Journal Pre-proofs

Article

Actor–Critic Reinforcement Learning and Application in Developing Computer-Vision-Based Interface Tracking

Oguzhan Dogru, Kirubakaran Velswamy, Biao Huang

PII: S2095-8099(21)00326-X
DOI: https://doi.org/10.1016/j.eng.2021.04.027
Reference: ENG 777

To appear in: *Engineering*

Received Date: 16 July 2020
Revised Date: 4 November 2020
Accepted Date: 2 April 2021

Please cite this article as: O. Dogru, K. Velswamy, B. Huang, Actor–Critic Reinforcement Learning and Application in Developing Computer-Vision-Based Interface Tracking, *Engineering* (2021), doi: https://doi.org/10.1016/j.eng.2021.04.027

Research
AI Energizes Process Manufacturing—Article

Actor–Critic Reinforcement Learning and Application in Developing Computer-Vision-Based Interface Tracking

Oguzhan Dogru [#], Kirubakaran Velswamy [#], Biao Huang *
Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada

* Corresponding author.
E-mail address: biao.huang@ualberta.ca (B. Huang)
[#] These authors contributed equally to this work.

ARTICLE INFO
Article history:
Received 16 July 2020
Revised 4 November 2020
Accepted 2 April 2021
Available

Keywords:
Interface tracking
Object tracking
Occlusion
Reinforcement learning
Uniform manifold approximation and projection

ABSTRACT
This paper synchronizes control theory with computer vision by formalizing object tracking as a sequential decision-making process. A reinforcement learning (RL) agent successfully tracks an interface between two liquids, which is often a critical variable to track in many chemical, petrochemical, metallurgical, and oil industries. This method utilizes less than 100 images for creating an environment, from which the agent generates its own data without the need for expert knowledge. Unlike supervised learning (SL) methods that rely on a huge number of parameters, this approach requires far fewer parameters, which naturally reduces its maintenance cost. Besides its frugal nature, the agent is robust to environmental uncertainties such as occlusion, intensity changes, and excessive noise. From a closed-loop control context, an interface location-based deviation is chosen as the optimization goal during training. The methodology showcases RL for real-time object-tracking applications in the oil sands industry. Along with a presentation of the interface tracking problem, this paper provides a detailed review of one of the most effective RL methodologies: actor–critic policy.

1. Introduction

Oil sands ore contains bitumen, water, and minerals. Bitumen is a high-viscosity hydrocarbon mixture, which can be extracted by means of several chemical and physical processes. The product is further treated in upgrader units or refineries [1] to obtain more valuable byproducts (e.g., gasoline, jet fuel). Oil sands are mined from open pits and loaded into trucks to be moved into the crushers [2]. Following this, the mixture is treated with hot water for hydro-transportation to the extraction plant. Aeration and several chemicals are introduced to enhance this process. In the extraction plant, the mixture is settled down in a *primary separation vessel* (PSV). A water-based oil sands separation process is summarized in Fig. 1.

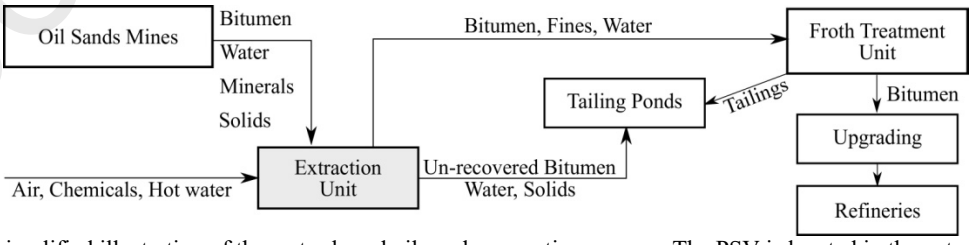

Fig. 1. A simplified illustration of the water-based oil sands separation process. The PSV is located in the extraction unit.

During the separation process inside the PSV, three layers are formed: froth, middlings, and tailings (Fig. 2). An *interface* (referred to as the froth–middlings interface (FMI) henceforth) is formed between the froth and middlings layer. Its level with reference to the PSV unit influences the quality of the extraction.

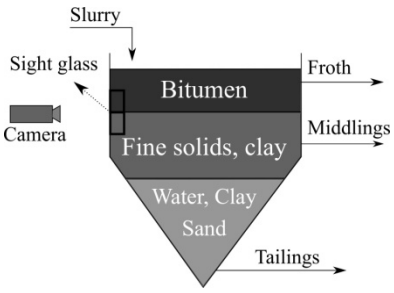

Fig. 2. A schematic of the PSV. During the separation process, three layers are formed. The camera is used to monitor the interface between the middlings and the froth layers in order to control the FMI level optimally.

To control the FMI level, it is crucial to have reliable sensors. Traditionally, differential pressure (DP) cells, capacitance probes, or nucleonic density profilers are used to monitor the FMI level. However, these are either inaccurate or reported to be unreliable [3]. Sight glasses are used to manually monitor the interface for any process abnormalities. To utilize this observation in closed-loop control, Ref. [3] proposed using a camera as a sensor. This scheme utilizes an edge detection model with particle filtering on the images to obtain the FMI level; feedback control is then established using this model. More recently, Ref. [4] combined edge detection with dynamic frame differencing to detect the interface. This method directly uses the edge detection technique to detect the interface, along with a frame-comparison mechanism that estimates the quality of the measurement; it also detects faults. Ref. [5] used a mixture of Gaussian distributions to model the appearances of the froth, interface, and middlings, and predicted the interface using a spatiotemporal Markov random field. Despite addressing several challenges utilizing models based on the appearance or behavior of the interface, these techniques fail to address the sensitivities to uncertain environmental conditions, such as occlusion and excessive/non-Gaussian noise.

Supervised learning (SL) methods try to build a map from input (i.e., image, x) to output (i.e., label, y) data by minimizing a cost (or loss) function. Usually, the cost function is convex, and the optimal parameters are calculated by applying a stochastic gradient descent algorithm [6,7] to the cost function. *Unsupervised learning* (UL) methods, on the other hand, are used to find the hidden features in the unlabeled data (i.e., uses x only) [8]. The goal is usually to compress the data or to find similarities within the data. Nevertheless, UL techniques do not consider the impact of the input on the output, even if such a causal relationship exists. In computer vision, these methods are implemented using convolutional neural networks (CNNs). A CNN is a parametric function that applies convolutional operation on the inputs. It can extract abstract features by processing not just a pixel, but also its neighboring pixels. It is used for classification, regression, dimensionality reduction, and so forth [9–12]. Even though they have been used for decades [13–16], CNNs have only lately gained significant popularity in different domains [17–20]. This is due to the developments that have occurred in hardware technology [21] and data availability [22]. Parallel to the developments in computer vision, a recurrent neural network (RNN) is used for time-series prediction, where the previous output of the network is fed back into itself [23] in what can be considered a recursive matrix multiplication. However, vanilla RNN [24] suffers from diminishing or exploding gradients, because it repeatedly feeds the previous information back into itself, leading to uneven back-propagated data sharing in between hidden layers. Therefore, it tends to fail when the data sequence is arbitrarily long. To overcome this issue, more complex networks such as long short-term memory (LSTM) [25] and gated recurrent units [26] have been proposed. These networks facilitate data transfer in between hidden layers to make the learning more efficient. More recently, a variant of LSTM called convolutional LSTM (ConvLSTM) [27] was reported to improve LSTM performance by replacing matrix multiplications with convolutional operations. Unlike fully connected LSTM, ConvLSTM receives an image rather than one-dimensional data; it utilizes spatial connections that are present within the input data and enhances estimation. Networks with many layers are considered to be *deep structures* [28]. Various deep architectures have been proposed [29–33] to enhance the prediction accuracy even further. However, these structures suffer from over-parameterization (i.e., the number of training data points is less than the number of parameters). Several regularization techniques (e.g., dropout, L2) [17] and transfer learning (also called fine-tuning) methods [34,35] try to find a work-around to improve the network's performance. However, the transferred information (e.g., network parameters) may not be general enough for the target domain. This issue becomes significant, especially when the training data is insufficient or their statistics are significantly different than the data in the target domain. Moreover, efficient transfer learning for recurrent networks currently remains as an opportunity for further research.

Reinforcement learning (RL) [36] combines the advantages of both SL and UL techniques and formalizes the learning process as a Markov decision process (MDP). Inspired by animal psychology [37] and optimal control [38–43], this learning scheme involves an intelligent agent (i.e., the controller). Unlike SL or UL methods, RL does not rely on an offline or batch dataset, but generates its own data by interacting with the environment. It evaluates the impacts of its actions by considering immediate consequences and predicts the value via *roll-out*. Hence, it is more suitable for real or continuous

processes involving decision-making for complex systems. However, in sampled data-based schemes, data distribution may be significantly different during training, which may cause high variance of estimations [36]. *Actor–critic methods* have been proposed [44–46] in order to combine the advantages of value estimation and the policy gradient. This approach segregates the agent into two parts: the *actor* decides which action to take, while the *critic* estimates the goodness of that action using an action-value [47] or state-value [48] function. These methods do not rely on any labels or system models. Therefore, exploration of the state or action space is an important factor that affects the agent's performance. In system identification [49–51], this is known as the identification problem. Various methods have been developed to address the exploration issue [36,48,52–58]. As a subfield of machine learning [59–61], RL is used in—but not limited to—process control [2,42,61–68], the game industry [69–77], robotics, and autonomous vehicles [78–81].

FMI tracking can be formulated as an object-tracking problem, which can be solved in one or two steps using detection-free or detection-based tracking approaches, respectively. Previous works [82–84] have used RL for object detection or localization, for which it can be combined with a tracking algorithm. In the case of such a combination, the tracking algorithm also needs to be reliable and fast for real-time implementation. Several object-tracking algorithms have been proposed, including multiple object-tracking algorithms using RL [85–90]. The proposed schemes combine pre-trained object detection with RL-based tracking or a supervised tracking solution. These simulations were carried out under ideal conditions [91,92]. The performance of object-detection-based methods often depends on the detection accuracy. Even if the agent learns to track based on a well-defined reward signal, the researcher should ensure that the sensory information is (or the features of the sensory information are) accurate. Model-based algorithms often assume that the object of interest has a rigid or a non-rigid shape [4] and that the noise or the motion has a particular pattern [3]. These assumptions may not hold when unexpected events occur. Therefore, a model-free approach may provide a more general solution.

Since a CNN may extract abstract features, it is important to analyze it after training. Common analysis techniques utilize the information of the activation functions, kernels, intermediate layers, saliency maps, and so forth [30,93–95]. In an RL context, a popular approach has been to reduce the dimensions of the observed features using *t*-distributed stochastic neighbor embedding (*t*-SNE) [96] to visualize the agent in different states [72,97,98]. This helps to cluster the behavior with respect to the different situations encountered by the agent. Another dimensionality-reduction technique—namely, uniform manifold approximation and projection (UMAP) [99]—projects the high-dimensional input (which may not be meaningful in the Euclidean space) into Riemannian space. In this way, the dimensionality of nonlinear features can be reduced.

Fig. 3 illustrates a general control hierarchy in the process industry. In a continuous process, each level in the hierarchy interacts with each other at different sampling frequencies. Interaction starts at the instrumentation level, which affects the upper levels significantly. Recently, Ref. [2] proposed a solution for the execution level. However, addressing other levels remains challenging.

