Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

PPLC: Data-driven offline learning approach for excavating control of cutter suction dredgers

Changyun Wei [a,b], Hao Wang [a,b], Haonan Bai [a,b], Ze Ji [c,*], Zenghui Liu [a,b]

[a] College of Mechanical and Electrical Engineering, Hohai University, China
[b] Engineering Research Center of Dredging Technology of Ministry of Education, Hohai University, China
[c] School of Engineering, Cardiff University, Cardiff CF24 3AA, United Kingdom

ARTICLE INFO

Keywords:
Cutter suction dredgers
Excavating operation
Prediction
Deep reinforcement learning

ABSTRACT

Cutter suction dredgers (CSDs) play a very important role in the construction of ports, waterways and navigational channels. Currently, most of CSDs are mainly manipulated by human operators, and a large amount of instrument data needs to be monitored in real time in case of unforeseen accidents. In order to reduce the heavy workload of the operators, we propose a data-driven offline learning approach, named Preprocessing-Prediction-Learning Control (PPLC), for obtaining the optimal control policy of the excavating operation of CSDs. The proposed framework consists of three modules, i.e., a data preprocessing module, a dynamics prediction module realized by a Convolutional Neural Network (CNN), and a deep reinforcement learning based control module. The first module is responsible for filtering out irrelevant variables through correlation analysis and dimensionality reduction of raw data. The second module works as a state transition function that provides the dynamics prediction of the excavating operation of a CSD. To realize the learning control, the third module employs the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm to control the swing speed during the excavating operation. The simulation results show that the proposed framework can provide an effective and reliable solution to the automated excavating control of a CSD.

1. Introduction

In dredging projects, the dredging vessels or dredgers are required to excavate underwater soil, sand and rocks for the purpose of construction and maintenance of ports, waterways and navigational channels. According to the classification of dredgers, CSDs combine the advantages of mechanical and hydraulic dredgers, with the distinctive features of the cutter head and slurry pumps (Bai et al., 2019). To be specific, the cutter head can excavate rocks or hard soil so as to construct channel slopes with high accuracy, while the slurry pumps can transport dredged sediments to discharge areas through a long pipeline, as shown in Fig. 1. Thus, CSDs are widely used as an effective construction equipment for broadening and deepening rivers, lakes, and reservoirs, and the dredged materials are regarded as the resources for the development of coastal cities.

The excavating operation of a CSD is extremely complicated, and, currently, well-trained operators are required to manually manipulate the excavating process. As the rotating cutter head works underwater, the operator cannot directly observe the working states of the CSD. In order to avoid occasional accidents, the operators need to pay attention to monitoring a large number of instruments and meters. Therefore, driving a CSD is a labor-intensive task, and, especially in a night shift, fatigue driving often leads to safety issues (Wei et al., 2022). It is also a challenge task to develop a fully automated control system to cope with the uncertainties during the excavating operation.

Unmanned driving of ships has been a topic of extensive research in recent years (Liu et al., 2022), with numerous studies exploring the possibility of reducing operator workload through the application of intelligent auxiliary tools in the intelligent control of CSDs. An expert system is developed based on monitoring data that optimizes control strategy in response to real-time excavating conditions, providing valuable feedback and advice to the operator (Tang et al., 2008; Tang and Wang, 2008; Tang et al., 2009). The work (Yue et al., 2015) presents a quantitative classification model based on dredging materials for the prediction of dredging production. The monitoring data has also been used to evaluate the construction efficiency of CSDs (Li et al., 2018). The above studies employ monitoring data to analyse the dynamics of the excavating process and provide real-time suggestions to alert operator in case of inappropriate manipulations.

