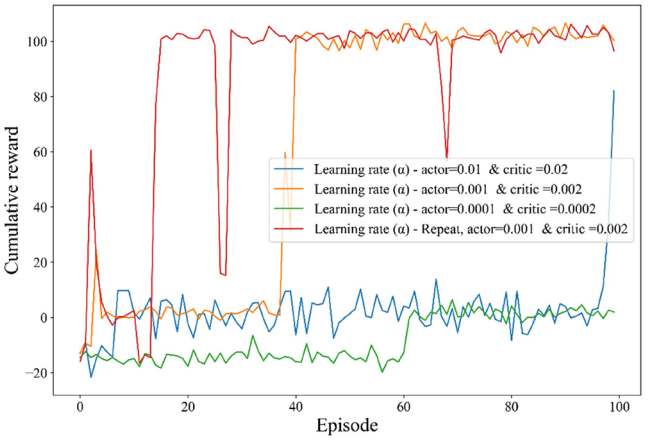


Figure 6. Convergence of training noisy action policies at different actor and critic learning rates.

clearly shows that both very low and very high learning rates result in poor convergence. The learning rate determines the magnitude of the parameter adjustments during training, directly influencing how quickly or stably the model learns. When the learning rate is too high, parameter adjustments become overly large, often destabilizing the training process. This can lead to oscillations or divergence by overshooting the optimal solution, even when using adaptive optimizers like Adam, which dynamically adjust step sizes based on gradient information. High learning rates are particularly prone to causing instabilities, as the excessively large parameter changes hinder convergence. Conversely, very low learning rates result in infinitesimal parameter changes, significantly increasing the number of iterations or episodes required for training and resulting in slower convergence. This slower progress can also increase the likelihood of the model becoming trapped in

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247–266

suboptimal solutions. Therefore, carefully selecting an appropriate learning rate is crucial, as it governs the scale and stability of parameter refinement throughout the training process. Based on the observed performance, the optimal learning rates for updating the actor and critic networks are $\alpha_{actor} = 0.001$ and $\alpha_{critic} = 0.002$, respectively. These values help achieve a good trade-off between stability and convergence speed.

The exploration noise parameter (σ_e), which introduce randomness into the actions taken by the agent, are crucial for effective exploration within the decision space. In this study, the impact of the exploration noise parameter was evaluated within the range of 0.0001 to 0.5. This range was chosen to be very wide to cover a broad spectrum of exploration behaviors, from minimal to extensive noise. The results of this investigation are presented in Figure 7. When σ_e is set too

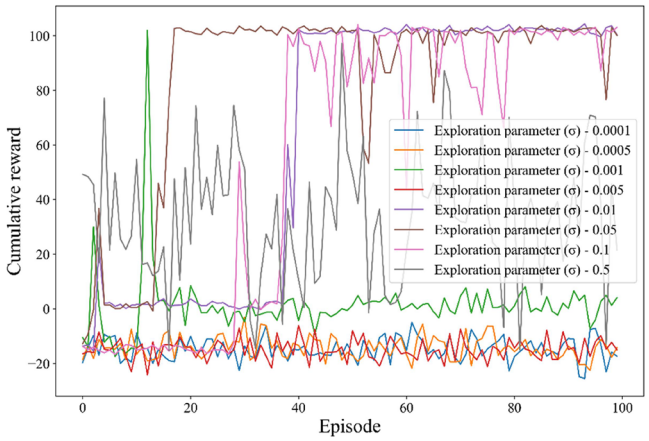


Figure 7. Training performance and convergence at different values of the noise exploration parameter (σ_e).

low (e.g., between 0.0001 and 0.005), the agent experiences minimal randomness, leading to inadequate exploration, premature convergence to suboptimal policies, and a tendency to become trapped in local optima. Conversely, larger values of σ_e result in more important noise distributions, encouraging the agent to explore a broader range of actions. However, the impact on performance varies depending on the specific value of σ_e. For example, with a higher value like $\sigma_e = 0.5$, the exploration noise is significant, leading to substantial randomness in the agent's actions. This excessive exploration can cause unstable performance, with cumulative rewards fluctuating between 20 and 40, as the agent struggles to maintain a stable policy due to the erratic behavior induced by the high noise level. Therefore, as shown in Figure 7, the observed optimal value of σ_e lies around 0.01, where exploration is sufficient to discover a broad set of potential policies without disrupting the stability and efficiency of the learning process.

The discount factor, denoted as γ, is a critical hyperparameter in the DDPG framework, as it determines the weight given to future rewards relative to immediate rewards, thereby influencing the balance between long-term and short-term learning and decision-making. In our study, we examined discount factors ranging from 0.001 to 0.999 to assess their impact on the agent's learning process and overall performance. When γ is less than 0.7, the agent tends to prioritize short-term rewards, which can lead to more stable but potentially suboptimal learning curves. This gratification of immediate rewards impairs the agent's ability to consider long-term benefits, resulting in poor training responses and limited overall performance, as illustrated in Figure 8. Conversely, a

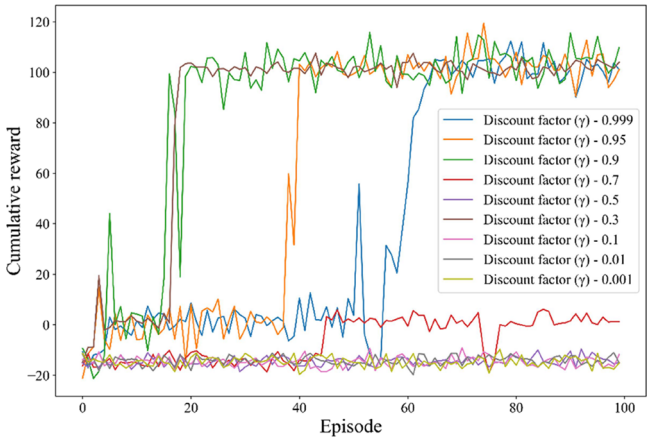


Figure 8. Training convergence under different discount factors.

high discount factor, here above 0.9, enhances the agent's resilience by mitigating its sensitivity to abrupt changes in the environment. This leads to improved stability in learning and better convergence, as the impact of noisy rewards is reduced. These findings suggest that a discount factor (γ) within the range of 0.9−0.999 provides a reasonable balance. By adopting values within this range, it, further stabilizes the learning process and places greater emphasis on long-term benefits, thereby improving overall learning performance.

The smoothing factor (τ) is also a critical parameter that controls the rate at which the parameters of the target network are updated relative to those of the main (or online) network. As discussed in Section 2.2.1, the main actor and critic networks are updated directly during training, whereas the target network undergoes a slower update using a soft update rule. This gradual update helps stabilize the learning process by preventing abrupt changes in the target values. As illustrated in Figure 9, varying τ values from 0.0001 to 1 significantly impacts the cumulative rewards achieved by the agent over the specified episodes. Specifically, higher τ values, such as 1 and 0.1, cause the target network's parameters to more closely track those of the main network, resulting in faster updates.

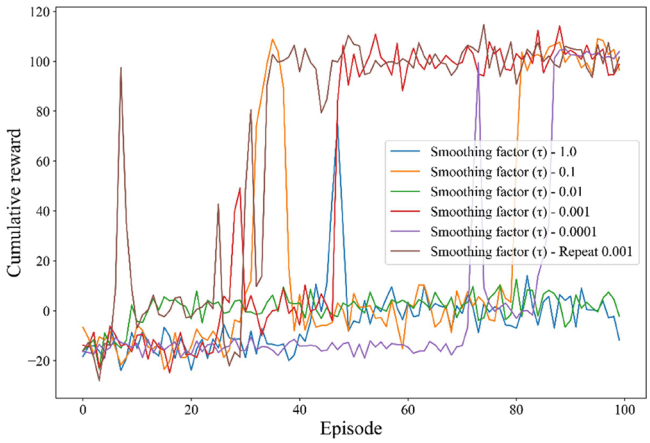


Figure 9. Smoothing factor (τ) influence on cumulative rewards under noisy environments.

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

Although this can accelerate the learning process, it may also introduce instability if τ is set too large, as the target network may react too quickly to changes in the main network. Conversely, lower τ values, such as 0.001 and 0.0001, result in slower updates to the target network's parameters, enhancing stability by reducing rapid fluctuations and oscillations. However, this comes at the cost of slower adaptation to changes in the main network, which could hinder learning. The observed performance trends in the graph underscore the importance of selecting an optimal smoothing factor that balances responsiveness and stability. An updated range of 0.001 to 0.01 emerges as a promising trade-off, allowing the agent to effectively adapt to changes in the environment while ensuring robust performance. Notably, the τ value of 0.01 demonstrates a good compromise, yielding substantial cumulative rewards while maintaining a stable learning trajectory. The observed performance trends illustrated in Figure 9 underscore the importance of selecting an optimal smoothing factor that balances responsiveness and stability. These findings highlight the critical role of the smoothing factor in shaping the learning dynamics of RL agents, emphasizing the need for careful hyperparameter tuning to enhance overall performance.