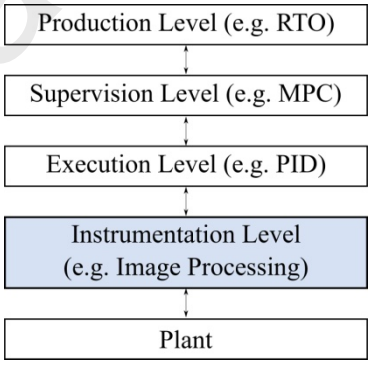

Fig. 3. A general control hierarchy in the process industry. RTO: real-time optimization; MPC: model predictive control; PID: proportional-integral-derivative controller.

Here, we propose a novel interface tracking scheme based on RL that is trained for a model-free sequential decision-making agent. This work:

- Provides a detailed review of actor–critic algorithms;
- Focuses on the instrumentation level to improve the overall performance of the hierarchy;
- Formulates interface tracking as a model-free sequential decision-making process;
- Combines CNN and LSTM to extract spatiotemporal features without any explicit models or unrealistic assumptions;
- Utilizes DP cell measurements in a reward function without any labels or human intervention;
- Trains the agent using temporal difference learning that allows the agent to learn continuously in a closed-loop control

setting;
- Validates robustness amidst uncertainties in an open-loop setting;
- Analyzes the agent's beliefs in a reduced feature space.

This paper is organized as follows: Section 2 provides a review on actor–critic algorithms and preliminary information, interface detection is formulated in Section 3, Section 4 presents the training and test results in detail, and conclusions and future work are drawn in Sections 5 and 6, respectively.

2. Review of actor–critic reinforcement learning

RL is a rigorous mathematical concept [36,39,42] in which an agent learns a behavior that maximizes an overall return in a dynamic environment. Similar to a human being, the agent learns how to make intelligent decisions by considering the future rewards. This implies contemplating temporal aspects of the observations, unlike simple classification or regression approaches. This ability allows RL to be used under uncertain conditions [40] with irregular sampling rates. Its versatile nature makes RL adaptive to different environmental conditions and allows it to be transferred from simulation environments to real processes [80].

2.1. Markov decision processes

An MDP formulates discrete sequential decision-making processes via a tuple, M, that consists of $\langle X,\ U,\ R,\ P,\ \gamma \rangle$, where $x \in X$, $u \in U$, $r \in R \subset \mathbb{R}$ are the state, action, and reward, respectively. $P(x',\ r|x,\ u)$ represents the system dynamics, or state transition probabilities, which may be deterministic or stochastic. It satisfies the Markov property [100]—that is, the future state depends solely on the current state, and does not depend on the history prior to that. In this work, system dynamics are unknown to the agent in order to make this approach more general. The discount factor $\gamma \in [0,\ 1)$ is a weight for future rewards in order to make their summation bounded. The stochastic policy, $\pi(u|x)$, is a mapping from the observed system states to the actions.

In an MDP, the agent observes a state $x_0 \sim \sigma_0$, where σ_0 represents the distribution of the initial states. It then selects an action $u \sim \pi(u|x)$ that carries the agent to a next state, $x' \sim P(x',\ r|x,\ u)$, and yields a reward, $r \sim P(x',\ r|x,\ u)$. By utilizing the sequence (i.e., x, u, r, x'), the agent learns a policy, π, that leads to maximizing the discounted return, G, as defined in Eq. (1) [36]:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad (1)$$

where t represents a discrete timestep. The state-value, $v_\pi(x)$, and action-value, $q_\pi(x, u)$, functions are calculated using the Bellman equations (Eqs. (2) and (3)):

$$v_\pi(x) = \mathbb{E}_\pi\left[G_t|X_t = x\right] = \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1}|X_t = x\right]$$
$$= \Sigma_u \pi(u|x) \Sigma_{x'} \Sigma_r Pr(x',\ r|x,\ u)\left[r + \gamma v_\pi(x')\right], \qquad (2)$$
$$\forall x \in X$$

$$q_\pi(x,\ u) = \mathbb{E}_\pi\left[G_t|X_t = x,\ U_t = u\right]$$
$$= \Sigma_{x'} \Sigma_r Pr(x',\ r|x,\ u)\left[r + \gamma \Sigma_{u'} \pi(u'|x') q_\pi(x',\ u')\right], \qquad (3)$$
$$\forall x,\ u \in X \times U$$

After the value functions are estimated for each state, the optimal value ($v_\pi^*(x)$ and $q_\pi^*(x,\ u)$) functions can be found using Eqs. (4) and (5):

$$v_\pi^*(x) = \max_\pi v_\pi(x),\ \forall x \in X \qquad (4)$$

$$q_\pi^*(x,\ u) = \max_\pi q_\pi(x,\ u)$$
$$= \mathbb{E}\left[R_{t+1} + \gamma v^*(X_{t+1})|X_t = x,\ U_t = u\right], \qquad (5)$$
$$\forall x,\ u \in X \times U$$

Following that, the optimal policy, π^*, can be found as follows:

$$\pi^*(x) = \arg\max_u q_\pi^*(x,\ u) \qquad (6)$$

For largescale problems, linear or nonlinear function approximation techniques can be used to find approximate value

functions by means of $\hat{Q}(x, u|\omega)$, $\hat{V}(x|\omega)$, or both, where ω represents the parameters of the approximation. These structures are also called *critics*. This work focuses on the state-value estimation and simplifies its notation as $V(\cdot)$.

2.2. A review of actor–critic algorithms

Earlier approaches used value-based (critic-only) RL [71,101] to solve control problems. In these approaches, actions are derived directly from a value function, which has been reported to be divergent for largescale problems [45,102]. Policy-based (actor-only) methods [103–105] tackle this problem and can learn stochastic behaviors by generating a policy directly from a parameterized function. This function is then directly optimized by using a performance metric. However, variance of the estimation and the extended learning time make the policy gradient impractical. Similar to generative adversarial networks (GANs) [106], which utilize generative and discriminative networks, actor–critic algorithms self-supervise without any labels [44,45,107,108]. These techniques combine policy and value-based methods via an actor and a critic, respectively. This assisted estimation reduces the variance significantly and helps in learning the optimal policy [36,55]. The actor and the critic can be represented as two neural networks, $\pi(u|x, \theta)$ and $V(x|\omega)$ (or $Q(x, u|\omega)$), respectively.

Although several model-based actor–critic schemes have been proposed [109,110], this paper focuses on the most commonly used model-free algorithms, as represented in Table 1. Some of these methods use entropy regularization, whereas the others take advantage of heuristic methods. A common example for these methods is the ε-greedy approach, in which the agent takes a random action with a probability $\varepsilon \in [0, 1)$. Other exploration techniques include—but are not limited to—introducing additive noise to the action space, introducing noise to the parameter space, and utilizing an upper confidence bound. Interested readers can see Ref. [67] for more detail.

Table 1
A comparison of actor–critic algorithms based on the type of action spaces and the exploration method. The state space can be either discrete or continuous for all the algorithms.

Algorithm	Action space	Exploration
DDPG	Continuous	Noisy actions
A2C or A3C	Discrete/continuous	Entropy regularization
ACER	Discrete/continuous	Entropy regularization
PPO	Discrete/continuous	N/A
ACKTR	Discrete/continuous	N/A
SAC	Continuous	Entropy regularization
TD3	Continuous	Noisy actions

DDPG: deep deterministic policy gradient; A2C: advantage actor–critic; A3C: asynchronous advantage actor–critic; ACER: actor–critic with experience replay; PPO: proximal policy optimization; ACKTR: actor–critic using Kronecker-factored trust region; SAC: soft actor–critic; TD3: twin delayed deep deterministic policy gradient.

The actor–critic algorithms are summarized as follows.

2.2.1. Deep deterministic policy gradient

This algorithm has been proposed to generalize discrete, low-dimensional value-based approaches [71] to continuous action spaces. The deep deterministic policy gradient (DDPG) [47] utilizes an actor and a critic (Q) as well as a target critic (Q') network, which is a copy of the critic network. After observing a state, real-valued actions are sampled from the actor network and are mixed with a random process (e.g., the Ornstein–Uhlenbeck process) [111] to encourage exploration. The agent stores state, action, and reward samples in an experience replay buffer to break the correlation between consecutive samples in order to improve learning. It minimizes the mean square error of the loss function to optimize its critic, as shown in Eq. (7).

$$L = R_t + \gamma Q'(X_{t+1}, U_{t+1}) - Q(X_t, U_t) \qquad (7)$$

The scheme utilizes a policy gradient to improve the actor network. Since the value function is learned for the target policy based on a different behavior policy, DDPG is an *off-policy* method.

2.2.2. Asynchronous advantage actor–critic

Instead of storing the experience in a replay buffer that requires memory, the asynchronous advantage actor–critic (A2C/A3C) scheme [48] involves local workers that interact with their environments and update a global network asynchronously, which inherently increases exploration. Instead of minimizing the error based on the Q function, this scheme minimizes the mean square error of the advantage function (A or δ) for the critic update, as shown in Eq. (8).

$$A = \delta = R_t + V\left(X_{t+1}\right) - V\left(X_t\right) \qquad (8)$$

In this scheme, the global critic is updated by using Eq. (9), and the entropy of the policy is used as a regularizer in the actor loss function to increase exploration, as shown in Eq. (10):

$$\mathrm{d}\omega_{\mathrm{G}} \leftarrow \mathrm{d}\omega_{\mathrm{G}} + \alpha_{\mathrm{c}} \nabla_{\omega_{\mathrm{L}}} \delta\left(x_t \middle| \omega_{\mathrm{L}}\right)^2 \qquad (9)$$

$$\mathrm{d}\theta_{\mathrm{G}} \leftarrow \mathrm{d}\theta_{\mathrm{G}} + \alpha_{\mathrm{a}} \nabla_{\theta_{\mathrm{L}}} \delta\left(x_t \middle| \omega_{\mathrm{L}}\right) \ln \pi\left(u_t \middle| x_t,\ \theta_{\mathrm{L}}\right) + \beta \pi\left(u_t \middle| x_t,\ \theta_{\mathrm{L}}\right) \ln \pi\left(u_t \middle| x_t,\ \theta_{\mathrm{L}}\right) \qquad (10)$$

where initially $\mathrm{d}\theta_G = \mathrm{d}\omega_G = 0$. A left arrow (←) represents the update operation; α_{c} and α_{a} are the learning rates for the critic and actor, respectively; ∇ is the derivative with respect to its subscript; and β is a fixed entropy term that is used to encourage exploration. Subscripts L and G stand for the local and global networks, respectively. Multiple workers (A3C) can be used in an offline manner, and the scheme can be reduced to a single worker (A2C) to be implemented online. Even though the workers are independent, they predict the value function based on the behavior policy of the global network, which makes A3C an *on-policy* method. This work utilizes an A3C algorithm to track the interface.

2.2.3. Actor–critic with experience replay

The actor–critic with experience replay (ACER) method [112] addresses the sample inefficiency of the A3C by utilizing a Retrace algorithm [113], which estimates Eq. (11):

$$Q^{\mathrm{ret}}\left(s_t,\ a_t\right) = R_t + \gamma \overline{\eta}_{t+1}\left[Q^{\mathrm{ret}}\left(X_{t+1},\ U_{t+1}\right) - Q\left(X_{t+1},\ U_{t+1}\right)\right] + \gamma V\left(X_{t+1}\right) \qquad (11)$$

where the truncated importance weight, $\overline{\eta}_t = \min\left\{c,\ \rho_t\right\}$, $\eta_t = \left[\pi\left(U_t \middle| X_t\right)\right] \middle/ \left[\mu\left(U_t \middle| X_t\right)\right]$, and c is a clipping constant. μ and π are the behavior and the target policies, respectively. Moreover, this scheme utilizes stochastic dueling networks (to estimate both V and Q in a consistent way) and a trust region policy optimization (TRPO) method that is more efficient than the previous method [114]. Because of its Retrace algorithm, ACER is an off-policy method.