In this work, we focus on investigating an intelligent control approach to the excavating operation of CSDs, and a reinforcement learning agent is trained to manipulate the swing process of a CSD instead of a well-trained operator. However, training the agent using a real

* Corresponding author.
E-mail address: jiz1@cardiff.ac.uk (Z. Ji).

https://doi.org/10.1016/j.engappai.2023.106708
Received 6 December 2022; Received in revised form 27 May 2023; Accepted 21 June 2023
Available online xxxx

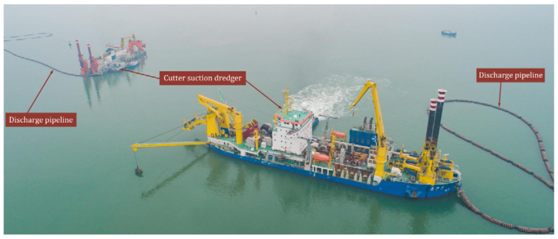


Fig. 1. The excavating process of cutter suction dredgers.

CSD in realistic dredging environments is not feasible. To overcome this challenge, we leverage large amounts of historical offline construction data to realize such an agent.

To utilize historical data, it is necessary to employ data mining techniques to preprocess the raw data. As the raw measured signals have the inherent limitations such as clutter (Evangelidis and Horaud, 2017; Zhang et al., 2017), repeatability (Zhang et al., 2018)and ambiguity (Dorrity et al., 2020), data preprocessing is a very important procedure of data mining to address the above problems (Wu et al., 2013). In the engineering domains, the raw signals collected from sensors are often noisy, and we need to denoise the data. For example, the Iterative Ensemble Filter (García-Gil et al., 2019) and the Ensemble Kalman Filter (Li et al., 2020) can be applied to preprocess the raw collected data. Once we have the reliable samples of the historical construction data, we can proceed to develop the learning agent for the automated control of the excavating operation.

Learning is a promising methodology that can utilize samples to recognize patterns. Among others, the Deep Reinforcement Learning (DRL) framework can address how an agent optimizes its control policy in uncertain environments. Currently, many scholars have carried out extensive studies on DRL algorithms for various applications, such as games (Silver et al., 2018), manufacturing scheduling (Lin et al., 2019), image analysis (Zhou et al., 2021), neural architecture search (Li et al., 2021, 2022) and robotics (Xu et al., 2018). Although the above successes have been achieved in a variety of domains, the applications of DRL are still limited to the situations where the states can be hand-crafted or the sensory inputs are low-dimensional (Mnih et al., 2015).

It is still a challenging task to apply DRL to real-world applications, such as the operation of CSDs. Firstly, a DRL agent needs to be trained with millions of learning steps, but we cannot afford to use a real CSD to train such an agent. Secondly, in robotics and games, a virtual simulator is usually ready-made or can be easily established to train the learning agent. However, the excavating process studied in this work contains many uncertain factors, and it is hard to build an accurate simulator to reflect the dynamic characteristics of a CSD. Thus, in order for the learning agent to obtain the optimal control policy of the excavating operation, we need to allow the agent to make trial-and-error interactions with an entity to collect experiences.

In this work, we aim at proposing a data-driven offline learning approach for the excavating operation control of a CSD, based on the historical dredging data from manual manipulations. To this end, we will first preprocess the raw collected data because many variables of the measured signals are irrelevant to the excavating operation. Afterwards, we will discuss how to develop a prediction model that is able to output the state transitions. This model works as a virtual simulator that provides feedback to the learning agent. Finally, the learning agent can interact with the prediction model to explore the state spaces so as to find the optimal control policy. The main contributions of this work are summarized as follows:

1. The methods described in the studies (Han et al., 2022; Wang et al., 2020; Bai et al., 2019; Li et al., 2018) only consider the prediction problem of slurry concentration of CSDs based on historical data. In contrast, we propose a Preprocessing-Prediction-Learning Control (PPLC) approach to address the control problem of the excavating operation of a CSD, and the optimal control policy is obtained via offline learning using historical data. A Deep Reinforcement Learning (DRL) module based on the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm is constructed to control the swing speed during the excavating operation.
2. The methods presented in the studies (Su et al., 2022; Wei et al., 2022; Kuhnle et al., 2021; Anderlini et al., 2020) need specific domain knowledge to design the state representations for modelling a Markov Decision Process (MDP). However, in our work a dimensionality reduction module is produced to filter out the variables irrelevant to the excavating operation and to denoise the raw collected data, which can avoid selecting main features through mechanism analysis and domain knowledge.
3. In conventional reinforcement learning methods described in the studies (Silver et al., 2018; Lin et al., 2019; Zhou et al., 2021; Li et al., 2022; Su et al., 2022), virtual simulators are usually ready-made or can be easily established to train the learning agent. However, since the excavating process contains many uncertain factors, it is hard to build an accurate simulator for the agent to collect experiences via trial-and-error interactions. In this work, the state transition function is realized by a Convolutional Neural Network (CNN), which can resolve the problem of the lack of an accurate simulator for interacting with the environments.
4. No method has been developed to directly train a control policy of the excavating operation of a CSD based on historical data generated by human operators. In this work, the agent can make use of historical data to find the optimal control policy, and the results show that our proposed approach can surpass the performance of well-trained human operators.