All the optimized hyperparameters, including the discount factor γ, the smoothing factor τ, the learning rates for the actor and critic networks, and the exploration noise parameter σ_e, are summarized in Table 3. The results reveal notable differences between the two approaches, highlighting the impact of systematic optimization on hyperparameter selection. A key advantage of Bayesian optimization is its ability to tune several or all hyperparameters simultaneously, whereas the trial-and-error method typically adjusts one parameter at a time, making it a more time-consuming and less efficient process. One notable observation is the significantly lower critic learning rate (α_{critic}) achieved through Bayesian optimization (0.00043) compared to trial-and-error tuning (0.002). A lower learning rate for the critic network often enhances stability in DRL by preventing large updates that could destabilize training. Similarly, the smoothing factor (τ) is optimized to a much smaller value (0.0004) through Bayesian optimization, which suggests a more conservative update strategy for target networks, potentially leading to more stable learning. Another important difference is seen in the exploration noise parameter (σ_e). Bayesian optimization results in a significantly higher value (0.1019 vs 0.01), indicating a preference for increased exploration during training. This suggests that Bayesian optimization identifies a balance between exploration and exploitation that may not be as easily achieved through manual tuning. The discount factor (γ) and actor learning rate (α_{actor}) show smaller differences, with Bayesian optimization yielding slightly higher values (0.961 vs 0.95 for γ and 0.002 vs 0.001 for α_{actor}). These adjustments may contribute to better long-term reward optimization and more effective policy updates.

Table 3. Ranges of DDPG Agent Training Hyperparameters and Their Optimal Values Obtained through Trial-and-Error and Bayesian Optimization

hyperparameter	hyperparameter predefined range	optimal value	
		trial-and-error	Bayesian optimization
critic learning rate (α_{critic})	0.0002−0.02	0.002	0.00043
actor learning rate (α_{actor})	0.0001−0.01	0.001	0.002
discount factor (γ)	0.001−0.999	0.95	0.961
exploration noise parameter (σ_e)	0.0001−0.5	0.01	0.1019
smoothing factor (τ)	0.0001−1.0	0.01	0.0004

The computational cost of the different tuning strategies is also key. As anticipated, the Bayesian optimization demonstrates clear advantages over trial-and-error. As shown in Table 4, Bayesian optimization completed 100 trials across the considered 5 hyperparameters in approximately 11.47 h. In contrast, trial-and-error required 7 h to complete only 30 trials to deliver suboptimal performance. Another way to establish a reliable comparison between the Bayesian optimization and trial-and-error consists of running the latter under similar numbers of trials (i.e., 100 trials). As expected, the extended trial-and-error approach comes with an even higher computational cost without guaranteeing optimal performance. It is worth mentioning that the computational time does not increase linearly with the increased number of trials in all cases because changes in any given hyperparameter have a different impact on the training features and performance and consequently on the inherent computational cost. In essence, the training performance and the computational cost have different sensitivities to changes or increments in the hyperparameter values.

Table 4. Computational Time Associated with the Hyperparameter Tuning Based on Trial-and-Error vs Bayesian Optimization

HPO method	No. of trials	No. of hyperparameter	computational time
Bayesian optimization	100	5	∼11.47 h (41,292 s).
trial-and-error	30	5	7 h (25,200 s).
trial-and-error (extended)	100	5	∼2 days (∼172,800 s)

Overall, the results demonstrate that Bayesian optimization can explore a broader search space and fine-tune hyperparameters more effectively than the trial-and-error method. These optimized values could potentially lead to improved convergence speed, training stability, and final policy performance. Table 5 provides additional parameters relevant to the overall design and training process of the DDPG algorithm.

Table 5. Additional Parameters and Settings of the DDPG Agent

parameters	values
episode	100
iterations per time step	10
optimizer exploration policy	OU-noise
optimizer	Adam
batch size	64

3.3. Training Performance Using ε-Greedy and Action-Noise Policy

In this part, we investigate the training performance and inherent exploration strategy of the DDPG agent discussed in section 2.5. The action-noise policy was evaluated using the set of hyperparameters optimized using Bayesian optimization as discussed in the previous section, with the training perform-

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

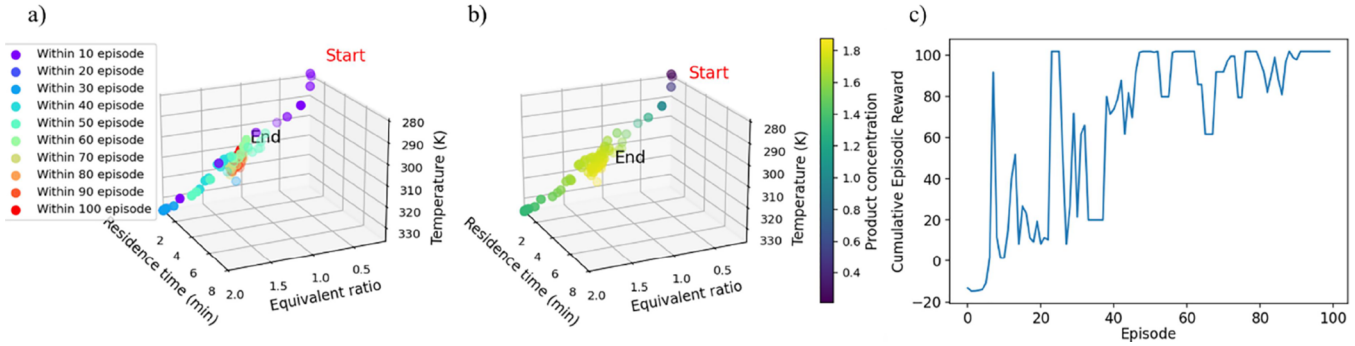


Figure 10. Training performance of noisy action policy at optimized hyperparameter.

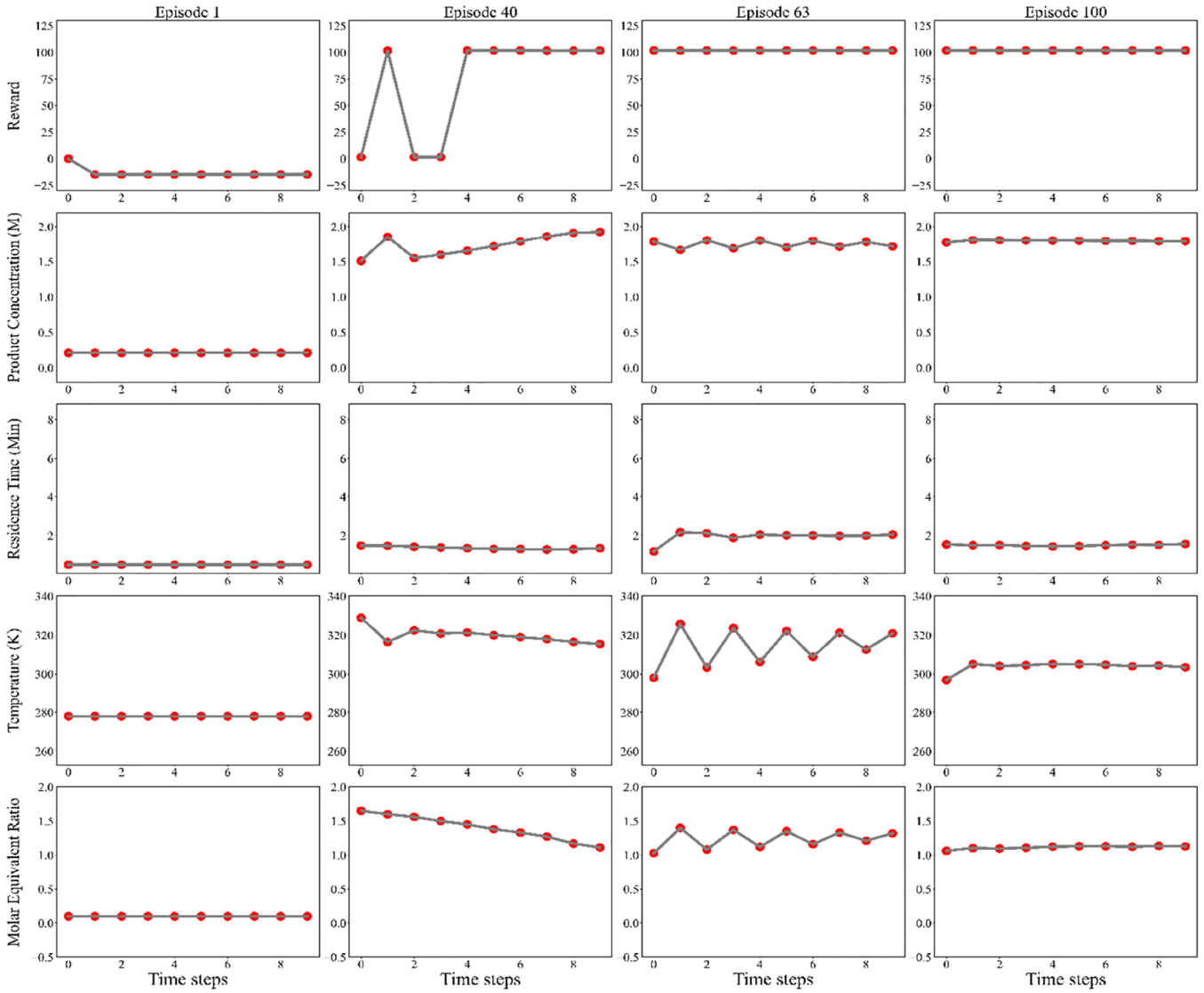


Figure 11. Evolution of the sequences of the decision variables (actions) implemented by the agent over the 1st, 40th, 63rd, and 100th training episodes. (Red dots indicate the moment at which the agent takes actions.)