2.2.4. Proximal policy optimization

The proximal policy optimization (PPO) method [115] improves TRPO [114] by clipping the surrogate objective function, as shown in Eq. (12):

$$J^{\mathrm{CLIP}}\left(\theta\right) = \mathbb{E}\left\{\min\left[r\left(\theta\right) A_{\theta_{\mathrm{old}}}\left(x,\ u\right),\ \mathrm{clip}\left(r\left(\theta\right), 1-\varepsilon, 1+\varepsilon\right) A_{\theta_{\mathrm{old}}}\left(x,\ u\right)\right]\right\} \qquad (12)$$

where θ represents the policy parameters (i.e., θ_{old} represents the old policy parameters), $r\left(\theta\right) = \left[\pi_\theta\left(u \middle| x\right)\right] \middle/ \left[\pi_{\theta_{\mathrm{old}}}\left(u \middle| x\right)\right]$, and ε is the clipping constant. A is the advantage estimate that represents how beneficial the agent's actions are, as shown in Eq. (8).

2.2.5. Actor–critic using Kronecker-factored trust region

Instead of a gradient descent [6] algorithm to optimize the actor and critic networks, the actor–critic using Kronecker-factored trust region (ACKTR) [116] utilizes second-order optimization, which provides more information. It overcomes the computational complexity by using Kronecker-factored approximation [117,118] to approximate the inverse of the Fisher information matrix (FIM), which otherwise scales exponentially with respect to the parameters of the approximation. Moreover, it keeps track of the Fisher statistics, which yields better curvature estimates.

2.2.6. Soft actor–critic

Unlike methods that use the entropy of the policy as a loss regularizer [48,114,115,119], the soft actor–critic (SAC) method [55,120] augments the reward function with the entropy term (as shown in Eq. (13)) to encourage exploration. This approach has also been reported [120] to improve the robustness of the policy against model errors.

$$J\left(\theta\right) = \Sigma_{t \in T} \mathbb{E}_{\left(x_t,\ u_t\right) \sim \pi}\left\{R\left(x_t,\ u_t\right) + \alpha H\left[\pi_\theta\left(\cdot\right)\right]\right\} \qquad (13)$$

where θ represents the parameters of the policy, α is a user-defined (fixed or time-varying) weight to adjust the contribution of the entropy, and $H = \mathbb{E}[-\log \pi(\cdot)]$. This scheme relies on both the Q and V functions to utilize the soft-policy iteration. Similar to DDPG and ACER, SAC stores the transitions in a replay buffer to address sample efficiency. Besides enhancing the exploration, entropy maximization compensates for stability loss, which is introduced by the off-policy approach.

2.2.7. Twin delayed deep deterministic policy gradient

The twin delayed deep deterministic policy gradient (TD3) [121] addresses error propagation (which is a non-trivial challenge in statistics and control) [122] due to function approximation and *bootstrapping* (i.e., instead of an exact value, using an estimated value in the update step). To achieve it, the scheme predicts two separate action-values and prefers the pessimistic value; hence, it avoids suboptimal policies. TD3 utilizes target networks, delays the update to the policy

function, and uses an average target value estimate by sampling N transitions from a replay buffer to reduce variance during learning. The scheme introduces exploration by adding Gaussian noise to the sampled actions and performs policy updates using the deterministic policy gradient [104].

Although the abovementioned algorithms provide general solutions to control problems, they may remain inadequate for more complex or specific tasks. Many other algorithms have been proposed to address these shortcomings. For example, Ref. [123] extended the discrete actor–critic method proposed by Ref. [44] to continuous time and space problems via the Hamiltonian–Jacobi–Bellman (HJB) equation [39,124]. This proposed algorithm was then tested in an action-constrained pendulum and a cart-pole swing up problem. Ref. [125] employed an actor–critic algorithm on a constrained MDP together with a detailed convergence analysis. Ref. [46] showcased four incremental actor–critic algorithms based on regular and natural gradient estimates. Ref. [126] introduced a natural actor–critic (NAC) and demonstrated its performance on the cart-pole problem as well as on a baseball swing task. Ref. [127] presented a continuous time actor–critic via converse HJB and tested the convergence in two nonlinear simulation environments. Ref. [128] proposed an online actor–critic algorithm for an infinite horizon, continuous time problems with a rigorous convergence analysis, and linear and nonlinear simulation examples. Ref. [129] proposed an incremental, online, and off-policy actor–critic algorithm. The proposal analyzed the convergence qualitatively and supported it with empirical results. Moreover, the temporal difference (TD) methods were compared with gradient-TD methods that minimize the projected Bellman error [36]. Ref. [130] proposed an actor–critic identifier that could provably approximate the HJB equation without a knowledge of the system dynamics. After the learning was complete, the scheme showed process stability. However, knowledge of the input gain matrix was required. Ref. [131] used a nominal controller as a supervisor to guide the actor and to yield a safer control in a simulated cruise-control system. Ref. [132] proposed learning the solution of an HJB equation for a partially unknown input-constrained system without the persistent excitation conditions while preserving the stability. By considering Lyapunov theory, Ref. [133] designed a fault-tolerant actor–critic algorithm and tested its stability on the Van der Pol system. Ref. [134] formulated an input-constrained nonlinear tracking problem by using the HJB equation and a quadratic cost function to define the value function. The scheme obtained an approximate value function with an actor–critic algorithm. Ref. [135] combined classification and time-series prediction techniques to solve an optimal control problem and showcased the proposed algorithm on a simulated continuous stirred-tank reactor (CSTR) and a simulated nonlinear oscillator. The mean actor–critic algorithm [136] was proposed to estimate the policy gradient by using a smooth Q function, which was averaged over the actions to reduce variance; the results were demonstrated on Atari games. Ref. [137] utilized an event-triggered actor–critic scheme to control a heating, ventilation, and air conditioning (HVAC) system. In addition to these, there are more recent studies on different actor–critic algorithms and their applications, as reported in Refs. [2,62,67,138–144].

Several methods have been proposed to improve value estimation in RL [146–148], which can be used in actor–critic algorithms. Moreover, different techniques [112,149] have been reported to improve the sample efficiency (i.e., to reduce the amount of data needed to learn the optimal policy). Unlike techniques that made use of experience replay [70] or supervised data [150], "parallel learning" makes use of multiple randomly initialized workers (*local networks*) that interact with different instances of the environment independently to reduce the variance in the policy during learning. These workers have the same infrastructure as a *global network* and, after collecting k-samples, are used to update parameters of the global network. This reduces the amount of memory used and improves exploration, because the workers have independent trajectories. Task distribution can be performed via multiple machines [151] or multiple central processing unit (CPU) threads of a single computer [48].

The optimal policy and the optimal critic are different in each process, and they are often unknown *a priori*. Monte Carlo-type methods calculate empirical return (given in Eq. (1)) at the end of the process (or an episode), which may be lengthy and noisy. Similar to Pavlovian conditioning [152] in psychology, TD learning predicts the value of the current state. Unlike Monte Carlo methods, it makes the predictions for a small horizon, as low as one step. This converts the infinite horizon problem into a finite horizon prediction problem. Instead of calculating the expectation of returns (as in Eq. (2)), the critic network can be updated using k-step ahead estimation of the TD error, δ, as shown in Eq. (14). This is called *policy evaluation*.

$$\delta\left(x_t\middle|\omega_L\right)=\sum_{i=0}^{k-1}\left[\gamma^i R_{t+i}+\gamma^k V\left(x_{t+k}\middle|\omega_L\right)\right]-V\left(x_t\middle|\omega_L\right) \quad (14)$$

where δ is the TD error for state x at a discrete sampling instant, t, given the critic parameters of the local network ω_L, and k represents the horizon length. If k approaches infinity, the summation terms converge to the empirical return given in Eq. (1). A baseline, $V\left(x_t\middle|\omega_L\right)$, is used to reduce the variance compared with the policy gradient algorithm [36].

At the end of k steps, the parameters of the global network (i.e., θ_G and ω_G) are updated using Eqs. (9) and (10).

3. Formulating the interface tracking as a sequential decision-making process

3.1. Interface tracking

A model is a mathematical means of describing the process dynamics that can occur either in a physical/chemical/biological system [153] or in a video [154]. The models derived for images often suffer from inaccuracies when there is an unexpected event (e.g., occlusion). To overcome this, either the information from the last valid observation is used in the next observation [4] or the images are reconstructed [154]. Although these solutions may substitute actual measurements for a short period of time, prolonged exposure can deteriorate closed-loop stability. As a consequence, if the FMI's level is too low, the bitumen from the froth layer drains into the tailings. This lowers the product quality and creates environmental footprints. In contrast, if its level is closer to the extraction point, the solid particles in the froth being extracted complicate downstream operations [3]. Since deviations in the FMI level affect the downstream processes, it is important to regulate the FMI at an optimum point.

RL can address inaccuracies during occlusion and excessive noise. This can be done by combining DP cell measurement or measurement from any other reliable instrument with the current FMI prediction by the agent to provide an accurate cost in the reward function, without external labels such as bounding boxes, during the training phase. Removing the dependence upon such labels minimizes human error. To achieve this, an agent can move a cropping box on the vertical axis over the PSV's sight glass and compare its center with the DP cell measurement. Based on this deviation, the agent can move the box to an optimal position, where the center of the box matches to that of the FMI. This deviation-minimizing feedback mechanism is inspired from control theory, and it can enhance an image-based estimation using the measurement obtained from the real process.

Consider a grayscale image, I, sampled from a video stream as $I \in \mathbb{R}^{H \times W}$ with an arbitrary width, W, and height, H, which captures the entire PSV. Consider a rectangular cropping box, $B \in \mathbb{R}^{N \times M}$, that has an arbitrary width, M, and height, N, where $\{N : N = 2\hat{z} - 1,\ \hat{z} > 1 \in \mathbb{N}\}$ and $\hat{z}$ is the center of the rectangle. An example image and a cropping box are shown in Fig. 6(a). This rectangle crops I at $\hat{z}$ into a size of $N \times M$. For the sake of completeness, $H > N$ and $W = M$. Consider an interface measurement obtained from a DP cell at time t as z. Note that the DP cell is used only in offline training of the RL agent and can be replaced by other interface measurement sensors, which is considered to be accurate in the offline laboratory environments. The components of the MDP for this problem can then be defined as follows:

States: The pixels inside the rectangle, $x \in B \subset X \subseteq I$. These pixels may be thought of as $N \times M$ independent sensors.

Actions: Move the center of the cropping box up or down by 1 pixel, or freeze; $u \in U = \{-1,\ 0,\ 1\}$.

Reward: The difference between the DP cell measurement and the position of the center of the box (with reference to the bottom of the PSV), at each timestep, t, given in Eq. (15).

$$R_t = -\left|z_t - \hat{z}_t\right| \qquad (15)$$

The relation between u_t and $\hat{z}_t$ is given as Eq. (16).

$$\hat{z}_t = \hat{z}_0 + \sum_{i=0}^{t-1} u_i \qquad (16)$$

where $\hat{z}_0$ is an arbitrary initial point, and the summation term represents the actions taken up to the tth instant ($u_i = +1$ for up, $u_i = -1$ for down).

Discount factor: $\gamma = 0.99$.

The goal of this agent is to generate a sequence of actions to overlay the cropping box, B, on the vertical axis of the PSV with the interface at its center. To achieve this, the agent needs to perform long-term planning and preserve the association between its actions and the information obtained from DP cell measurement. A flowchart of the proposed scheme is shown in Fig. 4. In addition, Fig. 5 and Table 2 show the networks in detail. More details about the ConvLSTM layer can be found in Ref. [27].