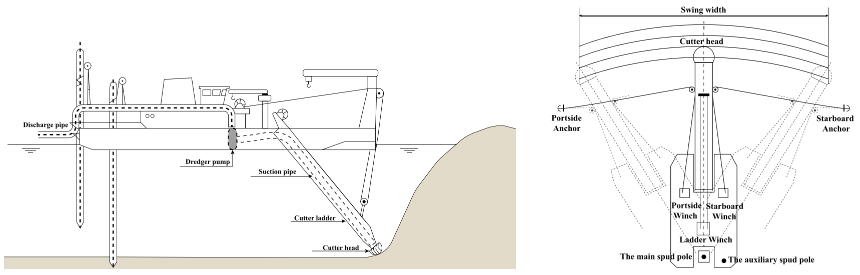
The paper is organized as follows. We first discuss the excavating problem of operating a CSD in Section 2, and then present our PPLC approach in Section 3. Then, we proceed to detail the data-driven offline learning algorithm in Section 4, and evaluate the performance of our approach and analyse the experiment results in Section 5. Finally, we summarize this work and discuss the future plans in Section 6.

2. Problem of excavating operation

The operation of a CSD is quite complicated, and the excavating process is the most tedious and recurring work for human operators. As depicted in Fig. 2, the excavating operation involves how to control the rotating cutter head so as to cut hard soil or rock into fragments in the seabed. Afterwards, the dredge pump can suck the dredged fragments into the pipeline and transport to a disposal zone. The cutter head is mounted on the cutter ladder that can be lowered down, and it can also swing around the main spud pole to excavate the soil in an arc trajectory. Therefore, once the ladder is lowered onto the mud surface, the operator needs to control the swing speed so as to adjust the production of dredged slurry.

During the excavating process, two spud poles are essential for the manipulation of a CSD. The main spud pole is deployed on a movable carriage, while the auxiliary spud pole is set out of the centreline of the CSD. Two anchors are placed on both sides of the dredger, and they are connected to the winches through cables. Thus, by pulling or slacking the corresponding cables, the CSD can achieve swing movements. To cut the soil on the seabed, the cutter head with the ladder moves around the main spud pole. In order to achieve this movement, the winch on one side needs to slack the cable, while the winch on the other side needs to pull the cable. When the cutter head reaches the maximum width, the two winches can change their rotation directions.

When the soil in front of the cutter head is excavated, the hydraulic cylinder connected to the main spud pole needs to push the dredger



(a) Front view.

(b) Top view.

Fig. 2. Illustration of the excavating process of a CSD.

forward so as to continue the excavation process. Therefore, the entire dredging process also includes switching the spud poles and moving the anchors. However, in the above mentioned two processes, the cutter head does not need to excavate the soil, and the time-consuming in the entire dredging process is relatively low. Thus, the above two processes are out of the scope of this paper, and we focus on the excavating process. It should be noted that during the excavating operation, the continuous control variable that needs to be taken into consideration is the swing speed of the cutter. Although the adjustment of the cutter rotation speed, ladder lowering depth, and the pole stepping length also need to be operated during the intervals of swing movement, these discrete control variables involve process optimization and do not require continuous control. Therefore, this paper only discusses how to control the swing speed of the cutter in order to improve the production of excavated sediments. In the dredging projects, the production W is the main indicator to evaluate the performance of the excavating operation, and it can be calculated by,