ance illustrated in Figure 10. Figure 10a depicts the agent's exploration performance within the decision space, transitioning to exploitation as training episodes progress. The DDPG agent's training starts as shown by the purple dots, gradually converging to the optimal reaction conditions (red dots). Figure 10b illustrates the agent's learning dynamics associated with the optimization of product concentration. The consistent initialization yet varied trajectories indicate the stochastic nature and complexity of the optimization process. The color gradient, representing product concentration, demonstrates the agent's adaptive decision-making as it refines its policy to achieve optimal reaction conditions over successive training episodes. Figure 10c demonstrates that the DDPG agent, based on the hyperparameters optimized via Bayesian optimization,

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

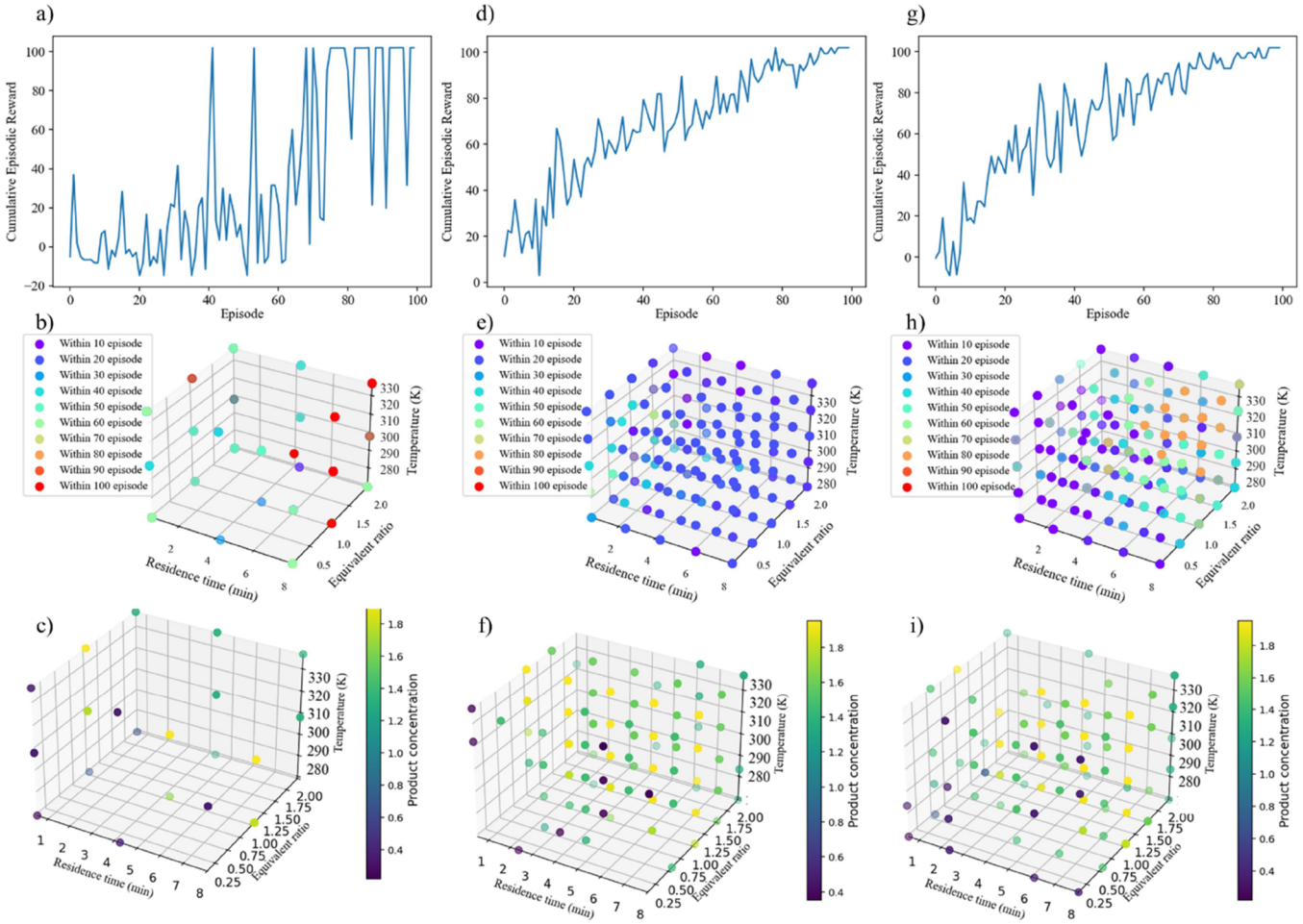


Figure 12. Training performance of the DDPG agent based on the optimized hyperparameters for the ϵ-greedy policy with batch sizes of 64 and 10 steps (a–c), 64 and 40 steps (d–f), and 128 and 40 steps (g–i), respectively.

exhibits the best training and convergence performance toward the optimal operating conditions of the imine synthesis reaction. Initially, the agent explores the decision space during the first 40 training episodes, subsequently shifting to exploitation guided by the best policy. Notably, the training persists beyond the 50th episodes, as indicated by the cyan-colored dots in Figure 10a, demonstrating the agent's continued learning and adaptation to optimize performance. This extended training period allows the DDPG agent to further refine its policy and gain deeper insights into the system's dynamic behavior based on finely tuned actions. The training is ultimately achieved after reaching the maximum number of episodes, marked by the red dots at the 100th episode, or when the moving average of cumulative rewards stabilizes, indicating convergence to the optimal solution. This comprehensive analysis underscores the DDPG agent's capability to navigate the decision space effectively, balancing exploration and exploitation to optimize performance of the imine synthesis reaction.

Figure 11 shows the detailed training performance of the DDPG agent over the iterative interactions with the environment. The agent's learning process begins with the initial actions selected randomly for each episode to explore the decision space. These initial action values, such as residence time, molar equivalent ratio, and reactor temperature, serve as a baseline from which the agent iteratively adjusts its decisions based on feedback (rewards) received from the environment. During early training episodes, the agent's actions received low rewards, indicating the exploratory phase, which is critical for the agent learning process. Over time, the agent progressively converged toward higher cumulative rewards and associated optimal conditions. By episode number 28, the agent began to refine its actions, leading to increased cumulative rewards and inherently higher product concentrations. This behavior also captures a shift from exploration to exploitation of the learned strategies. Specifically in episode 40, molar ratio and temperature sequentially decreased from 1.7 to 1.4 and from 338 to 319 K, respectively, to increase the product concentration. Additionally, the residence time approached its optimal value during this phase, further contributing to the agent's improved performance. As the training progressed, the agent continued exploring and gradually converged toward optimal conditions, as evidenced by the stabilization of the values of the decision variables in later episodes. For instance, actions between episodes 63 and 100 are finely tuned to continuously produce maximum productivity (more than 1.8 mol/lit), which suggests that during these first 63rd episode, the agent was mainly performing exploitation to deliver global optimal reaction conditions with the help of the additionally collected knowledge from the environment (i.e., the imine synthesis reactor).