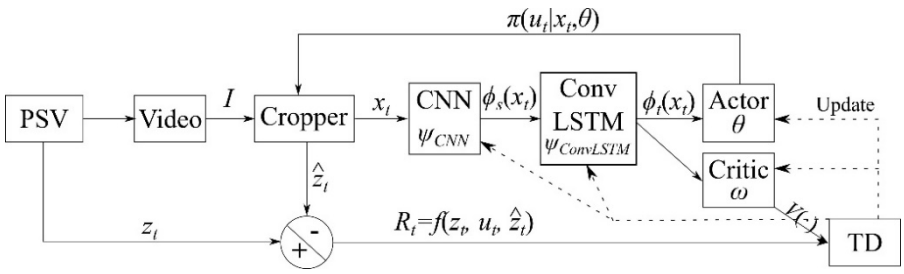

Fig. 4. Flow diagram for the proposed learning process. The update mechanism is shown in Eqs. (9) and (10) with the k-step policy evaluation, as shown in Eq. (14).

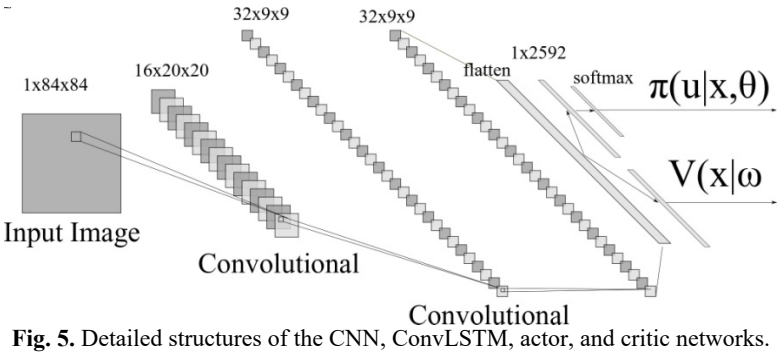

Fig. 5. Detailed structures of the CNN, ConvLSTM, actor, and critic networks.

Table 2
Structure of the global network (the same structure as the workers).

No.	Layer type	Output dimension	Filter size	Number of parameters
1	Convolutional	20 × 20 × 16	8 × 8	1 040
2	Convolutional	9 × 9 × 32	4 × 4	8 224
3	ConvLSTM	9 × 9 × 32	3 × 3	73 856
4	Fully connected (actor)	3		7 776
5	Fully connected (critic)	1		2 592
To tal				93 488

Unlike the previous works [4,5] that make predictions in the state space, this approach optimizes the value and the policy-spaces by using Eqs. (9), (10), and (14), respectively. Moreover, the CNN and ConvLSTM layers are updated by using Eq. (17).

$$\Psi \leftarrow \Psi + 0.5\times\alpha_c\nabla_{\Psi_L}\delta\left(\cdot\middle|\Psi_L\right)^2 + 0.5\times\alpha_a\nabla_{\Psi_L}\delta\left(\cdot\middle|\Psi_L\right)\ln\pi\left(\cdot\middle|\Psi_L\right) + \beta\pi\left(\cdot\middle|\Psi_L\right)\ln\pi\left(\cdot\middle|\Psi_L\right) \qquad (17)$$

where $\Psi = \left[\psi_{\text{CNN}},\ \psi_{\text{ConvLSTM}}\right]$ represents the parameters of the CNN and the ConvLSTM layers. This scheme trains the entire network end-to-end by using only the TD error. Multiple workers [48] that are initialized at different points (Fig. 6(b)) can be used to improve the exploration and hence the generalization.

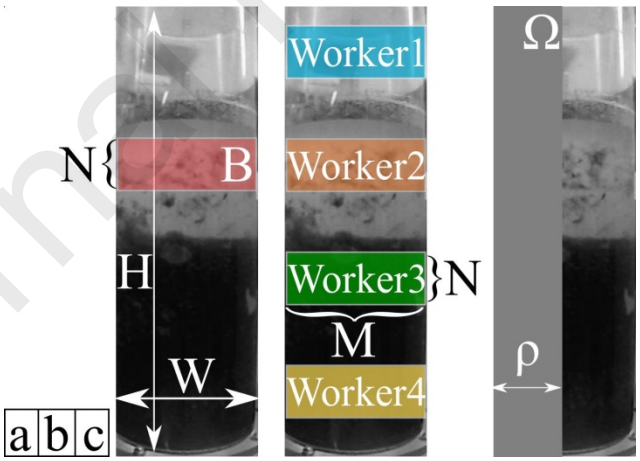

Fig. 6. A frame (I) obtained using a camera. (a) Sizes of the image ($H \times W$) and the cropping box ($N \times W$); (b) sizes of the cropping boxes ($N \times M$) and the initial cropping box positions; (c) an example occlusion with its ratio, ρ.

After a suboptimal policy is found, the agent is guaranteed to find the interface in a limited timestep k, independent of the initial point, as shown in Lemma 3.1.

Lemma 3.1: At any time t, for a constant z_t, with $P=1,\ \exists k : z_t - \left(\hat{z}_0 + \sum_{i=0}^{k} u_i\right) = 0,\ \left(\forall u \sim \pi\left(\cdot\middle|\theta^*\right)\right) \wedge \left(\forall x,\ u \in X \times U\right)$, as $k \to N$, for $\left(k \le N < \left|X\right| \ll \infty\right) \wedge \left(\forall z_0,\ z_t \in Z \equiv \left|X\right|\right)$.

Proof. Assume $\hat{z}_0,\ z_t \sim Z$, $\left\|z_t - \hat{z}_0\right\|_\infty \le \left|X\right| \ll \infty$, and the suboptimal parameters θ^* and ω^* are obtained using an iterative stochastic gradient descent over a continuous policy function $\pi\left(\cdot\middle|\theta^*\right)$. $V\left(\cdot\middle|\omega^*\right)$ is a Lipschitz continuous critic

network, parameterized by ω, and estimates the value of policy $\pi(\cdot)$ for a given state.

$\because u_i \sim \pi\left(\cdot\middle|\theta^*\right),\ |z_t - \hat{z}_0| \geq |z_t - \hat{z}_0 - u_0| \geq |z_t - \hat{z}_0 - u_0 - u_1| \Rightarrow V_{\pi^*}\left[x' = x\left(\hat{z}_0 + u_0\right)\right] \geq V_{\pi^*}\left[x\left(\hat{z}_0\right)\right].$

Similarly, $V_{\pi^*}\left[x'' = x\left(\hat{z}_0 + u_0 + u_1\right)\right] \geq V_{\pi^*}\left[x\left(\hat{z}_0 + u_0\right)\right]$

$\because \left\|z_t - \hat{z}_0\right\|_\infty \ll \infty,\ \lim_{k \to N \ll \infty} z_t - \left(\hat{z}_0 + \sum_{i=0}^{k} u_i\right) = 0$

This can be extended to a variable $z_t \in Z$.

3.2. Robustness to occlusion via training

CNNs interpret the spatial information by considering the connectivity of the pixels, which improves robustness up to a certain point. However, it does not guarantee robustness to occlusion, and the agent may fail even if a good policy is obtained under normal conditions. To overcome this issue, the agent may be trained using synthetically occluded images during the training phase. Another way is to recalibrate a policy (that was trained using occlusion-free images) with occluded images.

An occluding object, Ω, with an arbitrary pixel intensity, $\kappa \in [0,\ 255]$, can be defined as $\left\{\Omega : \Omega \in \mathbb{R}^{H \times (N \times \rho)}\right\}$, where $E[\Omega] = \kappa$. $\rho \in [0,\ 100\%]$ represents the ratio of occlusion, as shown in Fig. 6(c). If ρ = 1, the agent observes only the occlusion in that video frame (i.e., $x_t = \Omega$ if ρ = 100%). After defining its size, the ratio of occlusion can be sampled from an arbitrary probability distribution (i.e., continuous or discrete, e.g., Gaussian, uniform, Poisson). During training, the duration of the instance at which the occlusion appears may be adjusted arbitrarily. These can be stochastic or deterministic. That is, the occlusion may appear at a random (or specific) time for a random (or particular) duration. If multiple workers (as described in Section 2.2) are used, different occlusion ratios at different time instances with different durations may be introduced to each worker. This improves the diversity of the training data, which makes the process time efficient, because the agent does not need to wait for a long time to observe different types of occlusion.

4. Results and discussion

4.1. Experimental setup

A lab-scale setup that mimics an industrial PSV is used for the proposed scheme. This setup allows for the movement of the interface to a desired level using pumps, as shown in Fig. 7. Two DP cells are used to measure the interface level based on the liquid density, as described in Ref. [5].

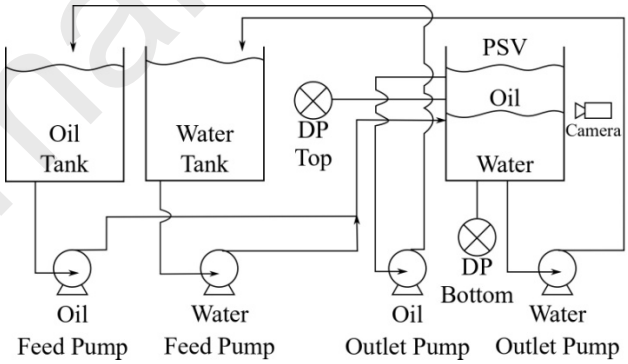

Fig. 7. The experimental setup.

4.1.1. Data acquisition and simulation creation

Images are obtained using D-Link DCS-8525LH camera at 15 frames per second (FPS). From the 15 FPS footages, a representative image for each second is obtained. Hence, 80 images from 80 consecutive seconds are obtained with necessary down-sampling. These images are processed to showcase the PSV portion, void of unwanted background. They are then converted into grayscale images. The DP cell measurements (for the same contiguous time period as the images), which are available in terms of water head (water-in), are converted to pixel positions, as given in Ref. [4]. After each action is taken, the video frame changes. Every action the agent takes generates a scalar reward (Eq. (15)), which is later utilized to calculate the TD error (Eq. (14)) that is used in training the agent's parameters (Eqs. (9) and (10)).

4.2. Implementation details

4.2.1. Software and network details

Both the training and the testing phases were conducted using an Intel Core i7-7500U CPU at 2.90 GHz (two cores, four threads), 8 GB RAM at 2133 MHz, and 64-bit Windows using Tensorflow 1.15.0. Unlike deeper networks (e.g., those in Ref. [32] that consisted of tens of millions of parameters), this agent consisted of fewer parameters, as summarized in Table 2. This prevents over-parameterization and reduces the computational time significantly, with the disadvantage of an inability to extract higher level features [155].

After each action is taken, the cropping box is resized to 84 × 84 pixels. An Adam optimizer with a learning rate of 0.0001 is used to optimize the parameters of the agent (including the CNN, ConvLSTM, actor, and critic) in a sample-based manner. This momentum-based stochastic optimization method has been reported to be computationally efficient [156].

4.2.2. Training without occlusion

An A3C algorithm was used during the experiments to reduce the training time, improve exploration, and achieve convergence to a suboptimal policy during learning [48]. All of the initial network parameters were sampled randomly from a Gaussian distribution with zero mean and unit variance. Offline training was performed after creating a continuous trajectory of the interface level by manually ordering 80 unique images, as shown in Fig. 8.