$$W = C_s(\pi r^2)V_f T, \tag{1}$$

where C_s indicates the concentration of the excavated slurry, r is the radius of the pipeline, V_f is the flow rate of the slurry mixture in the pipeline, and T is the working time. To improve the production, an intuitive idea is to increase the concentration of the excavated soil and the flow rate of the pipeline. However, these two variables are coupled and interdependent. If the concentration of the excavated slurry is too high, the flow rate of the transportation pipeline can be decreased. When the flow rate is too low, it will lead to the blockage of the pump or the pipeline. Moreover, we cannot enhance the flow rate by simply increasing the rotation speed of the pump, because the high flow rate will increase the wear of the pipeline and the consumption of pump energy.

Therefore, in this work, we aim at adjusting the swing speed by a reinforcement learning agent so that the concentration of the excavated soil can be maintained at a high level. At the same time, the flow rate of the pipeline needs to be stabilized within a reasonable range to prevent safety accidents. Moreover, we should also identify other safety concerns and take them into consideration in controlling the excavating process.

3. General framework of PPLC

In this section, we will propose the Preprocessing-Prediction-Learning Control (PPLC) approach for the excavating control of CSDs. As depicted in Fig. 3, the general framework of PPLC approach consists of three modules, i.e., the data preprocessing, the dynamics prediction module and the deep reinforcement learning module. In this work, since we intend to utilize the historical dredging data to construct an intelligent agent for the excavating process control, the high-dimensional raw and noisy data need to be cleaned up before we use them to obtain the dynamics model of the excavating process. Afterwards, the dynamics prediction module can work as an intermediate connector that provides the feedback about the coming state for the agent to decide an action. Based on the feedback, the learning agent can make interactive trial-and-error queries to accumulate the experiences for the control of the excavating process.

As mentioned before, the objective of the excavating operation is to maintain the concentration of dredged slurry at a high level. However, the raw and noisy historical data of a CSD contains more than 100 types of measured signals. Although we know that during manual operation, the driver usually only focuses on some key indicators, such as the degree of suction vacuum, motor current of the cutter head, motor current of underwater pump, and measured slurry concentration, this paper aims to explore whether these indicators can be discovered through data mining techniques instead of relying on domain knowledge. Thus, we will eliminate the variables that are irrelevant to the concentration by means of correlation analysis. Here we mainly adopt the Pearson linear correlation analysis and the MIC nonlinear correlation analysis to process the raw data. After this step, the data may still be redundant and noisy, so we will further reduce the dimensionality and noise of the data. In this work, we establish a Stack Denoising Autoencoder (SDAE) to obtain low-dimensional feature vectors.

The dynamics prediction module is responsible for predicting the dynamics of the excavating process. To be specific, we need to identify the state transitions, i.e., one-step prediction of state change. In this work, we adopt a CNN to output the coming state based on past n steps. Since each state at a time-step is represented as a feature vector, we use past n feature vectors to form a feature matrix that can be fed into the prediction network. In this way, the CNN model can extract the temporal features of past n steps to make predictions. Here, n equals to the number of dimensions of the main feature vector. In order to unify with the historical data of manual operation, in this paper one time-step also means one second.

Once the accuracy of dynamics prediction module satisfies the requirement of simulating the excavating process, the reinforcement learning agent can collect experiences through trial-and-error interactions. To this end, we need to decode the predicted vector of the next state using SDAE, and the decoded state can restore the physical information about the excavating process. Consequently, we develop a modified DRL algorithm based on TD3 to train the learning agent so as to obtain the optimal control policy.

3.1. Data preprocessing module

During the excavating process, the operator's primary concern is the concentration of the dredged slurry. For a CSD, there are morn than 100 types of measured data collected by the monitoring system, but not all of them are related to the slurry concentration. If we apply empirical formulas or semi-empirical formulas to determinate the key variables, the computation can be tedious, and the reliability and representativeness of selected variables cannot be guaranteed. Therefore, Correlation Analysis (CA) is carried out in this work to retain the key variables that are related to the slurry concentration.