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247–266

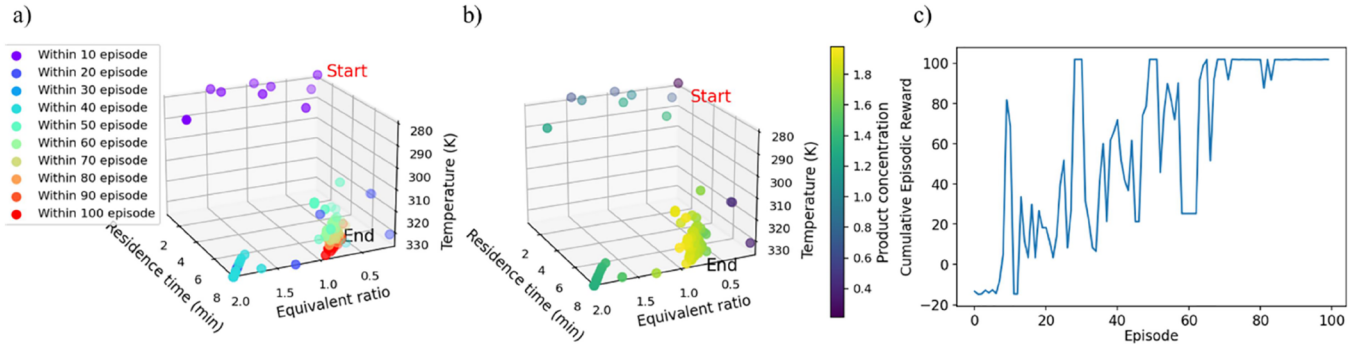


Figure 13. Training performance of the DDPG agent at adaptive noise.

The performance of the DDPG agent is evaluated under a noisy action policy and compared with the ε-greedy policy to gain insight into the agent's resilience under various conditions, as explained in detail in section 2.5. Additionally, this allows a comprehensive assessment of the convergence rates, effectiveness, and environmental adaptation, offering insightful information about the fundamental processes and guiding the best line of training action in complex systems. The training performance of the ε-greedy policy is shown in Figure 12a,d,g, whereas Figure 12b,e,h depicts the corresponding decision spaces that the agent explored, and Figure 12c,f,i emphasizes the product concentration. Initially, as shown in Figure 12a–c, the DDPG agent trained under the ε-greedy policy with the same set of hyperparameters and setting (Tables 3 and 5) required additional episodes and significant simulation time to achieve convergence. From Figure 12a, it can be observed that the agent engaged in exploration for the first 60 episodes, as indicated by unstable and low episodic rewards, reflecting its attempt to learn the environment's dynamics. However, as shown in Figure 12b,c, the agent explored only a limited section of the decision space before transitioning to the exploitation stage. This shallow exploration, coupled with the absence of convergence to specific optimal actions under the current conditions and settings, suggests the need for further improvements. Addressing this limitation may require additional data, either through increasing the number of episodes or the number of actions/steps per episode, to encourage deeper exploration of the environment representing the imine synthesis reaction.

To mitigate the observed shallow exploration, the number of steps per episode was increased from 10 to 40. This adjustment enabled the agent to evaluate a greater number of actions per episode, thereby generating more data points and enhancing its exploration capability. As shown in Figure 12d–f, the increased number of steps facilitated improved exploration, resulting in smoother and more stable training performance with higher cumulative rewards. Importantly, this approach reduced the need for additional episodes, though it came at the cost of increased computational demand. To further enhance training stability, the batch size was increased from 64 to 128. This modification allowed the agent to sample a larger number of experiences from the replay buffer at each training iteration, leading to more stable updates. As shown in Figure 12g–i, this adjustment resulted in smoother training curves and a more comprehensive exploration of the decision space. Ultimately, these changes enabled the agent to achieve improved and consistent performance, despite the trade-off of increased computational costs. The smoother training and more thorough exploration led to more effective exploitation of the learned policies, ensuring the agent could more reliably reach optimal conditions within the environment over time.

3.4. Adaptive Hyperparameters Tuning

The previous sections highlighted the performance of DDPG agents based on different hyperparameter tuning/optimization strategies. As clearly shown, the agent performance was significantly enhanced using the Bayesian hyperparameter optimization approach. However, fixing the hyperparameter values for the entire training duration may impose limitations due to the conflicting and competing nature of certain parameters, which require inherent tuning compromises. These trade-offs can affect the overall performance of the RL agent, as many of the parameters will be given suboptimal values, resulting in a poor training performance. To overcome these limitations, an adaptive or dynamic hyperparameter tuning will be investigated to deliver a more effective solution. Adapting the parameter values during training may enable the agent to maintain flexibility, fostering efficient exploration in the early stages while gradually emphasizing exploitation as learning progresses.

An effective adaptive tuning approach is employed to update the hyperparameters during training as described in eq 20 below. The method allows the implementation of a dynamically adjusted real hyperparameter value as opposed to methods based on discrete values, enhancing the adaptability of the proposed methodology.[75]

The hyperparameter values are dynamically adjusted over the training stages based on the equation below.

$$\mathrm{HP}_t = \max(\mathrm{HP}_{\max} - \alpha \times A_t, \mathrm{HP}_{\min}) \tag{20}$$

In the equation above, HP_t represents the dynamically adjusted hyperparameter at time t, bounded by $\mathrm{HP}_{\max}$ and $\mathrm{HP}_{\min}$. α is the decay rate which ensures a smooth transition from higher to lower values as training progresses. A_t a progress or adaptation metric (A_t) which depends on the total number of steps per episode (N_{total}), the current episode (Ep_t) and the current training step (N_t). A_t is defined as

$$A_t = N_{\text{total}} {}^{*}\mathrm{Ep}_t + N_t \tag{21}$$

During early training stages, higher hyperparameter values, closer to the maximum allowed value ($\mathrm{HP}_{\max}$), are implemented as the adjustment term relating to the number of episodes and steps (A_t) (eq 21) is still very small. As a result, the agent undergoes broader exploration. As the training progresses, the term A_t (in eq 21) increases and consequently the hyperparameter values gradually decrease and converge

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247–266

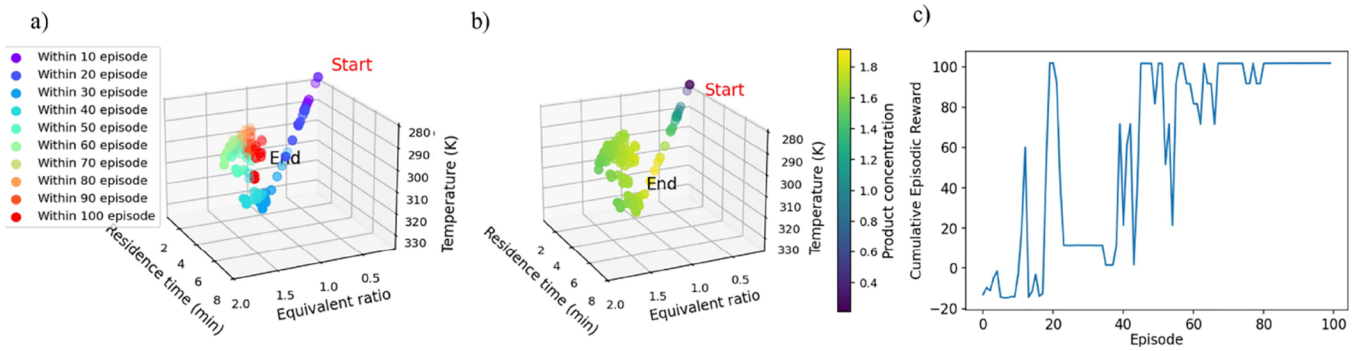


Figure 14. Training performance of the DDPG agent at adaptive learning rate.

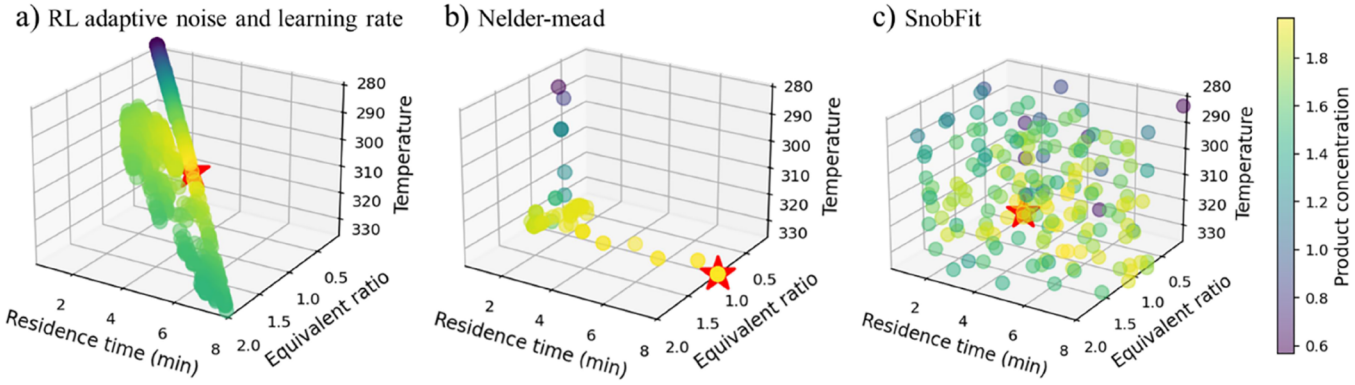


Figure 15. Performance of the DDPG agent vs Nelder−Mead and SnobFit.

toward their minimum values (HP_{min}). This gradual reduction ensures a smooth transition from exploration to exploitation and enhances convergence toward the global optimal solutions.