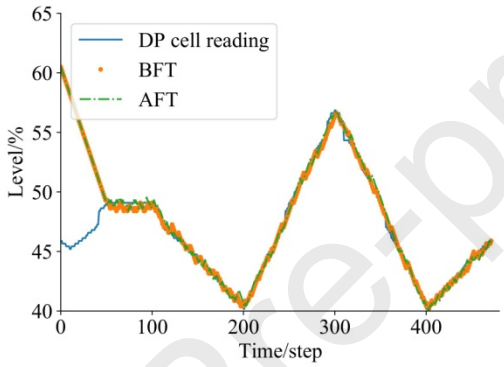

Fig. 8. Training results at the end of training (2650 episodes) and fine-tuning (3380 episodes). BFT: before fine-tuning; AFT: after fine-tuning.

This trajectory was then repeatedly shown to the agent for 470 steps for 2650 episodes (i.e., an episode consisted of 470 steps). At any time, the agent observed only the pixels within the cropping box. The cropping box of each agent was initialized at four different positions, as shown in Fig. 6(b). The agent's goal was to minimize the deviation of the center of the cropping box with respect to the DP cell measurements, given a maximum velocity of 1 pixel per step. The agent was not exposed to occlusion during training and was capable of processing 20 FPS (i.e., computational execution time) for four workers.

4.2.3. Fine-tuning with occlusion

The global network parameters were initialized using the parameters obtained at the end of the training without occlusion. The local networks initially shared the same parameters as the global network. All of the training hyperparameters (e.g., learning rate, interface trajectory) were kept unchanged. The images used in the previous training phase were overlayed with occlusion, whose ratio, ρ, was sampled from a Poisson distribution, as shown in Eq. (18). The distribution, Pois(x, λ), is given in Eq. (19).

$$\rho \sim \frac{\rho_{\max} - \mathrm{Pois}(x,\ \lambda)}{10} \times 100\% \qquad (18)$$

$$\mathrm{Pois}(x,\ \lambda) = \frac{\mathrm{e}^{-\lambda}\lambda^{x}}{x!} \qquad (19)$$

Eq. (18) bounds ρ between 0 and $\rho_{\max}$ = 80% at the beginning of an episode. Shape factor is arbitrarily defined as λ = 1. In each episode, occlusion occurs at the 200th step to the following 200 steps with a probability of 1. The intent behind fine-tuning (FT) is to make sure the agent is robust to the occlusion. The agent, with four workers, was trained for an arbitrary amount of 730 episodes until the episodic cumulative reward improved.

4.2.4. Interface tracking test

For a 1000-step episode, the agent was tested using a discontinuous trajectory that contained previously unseen images that were either noiseless or were laden with a Gaussian noise, $\nu \in \mathbb{R}^{H\times W} \sim N(0,\ 1)$, in three ways, as shown in Table 3.

These images were also occluded using a synthetic occlusion, whose constant intensity was arbitrarily selected as the mean of the image (i.e., $\kappa = 128$), while the occlusion ratio, ρ, varied linearly from 20% to 80%.

Table 3
Definition of noisy images based on their identities.

Identity of the noisy image	Noisy image	Condition
1	$I_t = I_t + \nu \odot \zeta$	$t < 300$
2	$I_t = I_t \odot (1 + \nu \odot \zeta)$	$300 \le t < 700$
3	$I_t = I_t \odot (1 + \nu \odot 2 \times \zeta)$	$t \ge 700$

$\odot$ represents the Hadamard product. ζ is the magnitude of noise.

4.2.5. Feature analysis

To illustrate the effectiveness of the network, a previously unseen PSV image was manually cropped starting from the top of the PSV to the bottom. These manually cropped images were then passed one by one through the CNN prior to training, the CNN was trained as in Section 4.2.2, and the CNN was fine-tuned as discussed in Section 4.2.3 to extract the features. These spatial features, ϕ_s, were then collected in a buffer with the size $9 \times 9 \times 32 \times 440$, from which the reduced dimension (2×440) features were obtained using UMAP [99]. These lower dimensional features will be represented in Section 4.6.

4.3. Training

The best policies were obtained at the end of training and FT, when there was no improvement in the cumulative reward for 500 consecutive episodes. Fig. 8 shows the trajectories using these policies. The position of the cropping box is initialized with its center at 60% of the PSV's maximum height. At the end of this phase, the agent tracked the interface with a negligible amount of offset. An example obtained from the 80th step is shown in Fig. 9(a). The green star represents where the agent thinks the interface is for the current frame.

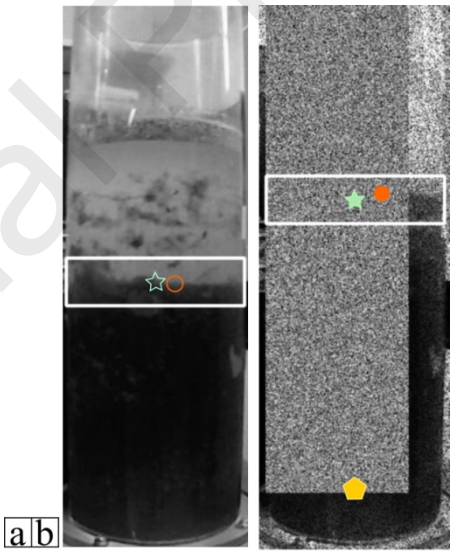

Fig. 9. (a) Training result at the 80th frame. (b) Test result after FT with 80% occlusion and excessive noise, at the 950th step. The white boxes represent the cropping box that the agent controls. The stars represent the center of the cropping box, and the circles are the exact interface level. The pentagon is the bottom of the occlusion, which looks like the FMI.

4.4. FT re-calibration for occlusion

FT improved the agent's overall performance, even for the occlusion-free images, by reducing the level-wise mean average error (MAE) by 0.51%, as summarized in Table 4. This result indicates that the agent adapted to the new environmental conditions without forgetting the previous conditions. This was due to the improvements in the value estimation and the policy, which started from near-optimal points. Note that the minimum value for the MAE is limited by the initial position of the cropping box, as shown in Fig. 8.

Table 4
Pixel- and level-wise MAE at the end of training and FT.

Stage	MAE pixel	MAE level
After training	4.9852	1.1382
After FT	4.9597	1.1324

Fig. 10 shows the cumulative rewards from one of the workers during training and after FT, as shown in solid and dash-dot lines, respectively.

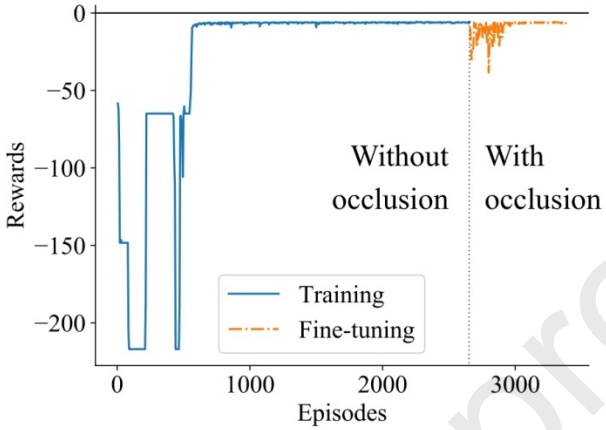

Fig. 10. Cumulative rewards. The graph shows that the agent can learn the occlusion and track the interface successfully.

Note that the initial decrease during FT was caused by the occlusion, because the agent was not able to track the interface level when occlusion occurred. This new feature was learned successfully by the closed-loop reward mechanism within 400 episodes. Note that the final cumulative reward obtained at the end of FT is almost the same as that obtained at the end of training. This is because the cumulative reward represents only the tracking performance during training and depends on the initial position of the cropping box, as shown in Fig. 8. This value can be zero only if the center of the box and the DP cell measurement overlap completely at the beginning of the episode and the agent tracks the interface without any offsets during the episode. The necessity of the FT is more pronounced when the agent is exposed to unseen environmental conditions such as excessive noise and occlusion, as discussed in Section 4.5.

4.5. Test

4.5.1. Before fine-tuning

The initial before fine-tuning (BFT) test was conducted at the end of the initial training (i.e., the 2650th episode, as shown in Fig. 10). Note that in the testing (online application) phase, DP cell information is not being used, and the RL agent works on its own. In fact, even if the DP cell is available, it will not be accurate in the field application environment. Fig. 11 shows that the agent was robust to up to 50% occlusion and additional noise prior to fine-tuning. This is a significant improvement over the existing schemes, all of which do not address occlusion. The reason for this improvement is that the neural networks extract more abstract features than edge and histogram information, in both the spatial and temporal domains [157]. This is due to the convolutional operations that smooth out disturbances and improve the agent's overall performance. On the other hand, any further increase to the occlusion ratio resulted in failure to track the interface. Since occlusion is of lighter intensity, the policy naturally moved toward the bottom of the PSV (where pixels of higher intensity were abundant) to find the interface.

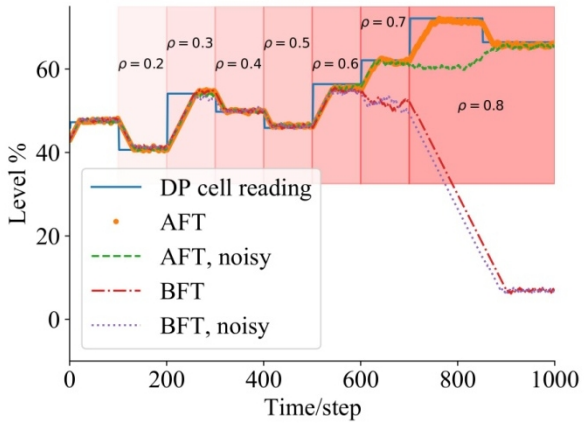

Fig. 11. Test results: Tracking, where ρ is the occlusion ratio (e.g., $\rho = 0.8$ means that the image is occluded by 80%.)

4.5.2. After fine-tuning

After fine-tuning (AFT), it was found that recalibrating the agent for occlusion improved its performance significantly, as seen from its ability to track the interface more accurately (Fig. 11). Additional noise caused its performance to degrade when the interface offset between the consecutive frames was around 5%. However, the agent was successful when this interface offset was reduced to 2.5%, as shown in Fig. 11. This is because the excessive noise corrupts the image significantly and the agent fails to locate the interface. An example frame obtained at the 950th frame is shown in Fig. 9(b). It should be noted that the noise is accompanied by 80% occlusion; this makes the tracking problem more challenging, since the amount of useful information extracted by the agent from the image is significantly reduced—that is, only 20% of the pixels can be used to locate the interface. This performance is due to the CNN and ConvLSTM combination. Fig. 12 shows the agent's beliefs (predicted by the critic) about the states (obtained from an unseen frame) using parameters obtained from a random network (solid), after training (dash-dot), and AFT (dot). According to Eq. (2), this figure defines the value of a state, assuming that the best trajectory toward the interface level would be generated by the policy.

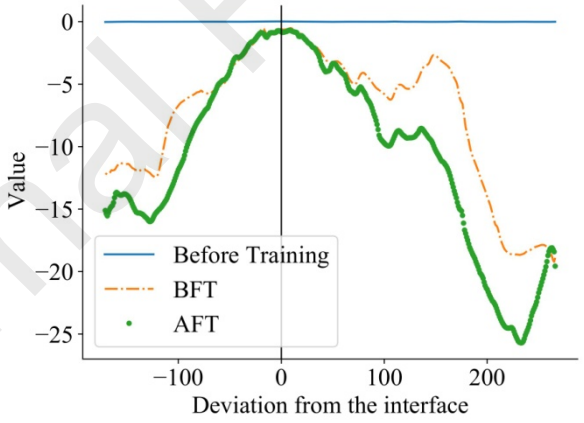

Fig. 12. Test results of value function versus deviation from the interface.