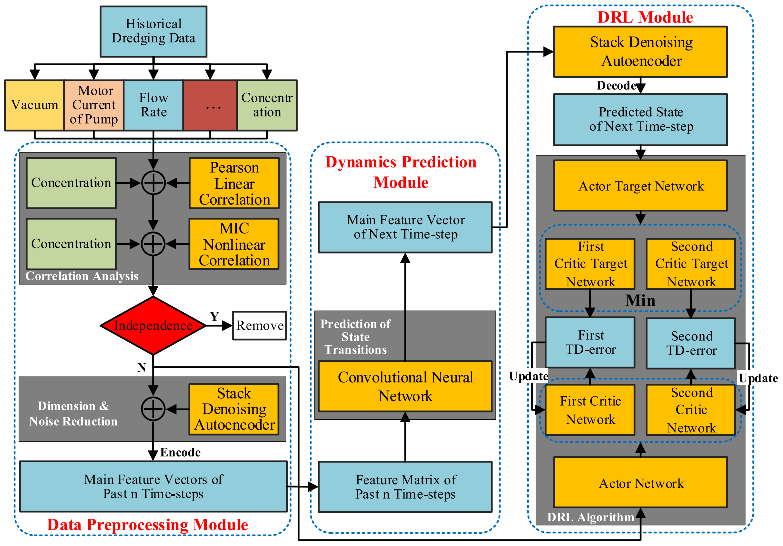

Historical Dredging Data
Vacuum
Motor Current of Pump
Flow Rate
…
Concentration
Concentration
Pearson Linear Correlation
Concentration
MIC Nonlinear Correlation
Correlation Analysis
Independence
Y
Remove
N
Dimension & Noise Reduction
Stack Denoising Autoencoder
Encode
Main Feature Vectors of Past n Time-steps
Data Preprocessing Module
Dynamics Prediction Module
Main Feature Vector of Next Time-step
Prediction of State Transitions
Convolutional Neural Network
Feature Matrix of Past n Time-steps
DRL Module
Stack Denoising Autoencoder
Decode
Predicted State of Next Time-step
Actor Target Network
First Critic Target Network
Second Critic Target Network
Min
First TD-error
Second TD-error
Update
Update
First Critic Network
Second Critic Network
Actor Network
DRL Algorithm

Fig. 3. The general framework of the proposed PPLC approach.

3.1.1. Correlation analysis

In order to identify the variables that can fully describe the dynamics of the excavating process, we adopt the Pearson linear correlation analysis and the Maximal Information Coefficient (MIC) (Reshef et al., 2011) analysis to measure the dependence of two-variable relationships. The Pearson coefficient is used to measure linear correlation, and can examine the degree of linear correlation between two variables. The calculation of Pearson coefficient is expressed by

$$\begin{aligned}\rho_{x,y} &= \frac{cov(x,y)}{\sigma_x\sigma_y}\\ &= \frac{E(xy)-E(x)E(y)}{\sqrt{E(x^2)-E^2(x)}\sqrt{E(y^2)-E^2(y)}},\end{aligned} \tag{2}$$

where x, y represent two types of dredging variables for calculating correlation, $cov(x,y)$ is the covariance between x and y, σ indicates the standard deviation, and E denotes the mathematical expectation. With respect to the concentration of the dredged slurry, all the other variables will be used to calculate the Pearson coefficient.

As the Pearson coefficient is mainly used to examine the linear correlation between two variables, the calculated results may be fluctuate for nonlinear variables. Therefore, we further adopt MIC to investigate the nonlinear relationships between two variables. Mutual information refers to the probability that a random variable changes as another known variable changes. Given a set of variables $N = \{x_i, y_i\}$, the mutual information of x_i and y_i is defined as

$$\mathrm{M}(x_i,y_i) = \sum_{x_i,y_i}\frac{p(x_i,y_i)}{p(x_i)p(y_i)}, \tag{3}$$

where $p(x_i,y_i)$ is the joint probability of x_i and y_i. In general, it is hard to directly apply mutual information for feature selection. With respect to the dredging data with high dimensions, we cannot normalize them by mutual information, so the nonlinear relationships among variables cannot be calculated conveniently. In comparison with mutual information, MIC overcomes this disadvantage and can obtain more accurate and extensive nonlinear correlations, and it can be calculated by

$$\mathrm{MIC}(x_i,y_i) = \max_{ab<B}\frac{\sum_{x_i,y_i}\frac{p(x_i,y_i)}{p(x_i)p(y_i)}}{\mathrm{logmin}(a,b)}, \tag{4}$$