This dynamic framework allows the agent to adapt its learning process, ensuring an optimal balance between exploration and exploitation throughout training. To demonstrate the capabilities of the adaptive hyperparameters tuning, two critical parameters will be dynamically tuned, namely the exploration noise and learning rates. The use of adaptive exploration noise, as opposed to fixed noise, delivers improved training performance, as shown in Figure 13a compared to Figure 10a, evidencing enhanced exploration and the agent's capability to identify the global optimal conditions. At the beginning of training, higher levels of noise introduce substantial randomness, encouraging broad exploration of the decision space. During this phase, the agent actively explores the reaction decision space, taking sequential steps toward identifying the optimal conditions, shown in Figure 13a,b. As the agent becomes more confident in its learned policy, the noise is systematically reduced, facilitating a smoother transition from exploration to exploitation. This shift is reflected in the cumulative rewards, which initially show strong exploration up until approximately the 40th episode. Following this, the agent begins to converge toward optimal reaction conditions, as evidenced by the clustering of points in the decision space toward the "End" point in Figure 13a,b, and, ensuring maximum cumulative rewards as it focuses on exploiting its learned strategy. Based on the convergence observed in the decision space (Figure 13a,b) and the stabilization of cumulative episodic reward (Figure 13c), the agent demonstrates effective learning throughout the training period. However, the distribution of data points within the decision space suggests that the extent of exploration may be insufficient to definitively conclude convergence toward a global optimum. To encourage further exploration and potentially identify superior reaction conditions, it is beneficial to gradually lower the learning rate for both the actor and critic networks.

In the case of learning rate adjustment, the dynamic adjustment of the learning rate ensures optimal progression throughout training. A higher effective learning rate at the start accelerates policy updates, allowing the agent to explore a broader range of state-action pairs and adapt to diverse operating conditions. As training progresses, the gradual reduction of the learning rate facilitates finer adjustments to the policy, improving stability and convergence. When combined with adaptive noise, this strategy improves performance by encouraging more efficient exploration and exploitation during the training phase. The action taken in the decision space and training performance are presented in Figure 14a, b, and c respectively. As shown in Figure 14a,b, the agent explores a broad range of conditions early in training, with data points scattered across the decision space within the first 50 episodes, especially at the start of the training process. After 50 episodes, there is a clear convergence toward the optimal conditions, represented by a cluster of data points toward the "End" region of the plots, indicating a shift towards exploitation. This transition aligns with the increasing cumulative episodic reward observed in Figure 14c, where a rapid increase occurs between episodes 20 and 50, followed by stabilization, and then converging toward a maximum between episodes 50 and 80. With the proposed dynamic hyperparameters tuning, the agent identifies the optimal residence time (min), equivalent ratio and temperature (K) at 3.8 min,

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

Table 6. Comparison of the Key Performance Indicators of the Different Self-Optimization Methods[a]

			optimal solutions			
method	initial guess	number of experiments/ iterations	manipulated variables	c_3 (M)	experimental c_3 (M)	computation cost (to achieve optimal solution)
Nelder-Mead	**Case 1**: $ER_{1/2}$: 0.714, Res_t: 0.66, T: 280.55	118	$ER_{1/2}$: 1, Res_t: 8, T: 330	1.81	1.79	∼66 s
	Case 2: $ER_{1/2}$: 7.9, Res_t: 1.48, T: 313.27	96	$ER_{1/2}$: 0.63, Res_t: 7, T: 330	1.65	1.50	∼54 s
SnobFit		250	$ER_{1/2}$: 1, Res_t: 6.8, T: 325	1.80	1.81	∼132 s
DRL (Bayesian)		85	$ER_{1/2}$: 1.12, Res_t: 2.1, T: 330.59	1.82	1.80	∼400 s
DRL (adaptive noise)		75	$ER_{1/2}$: 1.12, Res_t: 3.8, T: 320.59	1.83	1.81	∼400 s
DRL (adaptive noise and learning rate)		75	$ER_{1/2}$: 1.1, Res_t: 3.8, T: 321.57	1.84	1.83	∼400 s

[a]$ER_{1/2}$: equivalent ratio, Res_t: residence time, T: temperature, and c_3: product concentration.

1.1 and 321 K, respectively. The adaptive hyperparameter tuning delivers a more effective balance between exploration and exploitation, resulting in best training performance and an optimal solution to the flow chemistry self-optimization problem. It becomes clear that the adaptive hyperparameter tuning leads to smoother learning trajectories, faster convergence, and overall performance improvements.

3.5. Comparison of the DRL Agent against Gradient-Free Optimizers

The performance of the proposed RL strategy is compared against two widely used gradient-free optimization strategies (Nelder−Mead and SnobFit), for the self-optimization of the chemical reactions in flow. The exploration performance within decision space of these techniques is shown in Figure 15. The red stars in Figure 15a−c represent the optimal reaction conditions achieved by each optimizer. The figures show that each optimizer converges to optimal reaction conditions that deliver maximum product concentration of imine. It becomes clear that the proposed optimizer (RL based DDPG) outperforms Nelder−Mead and SnobFit based on the key performance indicators summarized in Table 6. This superiority is attributed to the intrinsic ability of DRL frameworks to manage the exploration-exploitation trade-off via policy gradient updates and deterministic action selection, a feature that gradient-free optimizers do not explicitly possess. Nelder−Mead identifies local optima through Simplex based refinement making it highly sensitive to initial guesses or starting conditions and therefore limited in its ability to explore the broader solution space. Conversely, SnobFit algorithms employs branch-and-fit strategies to achieve global optimization solution; however, this often comes at the cost of requiring a greater number of experimental evaluations.

These results underscore the differences in efficiency, sensitivity to initial conditions, and the number of experiments required by each method to converge to optimal solutions. The proposed RL based approach converges to an optimal solution in a mere 60 experiments, a very small number when compared to Nelder−Mead and SnobFit, which require 120 and 250 experiments, respectively. However, Nelder−Mead algorithms identify optimal reaction conditions at the boundaries of temperature and residence time (residence time = 8 min, molar equivalent ratio = 1, and temperature = 330 K). When the initial guesses are altered, a different local solution is obtained (residence time = 7 min, molar equivalent ratio = 0.63, and temperature = 330 K). SnobFit, on the other hand, requires a higher number of experiments/iterations to fully explore the decision space and find the global solution as shown in Figure 15c.

The optimal conditions identified by the DRL-based agents, based on several hypermeter tuning methods, as well as SnobFit, and Nelder−Mead algorithms were experimentally implemented. The resulting experimental product concentrations are shown against predicted optimal values in Table 6. Overall, the experimental results are consistent with the optimal solutions identified by the model-based self-optimization approaches. As evidenced by the experimental results of the products concentrations, the DRL agents with the proposed hypermeter tuning methods delivered enhanced productivity compared to the standard methods. This consolidates the idea that the DRL agents, with the proposed tuning methods, not only reduce the required experiments but most importantly tend to converge to the global optimal solutions. Interestingly, the best self-optimization performance, showing specifically reduced experimentation costs and increased productivity, was obtained by the DRL agent under the proposed adaptive hypermeter tuning of two parameters, namely the noise and learning rate. These results highlight the great benefits of the new adaptive hyperparameter tuning in the general context of reinforcement learning and more specifically in the self-optimization of flow chemistry systems.

Regarding computation costs, the DRL methods with adaptive noise and learning rate tuning were more computationally intensive compared to traditional optimization methods like Nelder−Mead and SnobFit. Specifically, while the Nelder−Mead method showed quick convergence with a computation time of approximately 54−66 s (for 118−96 iterations), and SnobFit required around 132 s for 250 iterations, the DRL agents with adaptive tuning needed 250−400 s to achieve optimal solutions. Despite the increased computation cost, the DRL agents' ability to dynamically balance exploration and exploitation allow them to reliably converge to global optima and significantly reduce the number of required experiments makes them highly beneficial in complex systems like flow chemistry. Therefore, while the computational cost is higher, the benefits of reduced experimentation, increased accuracy, and the ability to optimize intricate systems far outweigh these costs, reinforcing the value of DRL-based methods for self-optimization tasks.

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

Details of this can be found in Section C of the Supporting Information (SI).

4. CONCLUSIONS

This paper proposed a new self-optimization strategy of a key flow chemistry process based on deep reinforcement learning. While the approach contributes to the development of the next generation plug-and-play technologies, it opens new opportunities for bias free and cost-effective process development and optimization strategies.