Fig. 12 also shows that, prior to any training, the value predicted for any state is similar. However, during training, the agent regrets being in disadvantegous states, and the DP cell readings reinforce that moving the cropping box closer to the interface (i.e., a vertical solid line) yields a better value than being further away from it. At the end of FT, with more data, the agent further improves its parameters—and therefore its actions—to move the cropper box so that it becomes more accurate. This result shows that the agent tries to improve its actions based on a constantly changing belief (value). Note that the increase in AFT after a deviation value of 200 corresponds to the yellow pentagon in Fig. 9, which looks like the interface and causes an increase in the value function. However, the value obtained from that part is lower than that of the interface, meaning that the agent is more confident when it is close to the star, rather than to the pentagon.

4.6. Understanding the network: Feature analysis

The training and test results focused on the progress of the learning and control abilities of the agent. These alone may

not be sufficient to explain whether the agent's decisions are meaningful given an observation in the form of an image.

Fig. 13 shows the reduced dimensionality as a two-dimensional graph by representing the values of the corresponding cropped images (obtained in Section 4.2.5) using the gradual intensities of a color. The curve (from left to right) corresponds to the cropped images from top to bottom of the PSV tank side glass, as explained in Section 4.2.5.

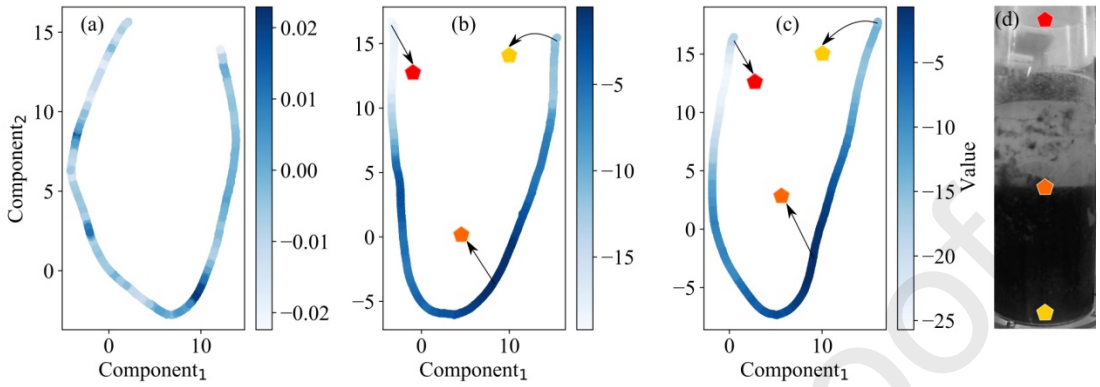

Fig. 13. Dimensionality reduction applied to the features of the states ($x \in X$) obtained from an unseen image. The features are obtained using the parameters obtained from (a) random, (b) trained, and (c) fine-tuned networks. The data points are then colored by their corresponding values. (d) Three regions that correspond to the top and the bottom of the tank and the FMI are highlighted on the unseen image. As the agent trains, the extracted features from similar regions are clustered closer in the Riemannian space.

The colored pentagons in Figs. 13(a)–(c) correspond to three points in Fig. 13(d). According to the results, the features obtained from the network prior to training are similar to each other without any particular arrangement. However, as training proceeds, features with similar values get closer. Upon combining Fig. 13 with Fig. 12, it could be inferred that the CNN was able to extract the features in a meaningful way, despite using unlabeled data in a model-free context, due to the RL methodology. This was possible because the texture and pixel intensity pattern of each cropped image was successfully converted into the value and the policy functions by employing a CNN–ConvLSTM combination. Also, the reward signal obtained from the DP cell (which was used as a feedback mechanism) trained the agent's behavior.

5. Conclusion

This work provided a comprehensive review on actor–critic algorithms and proposed a novel RL scheme that targets the instrumentation level of the control hierarchy in order to improve the performance of the entire structure. To achieve this result, interface tracking was formulated as a sequential decision-making process that requires long-term planning. The agent was composed of a CNN and ConvLSTM combination that does not require any shape or motion models and is hence more robust to uncertainties in the process conditions. Inspired from the feedback mechanism used in control theory, the agent utilized readings from DP cells to improve its actions. This technique removes the dependencies on explicit labels that are required for SL schemes. The agent's performance during validation using untrained images under occlusion and noise showed that the interface can be tracked under up to 80% occlusion and excessive noise. An analysis of the high-dimensional features validated the agent's generalization of its beliefs around its observations.

6. Future work

This work successfully demonstrated the tracking of a liquid interface by utilizing one of the most advanced RL techniques. The occlusion was handled by employing an agent composed of deep CNN structures, and the tolerance was improved by FT the policy, which showcased the adaptive nature of the proposed method. In addition to these, an agent that can reconstruct the occluded images may be an alternative method for future work.

Acknowledgements

The authors thank Dr. Fadi Ibrahim for his help in the laboratory to initiate this research and Dr. Artin Afacan for the lab-scale PSV setup. The authors also acknowledge the Natural Sciences Engineering Research Council of Canada (NSERC), and its Industrial Research Chair (IRC) Program for financial support.

Compliance with ethics guidelines

Oguzhan Dogru, Kirubakaran Velswamy, and Biao Huang declare that they have no conflict of interest or financial

conflicts to disclose.

Nomenclature

Abbreviations

A2C	advantage actor–critic
A3C	asynchronous advantage actor–critic
ACER	actor–critic with experience replay
ACKTR	actor–critic using Kronecker-factored trust region
AFT	after fine-tuning
BFT	before fine-tuning
CNN	convolutional neural network
ConvLSTM	convolutional long short-term memory
CSTR	continuous stirred-tank reactor
DDPG	deep deterministic policy gradient ()
DP	differential pressure
FIM	Fisher information matrix
FMI	froth–middlings interface
FPS	frames per second
FT	fine-tuning
GAN	generative adversarial network
HJB	Hamiltonian–Jacobi–Bellman
HVAC	heating, ventilation, air conditioning
LSTM	long short-term memory
MAE	mean average error
MDP	Markov decision process
NAC	natural actor–critic
PPO	proximal policy optimization
PSV	primary separation vessel
RL	reinforcement learning
RNN	recurrent neural network
SAC	soft actor–critic
SL	supervised learning
TD	temporal difference
TD3	twin delayed deep deterministic policy gradient
TRPO	trust region policy optimization
t-SNE	*t*-distributed stochastic neighbor embedding
UL	unsupervised learning
UMAP	uniform manifold approximation and projection

Symbols

$\mathbb{E}[\cdot]$	expectation
$\phi_{\mathrm{s}}(\cdot)$	spatial features
$\phi_{\mathrm{t}}(\cdot)$	temporal features
δ	temporal difference error
σ_0	distribution of initial states
ν	gaussian noise with zero mean unit variance
$(\cdot)^*$	optimum value for the variable, e.g., q^*
$\ln(\cdot)$	natural logarithm
$R,\ G$	empirical reward, return
$q,\ r,\ v$	expected action-value, reward, state-value
$x,x' \in X$	States $\in$ State space
$u \in U$	Actions $\in$ Action space
$\pi(\cdot)$	policy of the agent, also known as the actor
$\delta(x_t\|\omega_{\mathrm{L}})$	temporal difference error
$V(\cdot)$	estimate of state-value, also known as the critic
$Q(\cdot)$	estimate of action-value, also known as the critic

Ω occlusion

Parameters

α_a, α_c learning rates for the actor and critic: 0.0001
γ discount factor: 0.99
κ intensity of occlusion: 128/256
λ shape parameter of a Poisson distribution: 1
ρ occlusion ratio: %
ζ magnitude of noise: 0.2

References

[1] Masliyah J, Zhou ZJ, Xu Z, Czarnecki J, Hamza H. Understanding water-based bitumen extraction from Athabasca oil sands. Can J Chem Eng 2004;82(4):628–54.
[2] Shafi H, Velswamy K, Ibrahim F, Huang B. A hierarchical constrained reinforcement learning for optimization of bitumen recovery rate in a primary separation vessel. Comput Chem Eng 2020;140:106939.
[3] Jampana P, Shah SL, Kadali R. Computer vision based interface level control in separation cells. Control Eng Pract 2010;18(4):349–57.
[4] Vicente A, Raveendran R, Huang B, Sedghi S, Narang A, Jiang H, et al. Computer vision system for froth-middlings interface level detection in the primary separation vessels. Comput Chem Eng 2019;123:357–70.
[5] Liu Z, Kodamana H, Afacan A, Huang B. Dynamic prediction of interface level using spatial temporal Markov random field. Comput Chem Eng 2019;128:301–11.
[6] Ruder S. An overview of gradient descent optimization algorithms. 2016. arXiv:1609.04747.
[7] Xie R, Jan NM, Hao K, Chen L, Huang B. Supervised variational autoencoders for soft sensor modeling with missing data. IEEE Trans Industr Inform 2019;16(4):2820–8.
[8] Raveendran R, Kodamana H, Huang B. Process monitoring using a generalized probabilistic linear latent variable model. Automatica 2018;96:73–83.
[9] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation applied to handwritten zip code recognition. Neural Comput 1989;1(4):541–51.
[10] Babu GS, Zhao P, Li XL. Deep convolutional neural network based regression approach for estimation of remaining useful life. In: Navathe S, Wu W, Shekhar S, Du X, Wang X, Xiong H, editors. Database systems for advanced applications. DASFAA 2016. Lecture notes in computer science, vol 9642. Cham: Springer; 2016. p. 214–28.
[11] He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision; 2017 Oct 22–29; Venice, Italy; 2017. p. 2961–9.
[12] Kingma DP, Welling M. Auto-encoding variational Bayes. 2013. arXiv:1312.6114.
[13] Hubel DH, Wiesel TN. Receptive fields of single neurones in the cat's striate cortex. J Physiol 1959;148(3):574–91.
[14] Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol 1962;160(1):106–54.
[15] Hubel DH, Wiesel TN. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. J Neurophysiol 1965;28(2):229–89.
[16] Fukushima K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 1980;36(4):193–202.
[17] Guo Y, Liu Y, Oerlemans A, Lao S, Wu S, Lew MS. Deep learning for visual understanding: a review. Neurocomputing 2016;187:27–48.
[18] Lundervold AS, Lundervold A. An overview of deep learning in medical imaging focusing on MRI. Z Med Phys 2019;29(2):102–27.
[19] Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes MP, et al. A survey on deep learning: algorithms, techniques, and applications. ACM Comput Surv 2019;51(5):1–36.
[20] Wu X, Chen J, Xie L, Chan LLT, Chen CI. Development of convolutional neural network based Gaussian process regression to construct a novel probabilistic virtual metrology in multi-stage semiconductor processes. Control Eng Pract 2020;96:104262.
[21] Eklund A, Dufort P, Forsberg D, LaConte SM. Medical image processing on the GPU—past, present and future. Med Image Anal 2013;17(8):1073–94.
[22] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet large scale visual recognition challenge. Int J Comput Vision 2015;115(3):211–52.
[23] Xie RM, Hao KG, Huang B, Chen L, Cai X. Data-driven modeling based on two-stream λ gated recurrent unit network with soft sensor application. IEEE Trans Ind Electron 2019;67(8):7034–43.
[24] Elman JL. Finding structure in time. Cognitive Sci 1990;14(2):179–211.
[25] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.
[26] Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder–decoder for sta-tistical machine translation. 2014. arXiv:1406.1078.
[27] Shi X, Chen Z, Wang H, Yeung DY, Wong WK, Woo W. Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Cortes C, Lee DD, Sugiyama M, Garnett R, editors. Proceedings of the 28th International Conference on Neural Information Processing Systems, volume 1; 2015 Dec 7–12; Montreal, QC, Canada. Cambridge: MIT Press; 2015. p. 802–810.
[28] Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, et al. A survey on deep learning in medical image analysis. Med Image Anal 2017;42:60–88.
[29] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Proceedings of the 25th International Conference on Neural Information Processing Systems, volume 1; 2012 Dec 3–6; Lake Tahoe, NV, USA. Red Hook: Curran Associates Inc.; 2012. p. 1097–105.
[30] Matthew DZ, Rob F. Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T, editors. Computer vision—ECCV 2014. Cham: Springer; 2014. p. 818–33.
[31] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: Proceedings of 2015 IEEE Conference on