A state-of-the-art deep deterministic policy gradient (DDPG) agent was designed and trained to identify the optimal operating conditions that maximize the product concentration of the imine synthesis. The Bayesian optimization-based hyperparameter tuning demonstrated enhanced training performance compared to the traditional trial-and-error, which was evaluated based on the speed of convergence and training smoothness. These training key features were further enhanced using the proposed new adaptive dynamic hyperparameter tuning. To evaluate the effectiveness of different action policies, the DDPG agent was tested using two approaches: the noisy action policy and the ϵ-greedy action policy. Both policies demonstrated excellent performance in terms of computational speed and efficiency. However, the noisy action policy outperformed the ϵ-greedy approach by achieving a better balance between exploration and exploitation, as well as faster convergence to optimal solutions. This suggests that the DDPG agent equipped with the noisy action policy is more effective for this reaction optimization task.

Finally, the performance of the different DDPG agents, obtained based on the proposed architecture and hyperparameter tuning strategies, was validated and compared against state-of-the-art self-optimization methods, namely Nelder–Mead and SnobFit. Clearly, the DDPG agents demonstrated superior capabilities and exhibited enhanced tracking of the global optimal solution. The best performance was shown by the agent based the adaptive hyperparameter tuning which converged to the optimal solution in 60 experiments compared to 250 and up to 118 experiments in the case of SnobFit and Neder-Mead, respectively. Additionally, the experimental validations of the optimal profiles were conclusive and proved consistent with predicted model-based approaches. These key performance indicators underscore the capabilities of the proposed new DRL self-optimization strategies in the minimization of the experimental costs and associated environmental burdens.

The mathematical model identified based on the new experimental data was used as a training environmental to further reduce the experimentation costs. However, the approach can be fully implemented in real time, based solely on the experimentation or on a hybrid approach, which may be demonstrated in a future work. Moreover, the DRL can also be adapted to the multiobjective self-optimization case to help identify the set of the best compromises between different competing criteria.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acsengineeringau.5c00004.

Python packages used for reinforcement learning simulation, experimental data for parameter estimation, method used for parameter estimation, and optimization data for DRL (Bayesian, adaptive noise and adaptive noise and learning rate) (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Brahim Benyahia − *Department of Chemical Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, U.K.;* orcid.org/0000-0002-5397-1498; Email: b.benyahia@lboro.ac.uk

Authors

Ashish Yewale − *Department of Chemical Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, U.K.;* orcid.org/0000-0003-2784-6068

Yihui Yang − *Synthetic Molecule Process Development, Process Engineering and Technology, Takeda Pharmaceuticals International Company, Cambridge, Massachusetts 02139, United States;* orcid.org/0000-0002-0485-7680

Neda Nazemifard − *Synthetic Molecule Process Development, Process Engineering and Technology, Takeda Pharmaceuticals International Company, Cambridge, Massachusetts 02139, United States*

Charles D. Papageorgiou − *Synthetic Molecule Process Development, Process Engineering and Technology, Takeda Pharmaceuticals International Company, Cambridge, Massachusetts 02139, United States;* orcid.org/0000-0001-7959-6289

Chris D. Rielly − *Department of Chemical Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, U.K.;* orcid.org/0000-0002-1110-189X

Complete contact information is available at: https://pubs.acs.org/10.1021/acsengineeringau.5c00004

Author Contributions

CRediT: **Ashish Gyaniram Yewale** conceptualization, data curation, formal analysis, methodology, validation, visualization, writing - original draft; **Yihui Yang** project coordination, industrial feedback, visualization, writing - review & editing; **Neda Nazemifard** project coordination, industrial feedback, visualization, writing - review & editing; **Charles D. Papageorgiou** project coordination, industrial feedback, visualization, writing - review & editing; **Chris D. Rielly** supervision, visualization; **Brahim Beahia** funding acquisition, project administration, resources, supervision, visualization, writing - review & editing.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The authors acknowledge funding received from Takeda Pharmaceuticals International Company through the UK Engineering and Physical Sciences Research Council's (EPSRC) Future Continuous Manufacturing and Advanced Crystallization Research Hub, Grant EP/P006965/1.

■ NOMENCLATURE

A_t	adaptation metric
a_t	actions of the step, t
$C_{s,j}$	concentration of the stock solution of reactant, j
C_{jf}	concentration of each reactant in feed stream reactor, j

C_p	average specific heat capacity
$C_{p,i}$	specific heat capacity of i^{th} species
c_3	Product concentration
dt	time difference for noise
E_a	activation energy
Ep	current episode
$ER_{1/2}$	equivalent ratio
E_θ	expected value based on the policy
G_t	cumulative rewards
HP	hyperparameter
HP_{min}	hyperparameter minimum value
HP_{max}	hyperparameter maximum value
ΔH_{react}	reaction enthalpy
j	reactant species
k	reaction rate constant
k_{ref}	reference reaction rate coefficient
$L(Q^\theta)$	loss function critic network
N_{total}	total number of steps
N_t	training iteration
Q	total flow rate
Q_j	individual reactant flow rate, j
R	universal gas constant
Res_t	residence time
r	reaction rate
r_t	instantaneous reward at the step, t
s_t	state of the step, t
T	temperature inside tubular reactor
T_c	coolant temperature
T_f	feeding temperature
T_0	initial reactor temperature
T_{ref}	reference temperature
t	total reaction time
UA	overall heat transfer coefficient
V	volume of reactor
v_z	total velocity of reactant flow in z-direction
x_i	mole fraction of i^{th} species
z	flow reaction

Greek Letter

α	decay rate
α_{actor}	actor learning rate
α_{critic}	critic learning rate
χ_{at}	current value of noise
ε	greedy action
ε_{min}	lowest probability of choosing a random action
γ	future reward discount factor
λ	exponential decay rate
μ	long-term mean
$\mathcal{N}$	exploration noise
∇_{θ^μ}	actor network gradient
ρ	reaction mixture density
σ	volatility of noise/exploration noise parameter
τ	smoothing factor
θ	noise return rate
θ^μ	actor network parameters
θ^Q	critic network parameters
$\theta^{\mu'}$	target actor network parameters
$\theta^{Q'}$	target critic network parameters

Glossary

DDPG	deep deterministic policy gradient
DNNs	deep neutral networks
DRL	deep reinforcement learning
HP	hyperparameter
HPO	hyperparameter optimization
LSTM	long−short-term memory
MINLP	mixed integer nonlinear programming
ML	machine learning
ODEs	ordinary differential equations
OVAT	one variable at a time
OU	Ornstein−Uhlenbeck
RL	reinforcement learning
SNOBFIT	stable noisy optimization by branch and fit
TD	temporal difference
TS-EMO	Thomson sampling efficient multi-objective

REFERENCES

(1) Teixeira, R. I.; Andresini, M.; Luisi, R.; Benyahia, B. Computer-Aided Retrosynthesis for Greener and Optimal Total Synthesis of a Helicase-Primase Inhibitor Active Pharmaceutical Ingredient. *JACS Au* **2024**, *4* (11), 4263−4272.

(2) Heider, P. L.; Born, S. C.; Basak, S.; Benyahia, B.; Lakerveld, R.; Zhang, H.; Hogan, R.; Buchbinder, L.; Wolfe, A.; Mascia, S.; Evans, J. M. B.; Jamison, T. F.; Jensen, K. F. Development of a Multi-Step Synthesis and Workup Sequence for an Integrated, Continuous Manufacturing Process of a Pharmaceutical. *Org. Process Res. Dev* **2014**, *18* (3), 402−409.

(3) Fysikopoulos, D.; Benyahia, B.; Borsos, A.; Nagy, Z. K.; Rielly, C. D. A Framework for Model Reliability and Estimability Analysis of Crystallization Processes with Multi-Impurity Multi-Dimensional Population Balance Models. *Comput. Chem. Eng.* **2019**, *122*, 275−292.

(4) Benyahia, B. Chapter 6: Applications of a Plant-Wide Dynamic Model of an Integrated Continuous Pharmaceutical Plant: Design of the Recycle in the Case of Multiple Impurities. In *Computer Aided Chemical Engineering*; Elsevier B.V., 2018; Vol. *41*, pp 141−157.

(5) Livingston, A. N.; Mattingly, T. J., II Drug and Medical Device Product Failures and the Stability of the Pharmaceutical Supply Chain. *Journal of the American Pharmacists Association* **2021**, *61* (1), e119−e122.

(6) Nagaich, U.; Sadhna, D. Drug Recall: An Incubus for Pharmaceutical Companies and Most Serious Drug Recall of History. *Int. J. Pharm. Investig* **2015**, *5* (1), 13−19.

(7) Campbell, T. J. S.; Rielly, C. D.; Benyahia, B. Sustainability and Quality-by-Digital Design of an Integrated End-to-End Continuous Pharmaceutical Process. *Comput.-Aided Chem. Eng.* **2024**, *53*, 343−348.

(8) Mascia, S.; Heider, P. L.; Zhang, H.; Lakerveld, R.; Benyahia, B.; Barton, P. I.; Braatz, R. D.; Cooney, C. L.; Evans, J. M. B.; Jamison, T. F.; Jensen, K. F.; Myerson, A. S.; Trout, B. L. End-to-End Continuous Manufacturing of Pharmaceuticals: Integrated Synthesis, Purification, and Final Dosage Formation. *Angew. Chem.* **2013**, *125* (47), 12585−12589.

(9) Benyahia, B.; Lakerveld, R.; Barton, P. I. A Plant-Wide Dynamic Model of a Continuous Pharmaceutical Process. *Ind. Eng. Chem. Res.* **2012**, *51* (47), 15393−15412.

(10) Inuwa, H. M.; Ravi Raja, A.; Kumar, A.; Singh, B.; Singh, S. Status of Industry 4.0 Applications in Healthcare 4.0 and Pharma 4.0. In *Materials Today: Proceedings*; Elsevier Ltd, 2022; Vol. *62*, pp 3593−3598.

(11) Marosi, G.; Hirsch, E.; Bocz, K.; Toldy, A.; Szolnoki, B.; Bodzay, B.; Csontos, I.; Farkas, A.; Balogh, A.; Démuth, B.; Nagy, Z. K.; Pataki, H. Pharmaceutical and Macromolecular Technologies in the Spirit of Industry 4.0. *Periodica Polytechnica Chemical Engineering* **2018**, *62* (4), 457−466.

(12) Arden, N. S.; Fisher, A. C.; Tyner, K.; Yu, L. X.; Lee, S. L.; Kopcha, M. Industry 4.0 for Pharmaceutical Manufacturing: Preparing for the Smart Factories of the Future. *Int. J. Pharm.* **2021**, *602*, No. 120554.

(13) Ezell, S. J. *A Policymaker's Guide to Smart Manufacturing*, 2016. https://api.semanticscholar.org/CorpusID:190504769 (accessed Dec 05, 2024).

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

(14) Williams, W. L.; Zeng, L.; Gensch, T.; Sigman, M. S.; Doyle, A. G.; Anslyn, E. V. The Evolution of Data-Driven Modeling in Organic Chemistry. *ACS Cent Sci.* **2021**, *7* (10), 1622−1637.
(15) Voinarovska, V.; Kabeshov, M.; Dudenko, D.; Genheden, S.; Tetko, I. V. When Yield Prediction Does Not Yield Prediction: An Overview of the Current Challenges. *J. Chem. Inf Model* **2024**, *64* (1), 42−56.
(16) Henson, A. B.; Gromski, P. S.; Cronin, L. Designing Algorithms to Aid Discovery by Chemical Robots. *ACS Cent Sci.* **2018**, *4* (7), 793−804.
(17) Hammer, A. J. S.; Leonov, A. I.; Bell, N. L.; Cronin, L. Chemputation and the Standardization of Chemical Informatics. *JACS Au* **2021**, *1* (10), 1572−1587.
(18) Nelder, J. A.; Mead, R. A Simplex Method for Function Minimization. *Comput. J.* **1965**, *7* (4), 308−313.
(19) Liang, R.; Duan, X.; Zhang, J.; Yuan, Z. Bayesian Based Reaction Optimization for Complex Continuous Gas-Liquid-Solid Reactions. *React. Chem. Eng.* **2022**, *7* (3), 590−598.
(20) Avila, C.; Cassani, C.; Kogej, T.; Mazuela, J.; Sarda, S.; Clayton, A. D.; Kossenjans, M.; Green, C. P.; Bourne, R. A. Automated Stopped-Flow Library Synthesis for Rapid Optimisation and Machine Learning Directed Experimentation. *Chem. Sci.* **2022**, *13* (41), 12087−12099.
(21) Jeraal, M. I.; Holmes, N.; Akien, G. R.; Bourne, R. A. Enhanced Process Development Using Automated Continuous Reactors by Self-Optimisation Algorithms and Statistical Empirical Modelling. *Tetrahedron* **2018**, *74* (25), 3158−3164.
(22) Huyer, W.; Neumaier, A. SNOBFIT - Stable Noisy Optimization by Branch and Fit. *ACM Transactions on Mathematical Software* **2008**, *35* (2), 1−25.
(23) Baumgartner, L. M.; Coley, C. W.; Reizman, B. J.; Gao, K. W.; Jensen, K. F. Optimum Catalyst Selection over Continuous and Discrete Process Variables with a Single Droplet Microfluidic Reaction Platform. *React. Chem. Eng.* **2018**, *3* (3), 301−311.
(24) Fu, Z.; Li, X.; Wang, Z.; Li, Z.; Liu, X.; Wu, X.; Zhao, J.; Ding, X.; Wan, X.; Zhong, F.; Wang, D.; Luo, X.; Chen, K.; Liu, H.; Wang, J.; Jiang, H.; Zheng, M. Optimizing Chemical Reaction Conditions Using Deep Learning: A Case Study for the Suzuki-Miyaura Cross-Coupling Reaction. *Organic Chemistry Frontiers* **2020**, *7* (16), 2269−2277.
(25) Park, S.; Han, H.; Kim, H.; Choi, S. Machine Learning Applications for Chemical Reactions. *Chem. Asian J.* **2022**, *17* (14), No. e202200203.
(26) Christensen, M.; Yunker, L. P. E.; Adedeji, F.; Häse, F.; Roch, L. M.; Gensch, T.; dos Passos Gomes, G.; Zepel, T.; Sigman, M. S.; Aspuru-Guzik, A.; Hein, J. E. Data-Science Driven Autonomous Process Optimization. *Commun. Chem.* **2021**, *4* (1), 112.
(27) Frazier, P. I. A Tutorial on Bayesian Optimization. *arXiv* **2018**.
(28) Shields, B. J.; Stevens, J.; Li, J.; Parasram, M.; Damani, F.; Alvarado, J. I. M.; Janey, J. M.; Adams, R. P.; Doyle, A. G. Bayesian Reaction Optimization as a Tool for Chemical Synthesis. *Nature* **2021**, *590* (7844), 89−96.
(29) Schweidtmann, A. M.; Clayton, A. D.; Holmes, N.; Bradford, E.; Bourne, R. A.; Lapkin, A. A. Machine Learning Meets Continuous Flow Chemistry: Automated Optimization towards the Pareto Front of Multiple Objectives. *Chemical Engineering Journal* **2018**, *352*, 277−282.
(30) Clayton, A. D.; Pyzer-Knapp, E. O.; Purdie, M.; Jones, M. F.; Barthelme, A.; Pavey, J.; Kapur, N.; Chamberlain, T. W.; Blacker, A. J.; Bourne, R. A. Bayesian Self-Optimization for Telescoped Continuous Flow Synthesis. *Angew. Chem., Int. Ed.* **2023**, *62* (3), No. e202214511.
(31) Angello, N. H.; Rathore, V.; Beker, W.; Wołos, A.; Jira, E. R.; Roszak, R.; Wu, T. C.; Schroeder, C. M.; Aspuru-Guzik, A.; Grzybowski, B. A.; Burke, M. D. Closed-Loop Optimization of General Reaction Conditions for Heteroaryl Suzuki-Miyaura Coupling. *Science (1979)* **2022**, *378*, 399−405.
(32) Amar, Y.; Schweidtmann, A. M.; Deutsch, P.; Cao, L.; Lapkin, A. Machine Learning and Molecular Descriptors Enable Rational Solvent Selection in Asymmetric Catalysis. *Chem. Sci.* **2019**, *10* (27), 6697−6706.
(33) Bradford, E.; Schweidtmann, A. M.; Lapkin, A. Efficient Multiobjective Optimization Employing Gaussian Processes, Spectral Sampling and a Genetic Algorithm. *Journal of Global Optimization* **2018**, *71* (2), 407−438.
(34) Taylor, C. J.; Pomberger, A.; Felton, K. C.; Grainger, R.; Barecka, M.; Chamberlain, T. W.; Bourne, R. A.; Johnson, C. N.; Lapkin, A. A. A Brief Introduction to Chemical Reaction Optimization. *Chem. Rev.* **2023**, *123* (6), 3089−3126.
(35) Felton, K. C.; Rittig, J. G.; Lapkin, A. A. Summit: Benchmarking Machine Learning Methods for Reaction Optimisation. *Chemistry-Methods* **2021**, *1* (2), 116−122.
(36) Häse, F.; Roch, L. M.; Aspuru-Guzik, A. Chimera: Enabling Hierarchy Based Multi-Objective Optimization for Self-Driving Laboratories. *Chem. Sci.* **2018**, *9* (39), 7642−7655.
(37) Kershaw, O. J.; Clayton, A. D.; Manson, J. A.; Barthelme, A.; Pavey, J.; Peach, P.; Mustakis, J.; Howard, R. M.; Chamberlain, T. W.; Warren, N. J.; Bourne, R. A. Machine Learning Directed Multi-Objective Optimization of Mixed Variable Chemical Systems. *Chemical Engineering Journal* **2023**, *451*, No. 138443.
(38) Arulkumaran, K.; Deisenroth, M. P.; Brundage, M.; Bharath, A. A. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Process Mag* **2017**, *34* (6), 26−38.
(39) Zhang, F.; Leitner, J.; Milford, M.; Upcroft, B.; Corke, P. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control. *arXiv* **2015**.
(40) Kulhánek, J.; Derner, E.; de Bruin, T.; Babuška, R. In Vision-Based Navigation Using Deep Reinforcement Learning, European Conference on Mobile Robots (ECMR); IEEE, 2019; pp 1−8.
(41) Tammewar, A.; Chaudhari, N.; Saini, B.; Venkatesh, D.; Dharahas, G.; Vora, D.; Patil, S.; Kotecha, K.; Alfarhood, S. Improving the Performance of Autonomous Driving through Deep Reinforcement Learning. *Sustainability* **2023**, *15* (18), 13799.
(42) Chen, Y.; Ji, C.; Cai, Y.; Yan, T.; Su, B. Deep Reinforcement Learning in Autonomous Car Path Planning and Control: A Survey. *arXiv* **2024**.
(43) Mocanu, E.; Mocanu, D. C.; Nguyen, P. H.; Liotta, A.; Webber, M. E.; Gibescu, M.; Slootweg, J. G. On-Line Building Energy Optimization Using Deep Reinforcement Learning. *IEEE Transactions on Smart Grid* **2019**, *10* (4), 3698−3708.
(44) Vamvakas, D.; Michailidis, P.; Korkas, C.; Kosmatopoulos, E. Review and Evaluation of Reinforcement Learning Frameworks on Smart Grid Applications. *Energies (Basel)* **2023**, *16* (14), 5326.
(45) Abishu, H. N.; Seid, A. M.; Márquez-Sánchez, S.; Fernandez, J. H.; Corchado, J. M.; Erbad, A. In Multi-Agent DRL-Based Multi-Objective Demand Response Optimization for Real-Time Energy Management in Smart Homes, 20th International Wireless Communications and Mobile Computing Conference, IWCMC, 2024; pp 1210−1217.
(46) Benyahia, B.; Anandan, P. D.; Rielly, C. In Robust Model-Based Reinforcement Learning Control of a Batch Crystallization Process, 9th International Conference on Systems and Control, ICSC, 2021; pp 89−94.
(47) Meng, Q.; Anandan, P. D.; Rielly, C.; Benyahia, B. In Multi-Agent Reinforcement Learning and RL-Based Adaptive PID Control of Crystallization Processes, 33rd European Symposium on Computer Aided Process Engineering ESCAPE33, 2023; pp 1667−1672.
(48) Li, C.; Zheng, P.; Yin, Y.; Wang, B.; Wang, L. Deep Reinforcement Learning in Smart Manufacturing: A Review and Prospects. *CIRP J. Manuf Sci. Technol.* **2023**, *40*, 75−101.
(49) Geurtsen, M.; Adan, I.; Atan, Z. Deep Reinforcement Learning for Optimal Planning of Assembly Line Maintenance. *J. Manuf Syst* **2023**, *69*, 170−188.
(50) Li, H.; Collins, C. R.; Ribelli, T. G.; Matyjaszewski, K.; Gordon, G. J.; Kowalewski, T.; Yaron, D. J. Tuning the Molecular Weight Distribution from Atom Transfer Radical Polymerization Using Deep Reinforcement Learning. *Mol. Syst. Des Eng.* **2018**, *3* (3), 496−508.