Computer Vision and Pattern Recognition; 2015 Jun 7–12; Boston, MA, USA. New York: IEEE; 2015. p. 1–9.
[32] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition; 2016 Jun 27–30; Las Vegas, NV, USA. New York: IEEE; 2016. p. 770–8.
[33] Yuan X, Huang B, Wang Y, Yang C, Gui W. Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted SAE. IEEE Trans Industr Inform 2018;14(7):3235–43.
[34] Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng 2010;22(10):1345–59.
[35] Tan C, Sun F, Kong T, Zhang W, Yang C, Liu C. A survey on deep transfer learning. In: Kůrková V, Manolopoulos Y, Hammer B, Iliadis L, Maglogiannis I, editors. Artificial neural networks and machine learning—ICANN 2018. Cham: Springer; 2018. p. 270–9.
[36] Sutton RS, Barto AG. Reinforcement learning: an introduction. Cambridge: MIT press; 2000.
[37] Thorndike EL. Animal intelligence. Nature 1898;58(390):520.
[38] Farley B, Clark W. Simulation of self-organizing systems by digital computer. Trans IRE Prof Group Inform Theory 1954;4(4):76–84.
[39] Bellman R. The theory of dynamic programming. Bull Am Math Soc 1954;60(6):503–16.
[40] Sutton RS, Barto AG, Williams RJ. Reinforcement learning is direct adaptive optimal control. IEEE Contr Syst Mag 1992;12(2):19–22.
[41] Donald EK. Optimal control theory: an introduction. New York: Dover Publication; 2004.
[42] Bertsekas DP. Reinforcement learning and optimal control. Belmont: Athena Scientific; 2019.
[43] Szepesvári C. Algorithms for reinforcement learning. Edmonton: Morgan and Claypool Publishers; 2010.
[44] Barto AG, Sutton RS, Anderson CW. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans Syst Man Cybern 1983;SMC-13(5):834–46.
[45] Konda VR, Tsitsiklis JN. Actor–critic algorithms. In: Solla SA, Leen TK, Müller K, editors. Proceedings of the 12th International Conference on Neural Information Processing Systems; 1999 Nov 29–Dec 4; Denver, CO, USA. Cambridge: MIT Press; 2000. p. 1008–14.
[46] Bhatnagar S, Ghavamzadeh M, Lee M, Sutton RS. Incremental natural actor–critic algorithms. In: Platt JC, Koller D, Singer Y, Roweis ST, editors. Proceedings of the 20th International Conference on Neural Information Processing Systems; 2007 Dec 3–6; Vancouver, BC, Canada. Red Hook: Curran Associates Inc.; 2007. p. 105–12.
[47] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. 2015. arXiv:1509.02971.
[48] Mnih V, Badia AP, Mirza M, Graves A, Harley T, Lillicrap TP, et al. Asynchronous methods for deep reinforcement learning. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd International Conference on International Conference on Machine Learning, volume 48; 2016 Jun 19–24; New York, NY, USA; 2016. p. 1928–37.
[49] Ljung L. System identification. New York: American Cancer Society; 1999.
[50] Huang B, Qi Y, Monjur Murshed AKM. Dynamic modeling and predictive control in solid oxide fuel cells: first principle and data-based approaches. Chichester: John Wiley & Sons; 2013.
[51] Kodamana H, Huang B, Ranjan R, Zhao Y, Tan R, Sammaknejad N. Approaches to robust process identification: a review and tutorial of probabilistic methods. J Process Contr 2018;66:68–83.
[52] Pendrith M. On reinforcement learning of control actions in noisy and non-Markovian domains. Sydney: The University of New South Wales; 1994.
[53] Plappert M, Houthooft R, Dhariwal P, Sidor S, Chen RY, Chen X, et al. Parameter space noise for exploration. 2017. arXiv:1706.01905.
[54] Tang H, Houthooft R, Foote D, Stooke A, Chen X, Duan Y, et al. #Exploration: a study of count-based exploration for deep reinforcement learning. In: von Luxburg U, Guyon I, Bengio S, Wallach H, Fergus R, editors. Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4; Long Beach, CA, USA. Red Hook: Curran Associates Inc.; 2017. p. 2750–9.
[55] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor–critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2018. arXiv:1801.01290.
[56] Buşoniu L, de Bruin T, Tolić D, Kober J, Palunko I. Reinforcement learning for control: performance, stability, and deep approximators. Annu Rev Contr 2018;46:8–28.
[57] Ciosek K, Vuong Q, Loftin R, Hofmann K. Better exploration with optimistic actor–critic. 2019. arXiv:1910.12807.
[58] Luck KS, Vecerik M, Stepputtis S, Amor HB, Scholz J. Improved exploration through latent trajectory optimization in deep deterministic policy gradient. 2019. arXiv:1911.06833.
[59] Couffignal L. Les Machines à calculer, leurs principes, leur evolution. Paris: Gauthier-Villars.; 1933. French.
[60] Turing AM. I.—Computing machinery and intelligence. Mind 1950;LIX(236):433–60.
[61] Arf C. Makine Düşünebilir Mi ve Nasıl Düşünebilir? In: Üniversite Çalışmalarını Muhite Yayma ve Halk Eğitimi Yayınları Konferanslar Serisi No: 1. Erzurum: Atatürk Üniversitesi; 1959. p. 91–103. Turkish.
[62] Wang Y, Velswamy K, Huang B. A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. Processes 2017;5(4):46.
[63] Spielberg SPK, Gopaluni RB, Loewen PD. Deep reinforcement learning approaches for process control. In: Proceedings of the 6th International Symposium on Advanced Control of Industrial Processes; 2017 May 28–31; Taipei, China; New York: IEEE; 2017. p. 201–6.
[64] Pandian BJ, Noel MM. Tracking control of a continuous stirred tank reactor using direct and tuned reinforcement learning based controllers. Chem Prod Process Mo 2018;13(3):20170040.
[65] Badgwell TA, Lee JH, Liu KH. Reinforcement learning overview of recent progress and implications for process control. Comput Chem Eng 2019;127:282–94.
[66] Ruan Y, Zhang Y, Mao T, Zhou X, Li D, Zhou H. Trajectory optimization and positioning control for batch process using learning control. Control Eng Pract 2019;85:1–10.
[67] Nian R, Liu J, Huang B. A review on reinforcement learning: introduction and applications in industrial process control. Comput Chem Eng 2020;139:106886.
[68] Zhu L, Cui Y, Takami G, Kanokogi H, Matsubara T. Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process. Control Eng Pract 2020;97:104331.
[69] Todorov E, Erez T, Tassa Y. MuJoCo: a physics engine for model-based control. In: Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2012 Oct 7–12; Vilamoura-Algarve, Portugal. New York: IEEE; 2012. p. 5026–33.
[70 Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing Atari with deep reinforcement learning. 2013. arXiv:1312.5602.
[71] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33.

[72] Jaderberg M, Czarnecki WM, Dunning I, Marris L, Lever G, Castañeda AG, et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning. Science 2019;364(6443):859–65.
[73] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. OpenAI Gym. 2016. arXiv:1606.01540.
[74] Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science 2018;362(6419):1140–4.
[75] Baker B, Kanitscheider I, Markov T, Wu Y, Powell G, McGrew B, et al. Emergent tool use from multi-agent autocurricula. 2019. arXiv:1909.07528.
[76] Berner C, Brockman G, Chan B, Cheung V, Dębiak P, Dennison C, et al. Dota 2 with large scale deep reinforcement learning. 2019. arXiv:1912.06680.
[77] Badia AP, Piot B, Kapturowski S, Sprechmann P, Vitvitskyi A, Guo D, et al. Agent57: outperforming the Atari human benchmark. 2020. arXiv:2003.13350.
[78] Bucak IO, Zohdy MA. Reinforcement learning control of nonlinear multi-link system. Eng Appl Artif Intell 2001;14(5):563–75.
[79] Kober J, Bagnell JA, Peters J. Reinforcement learning in robotics: a survey. Int J Robot Res 2013;32(11):1238–74.
[80] Amini A, Gilitschenski I, Phillips J, Moseyko J, Banerjee R, Karaman S, et al. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. IEEE Robot Autom Lett 2020;5(2):1143–50.
[81] Pi CH, Hu KC, Cheng S, Wu IC. Low-level autonomous control and tracking of quadrotor using reinforcement learning. Control Eng Pract 2020;95:104222.
[82] Mathe S, Pirinen A, Sminchisescu C. Reinforcement learning for visual object detection. In: He KM, Zhang XY, Ren SQ, Sun J, editors. Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition; 2016 Jun 27–30; Las Vegas, NV, USA. New York: IEEE; 2016. p. 2894–902.
[83] König J, Malberg S, Martens M, Niehaus S, Krohn-Grimberghe A, Ramaswamy A. Multi-stage reinforcement learning for object detection. In: Arai K, Kapoor S, editors. Advances in computer vision. Cham.: Springer; 2019. p. 178–91.
[84] Halici E, Alatan AA. Object localization without bounding box information using generative adversarial reinforcement learning. In: Liu D, Lee S, Li XL, Bhanu B, Li HQ, Jung C, et al. editors. Proceedings of the 25th IEEE International Conference on Image Processing; 2018 Oct 7–10; Athens, Greece. New York: IEEE; 2018. p. 3728–32.
[85] Zhang D, Maei H, Wang X, Wang YF. Deep reinforcement learning for visual object tracking in videos. 2017. arXiv:1701.08936.
[86] Luo W, Sun P, Zhong F, Liu W, Zhang T, Wang Y. End-to-end active object tracking via reinforcement learning. 2017. arXiv:1705.10561.
[87] Ren LL, Lu JW, Wang ZF, Tian Q, Zhou J. Collaborative deep reinforcement learning for multi-object tracking. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y, editors. Computer Vision—ECCV 2018. Cham: Springer; 2018. p. 586–602.
[88] Yun S, Choi J, Yoo Y, Yun K, Choi JY. Action-decision networks for visual tracking with deep reinforcement learning. In: Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition; 2017 Jul 21–26; Honolulu, HI, USA. New York: IEEE; 2017. p. 2711–20.
[89] Choi J, Kwon J, Lee KM. Real-time visual tracking by deep reinforced decision making. 2017. arXiv:1702.06291.
[90] Li P, Wang D, Wang L, Lu H. Deep visual tracking: review and experimental comparison. Pattern Recognit 2018;76:323–38.
[91] Chen BX, Tsotsos JK. Fast visual object tracking with rotated bounding boxes. 2019. arXiv:1907.03892.
[92] Wang Z, Xu J, Liu L, Zhu F, Shao L. RANet: ranking attention network for fast video object segmentation. In: Proceedings of 2019 IEEE/CVF International Conference on Computer Vision; 2019 Oct 27–Nov 2; Seoul, Republic of Korea. New York: IEEE; 2019. p. 3978–87.
[93] Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Li FF. Large-scale video classification with convolutional neural networks. In: Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition; 2014 Jun 23–28; Columbus, OH, USA. New York: IEEE; 2014. p. 1725–32.
[94] Li J, Chen X, Hovy E, Jurafsky D. Visualizing and understanding neural models in NLP. 2015. arXiv:1506.01066.
[95] Yosinski J, Clune J, Nguyen A, Fuchs T, Lipson H. Understanding neural networks through deep visualization. 2015. arXiv:1506.06579.
[96] Wattenberg M, Viégas F, Johnson I. How to use t-SNE effectively. Distill 2016;1(10):e2.
[97] Zrihem NB, Zahavy T, Mannor S. Visualizing dynamics: from t-SNE to SEMI-MDPs. 2016. arXiv:1606.07112.
[98] François-Lavet V, Bengio Y, Precup D, Pineau J. Combined reinforcement learning via abstract representations. Proc AAAI Conf Artif Intell 2019;33(1):3582–9.
[99] McInnes L, Healy J, Melville J. UMAP: uniform manifold approximation and projection for dimension reduction. 2018. arXiv:1802.03426.
[100] Zhao Y, Fatehi A, Huang B. A data-driven hybrid ARX and Markov chain modeling approach to process identification with time-varying time delays. IEEE Trans Ind Electron 2017;64(5):4226–36.
[101] Watkins CJCH, Dayan P. *Q*-learning. Mach Learn 1992;8(3–4):279–92.
[102] Tsitsiklis JN, Roy BV. Analysis of temporal-difference learning with function approximation. In: Jordan MI, Petsche T, editors. Proceedings of the 9th International Conference on Neural Information Processing Systems; 1996 Dec 3–5; Denver, CO, USA. Cambridge: MIT Press; 1997. p. 1075–81.
[103] Gullapalli V. A stochastic reinforcement learning algorithm for learning real-valued functions. Neural Networks 1990;3(6):671–92.
[104] Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. In: Xing EP, Jebara T, editors. Proceedings of the 31st International Conference on International Conference on Machine Learning; 2014 Jun 21–26; Beijing, China; 2014. p. I-387–95.
[105] Levine S, Finn C, Darrell T, Abbeel P. End-to-end training of deep visuomotor policies. J Mach Learn Res 2016;17(1):1334–73.
[106] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, editors. Proceedings of the 27th International Conference on Neural Information Processing Systems; 2014 Dec 8–13; Montreal, QC, Canada. Cambridge: MIT Press; 2014. p. 2672–80.
[107] Konda VR, Tsitsiklis JN. On actor–critic algorithms. SIAM J Control Optim 2003;42(4):1143–66.
[108] Grondman I, Busoniu L, Lopes GAD, Babuska R. A survey of actor–critic reinforcement learning: standard and natural policy gradients. IEEE Trans Syst Man Cybern C 2012;42(6):1291–307.
[109] Grondman I, Buşoniu L, Babuška R. Model learning actor–critic algorithms: performance evaluation in a motion control task. In: Proceedings of the 51st IEEE Conference on Decision and Control; 2012 Dec 10–13; Maui, HI, USA. New York: IEEE; 2012. p. 5272–7.
[110] Costa B, Caarls W, Menasché DS. Dyna-MLAC: trading computational and sample complexities in actor–critic reinforcement learning. In: 2015 Brazilian Conference on Intelligent Systems; 2015 Nov 4–7; Natal, Brazil. New York: IEEE; 2015. p. 37–42.
[111] Langevin P. Sur la théorie du mouvement brownien. In: Comptes Rendus Hebdomadaires des Seances de l'Academie des Sciences. Paris: Gauthier-Villars; 1908. p. 530–3.
[112] Wang Z, Bapst V, Heess N, Mnih V, Munos R, Kavukcuoglu K, et al. Sample efficient actor–critic with experience replay. 2016. arXiv:1611.01224.