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266

(51) Ma, Y.; Zhu, W.; Benton, M. G.; Romagnoli, J. Continuous Control of a Polymerization System with Deep Reinforcement Learning. *J. Process Control* **2019**, *75*, 40−47.
(52) Zhou, Z.; Li, X.; Zare, R. N. Optimizing Chemical Reactions with Deep Reinforcement Learning. *ACS Cent Sci.* **2017**, *3* (12), 1337−1344.
(53) Neumann, M.; Palkovits, D. S. Reinforcement Learning Approaches for the Optimization of the Partial Oxidation Reaction of Methane. *Ind. Eng. Chem. Res.* **2022**, *61* (11), 3910−3916.
(54) Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. In Deep Reinforcement Learning That Matters, Proceedings of the AAAI Conference on Artificial Intelligence, 2017; pp 3207−3214.
(55) Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; Abbeel, P. In Benchmarking Deep Reinforcement Learning for Continuous Control, Proceedings of the 33rd International Conference on Machine Learning, 2016.
(56) Popova, M.; Isayev, O.; Tropsha, A. Deep Reinforcement Learning for de Novo Drug Design. *Sci. Adv.* **2018**, *4* (7), No. eaap7885.
(57) Granda, J. M.; Donina, L.; Dragone, V.; Long, D. L.; Cronin, L. Controlling an Organic Synthesis Robot with Machine Learning to Search for New Reactivity. *Nature* **2018**, *559* (7714), 377−381.
(58) Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. In Continuous Control with Deep Reinforcement Learning, Published at ICLR 2016, 2015.
(59) Benyahia, B.; Anandan, P. D.; Rielly, C. Control of Batch and Continuous Crystallization Processes Using Reinforcement Learning. In *Computer Aided Chemical Engineering*; Elsevier B.V., 2021; Vol. *50*, pp 1371−1376.
(60) Tatlidil, D.; Raza, M. A.; Dege, N.; Agar, A. A.; Farwa, U.; Rehman, S. U. Therapeutical Potential of Imines; Synthesis, Single Crystal Structure, Computational, Molecular Modeling, and ADMET Evaluation. *ACS Omega* **2022**, *7* (12), 10568−10579.
(61) Dzeikala, A.; Sykula, A. Schiff Bases as Important Class of Pharmacological Agents. *J. Pharm. Pharmacol.* **2018**, *6* (12), 989−1009.
(62) Sutton, R. S.; Barto, A. G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press Cambridge: Massachusetts, London, England, 2015.
(63) Sutton, R. S.; Maei, H. R.; Precup, D.; Bhatnagar, S.; Silver, D.; Szepesvári, C.; Wiewiora, E. In Fast Gradient-Descent Methods for Temporal-Difference Learning with Linear Function Approximation, Proceedings of the 26th International Conference on Machine Learning, 2009; pp 993−1000.
(64) Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**.
(65) Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; Hassabis, D. Human-Level Control through Deep Reinforcement Learning. *Nature* **2015**, *518* (7540), 529−533.
(66) Lecun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521* (7553), 436−444.
(67) Anandan, P. D.; Rielly, C. D.; Benyahia, B. Optimal Control Policies of a Crystallization Process Using Inverse Reinforcement Learning. In *Computer Aided Chemical Engineering*; Elsevier B.V., 2022; Vol. *51*, pp 1093−1098.
(68) Alavizadeh, H.; Alavizadeh, H.; Jang-Jaccard, J. Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection. *Computers* **2022**, *11*, 41.
(69) Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; Hassabis, D. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* **2016**, *529* (7587), 484−489.
(70) Eimer, T.; Lindauer, M.; Raileanu, R. In Hyperparameters in Reinforcement Learning and How To Tune Them, Proceedings of the 40th International Conference on Machine Learning, PMLR, 2023; pp 1−46.
(71) Hussenot, L.; Andrychowicz, M.; Vincent, D.; Dadashi, R.; Raichuk, A.; Stafiniak, L.; Girgin, S.; Marinier, R.; Momchev, N.; Ramos, S.; Orsini, M.; Bachem, O.; Geist, M.; Pietquin, O. In Hyperparameter Selection for Imitation Learning, Proceedings of the 38th International Conference on Machine Learning, PMLR, 2021; Vol. *139*, pp 4511−4522.
(72) Hutter, F.; Hoos, H.; Leyton-Brown, K. In An Efficient Approach for Assessing Hyperparameter Importance, Proceedings of the 31st International Conference on Machine Learning, ICML, 2014; pp 754−762.
(73) Watanabe, S. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. *arXiv* **2023**.
(74) Uhlenbeck, G. E.; Ornstein, L. S. On the Theory of the Brownian Motion. *Physical review* **1930**, *36*, 823−841.
(75) Dai, J.; Liu, P.; Qu, Q.; Li, L.; Niu, T. Aerodynamic Optimization of High-Lift Devices Using a 2D-to-3D Optimization Method Based on Deep Reinforcement Learning and Transfer Learning. *Aerosp. Sci. Technol.* **2022**, *121*, No. 107348.

https://doi.org/10.1021/acsengineeringau.5c00004
ACS Eng. Au 2025, 5, 247−266