[113] Munos R, Stepleton T, Harutyunyan A, Bellemare MG. Safe and efficient off-policy reinforcement learning. In: Lee DD, von Luxburg U, Garnett R, Sugiyama M, Guyon I, editors. Proceedings of the 30th International Conference on Neural Information Processing Systems; 2016 Dec 5; Barcelona, Spain. Red Hook: Curran Associates Inc.; 2016. p. 1054–62.
[114] Schulman J, Levine S, Abbeel P, Jordan M, Moritz P. Trust region policy optimization. In: Proceedings of the 32nd International Conference on Machine Learning; 2015 Jul 7–9; Lille, France; 2015. p. 1889–97.
[115] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017. arXiv:1707.06347.
[116] Wu Y, Mansimov E, Grosse RB, Liao S, Ba J. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, et al., editors. Advances in neural information processing systems 30: 31st Annual Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. San Diego: Neural Information Processing Systems Foundation, Inc.; 2017. p. 5279–88.
[117] Grosse R, Martens J. A Kronecker-factored approximate fisher matrix for convolution layers. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd International Conference on International Conference on Machine Learning; 2016 Jun 19–24,;New York, NY, USA; 2016. p. 573–82.
[118] Martens J, Ba J, Johnson M. Kronecker-factored curvature approximations for recurrent neural networks. In: Proceedings of the 6th International Conference on Learning Representations; 2018 Apr 30–May 3; Vancouver, BC, Canada; 2018.
[119] Gruslys A, Dabney W, Azar MG, Piot B, Bellemare M, Munos R. The reactor: a fast and sample-efficient actor–critic agent for reinforcement learning. 2017. arXiv:1704.04651.
[120] Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, et al. Soft actor–critic algorithms and applications. 2018. arXiv:1812.05905.
[121] Fujimoto S, Van Hoof H, Meger D. Addressing function approximation error in actor–critic methods. 2018. arXiv:1802.09477.
[122] Ljung S, Ljung L. Error propagation properties of recursive least-squares adaptation algorithms. Automatica 1985;21(2):157–67.
[123] Doya K. Reinforcement learning in continuous time and space. Neural Comput 2000;12(1):219–45.
[124] Bellman R. Dynamic programming. Science 1966;153(3731):34–7.
[125] Borkar VS. An actor–critic algorithm for constrained Markov decision processes. Syst Control Lett 2005;54(3):207–13.
[126] Peters J, Schaal S. Natural actor–critic. Neurocomputing 2008;71(7–9):1180–90.
[127] Vrabie D, Lewis F. Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. Neural Networks 2009;22(3):237–46.
[128] Vamvoudakis KG, Lewis FL. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. Automatica 2010;46(5):878–88.
[129] Degris T, White M, Sutton RS. Off-policy actor–critic. 2012. arXiv:1205.4839.
[130] Bhasin S, Kamalapurkar R, Johnson M, Vamvoudakis KG, Lewis FL, Dixon WE. A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. Automatica 2013;49(1):82–92.
[131] Zhao D, Wang B, Liu D. A supervised actor–critic approach for adaptive cruise control. Soft Comput 2013;17(11):2089–99.
[132] Modares H, Lewis FL, Naghibi-Sistani MB. Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. Automatica 2014;50(1):193–202.
[133] Chang SJ, Lee JY, Park JB, Choi YH. An online fault tolerant actor–critic neuro-control for a class of non-linear systems using neural network HJB approach. Int J Control Autom Syst 2015;13(2):311–8.
[134] Kiumarsi B, Lewis FL. Actor–critic-based optimal tracking for partially unknown nonlinear discrete-time systems. IEEE Trans Neural Netw Learn Syst 2015;26(1):140–51.
[135] Song R, Lewis F, Wei Q, Zhang HG, Jiang ZP, Levine D. Multiple actor–critic structures for continuous-time optimal control using input-output data. IEEE Trans Neural Netw Learn Syst 2015;26(4):851–65.
[136] Allen C, Asadi K, Roderick M, Mohamed A, Konidaris G, Littman M. Mean actor critic. 2017. arXiv:1709.00503.
[137] Dhar NK, Verma NK, Behera L. Adaptive critic-based event-triggered control for HVAC system. IEEE Trans Industr Inform 2018;14(1):178–88.
[138] Fan QY, Yang GH, Ye D. Quantization-based adaptive actor–critic tracking control with tracking error constraints. IEEE Trans Neural Netw Learn Syst 2018;29(4):970–80.
[139] Wei Y, Yu FR, Song M, Han Z. User scheduling and resource allocation in HetNets with hybrid energy supply: an actor–critic reinforcement learning approach. IEEE Trans Wirel Commun 2018;17(1):680–92.
[140] Chen B, Wang D, Li P, Wang S, Lu H. Real-time 'actor–critic' tracking. In: Yang MH, Gool LV, Liu W, Wang XG, Kautz J, Shen CH, et al. editors. Proceedings of the European Conference on Computer Vision; 2018 Sep 8–14; Munich, Germany; 2018. p. 318–34.
[141] Radac MB, Precup RE, Roman RC. Data-driven model reference control of MIMO vertical tank systems with model-free VRFT and *Q*-learning. ISA Trans 2018;73:227–38.
[142] Yang Z, Chen Y, Hong M, Wang Z. On the global convergence of actor–critic: a case for linear quadratic regulator with ergodic cost. 2019. arXiv:1907.06246.
[143] Lv Y, Na J, Ren X. Online *H*∞ control for completely unknown nonlinear systems via an identifier–critic-based ADP structure. Int J Control 2019;92(1):100–11.
[144] Hou Z, Zhang K, Wan Y, Li D, Fu C, Yu H. Off-policy maximum entropy reinforcement learning: soft actor–critic with advantage weighted mixture policy (SAC-AWMP). 2020. arXiv:2002.02829.
[145] Zhang Y, Zhao B, Liu D. Deterministic policy gradient adaptive dynamic programming for model-free optimal control. Neurocomputing 2020;387:40–50.
[146] Schulman J, Moritz P, Levine S, Jordan M, Abbeel P. High-dimensional continuous control using generalized advantage estimation. 2015. arXiv:1506.02438.
[147] Shashua SDC, Mannor S. Trust region value optimization using Kalman filtering. 2019. arXiv:1901.07860.
[148] Shashua SDC, Mannor S. Kalman meets Bellman: improving policy evaluation through value tracking. 2020. arXiv:2002.07171.
[149] Su PH, Budzianowski P, Ultes S, Gasic M, Young S. Sample-efficient actor–critic reinforcement learning with supervised data for dialogue management. 2017. arXiv:1707.00130.
[150] Fu J, Kumar A, Nachum O, Tucker G, Levine S. D4RL: datasets for deep data-driven reinforcement learning. 2020. arXiv:2004.07219.
[151] Nair A, Srinivasan P, Blackwell S, Alcicek C, Fearon R, De Maria A, et al. Massively parallel methods for deep reinforcement learning. 2015. arXiv:1507.04296.
[152] Pavlov PI. Conditioned reflexes: an investigation of the physiological activity of the cerebral cortex. Ann Neurosci 2010;17(3):136–41.
[153] Huang B, Kadali R. Dynamic modeling, predictive control and performance monitoring: a data-driven subspace approach. London: Springer;

2008.
[154] Gonzalez RC, Woods RE. Digital image processing. 4th ed. London: Pearson Publishing Co.; 2018.
[155] Chen M, Radford A, Child R, Wu J, Jun H, Luan D, et al. Generative pretraining from pixels. In: Proceedings of the 37th International Conference on Machine Learning; 2020 Jul 12–18; online conference; 2020. p. 1691–703.
[156] Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014. arXiv:1412.6980.
[157] Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. In: Proceedings of 2017 International Conference on Engineering and Technology; 2017 Aug 21–23; Antalya, Turkey. New York: IEEE; 2017. p. 1–6.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Yours truly,

Oguzhan Dogru
Kirubakaran Velswamy
Biao Huang