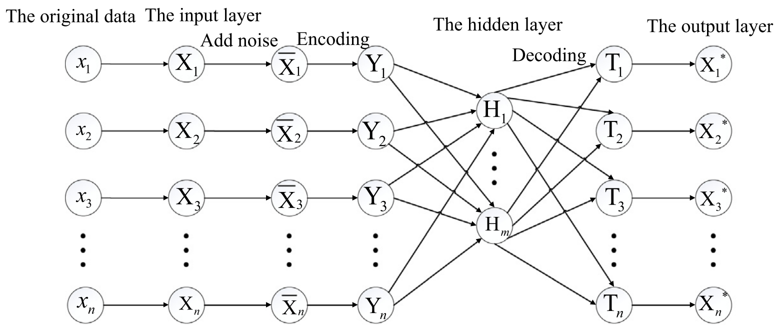
where B denotes the constraint threshold of grid partitioning, and it is set to the power of 0.6 of total data in this work. Here a and b represent the number of rows and columns of the grid, respectively.

3.1.2. Dimension and noise reduction

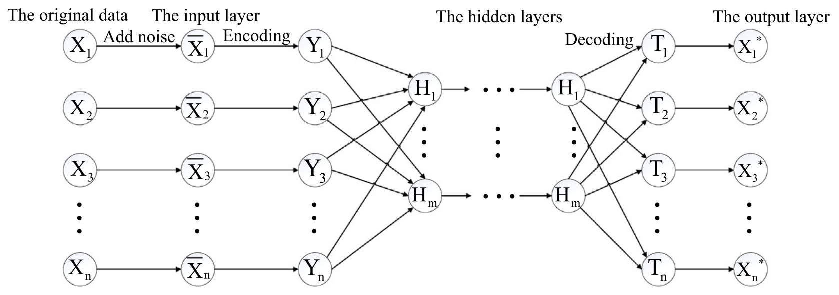
During the excavating process of a CSD, the vibration of the dredger's hall causes noise to interfere with the collected sensor data. Moreover, as discussed above, applying Pearson linear correlation and MIC nonlinear correlation can remove the variables that are independent of the slurry concentration. Those filtered variables may still have certain correlation and redundancy between each other. The dimensions of those filtered data still remain relatively high for constructing neural networks, and the sensor data are inherently noisy. Therefore, dimension reduction and noise reduction should be carried out on those filtered variables to obtain completely independent, low-dimensional and denoising feature vectors.

Autoencoder is a feedforward model in unsupervised learning. According to different constraints on loss functions, Denoising Autoencoder(DAE) (Vincent et al., 2008) is a special structure of autoencoder (see Fig. 4(a)). It can learn from the noisy input, and can realize the denoising function of the input data by continuously cycling and reducing the error value. The DAE not only reduces noise from data, but also encodes the input high-dimensional samples to obtain the independent main feature vector, which contains enough information, to improve the robustness of the model. However, when a single DAE contains multiple hidden layers, the adjustment of weights of hidden layers will decrease along with the increase of the number of hidden layers, because it uses the stochastic gradient descent to adjust the weights. After the processing of the first few hidden layers, the error will quickly converge to a local minimum. Due to this problem, it is difficult for a single DAE to escape the local optimum, and the subsequent hidden layers cannot play the role of noise reduction, which leads to low learning accuracy. In order to solve the above problem, this paper adopts SDAE formed by stacking multiple DAE. The benefits of SDAE are that it can perform unsupervised pre-training for each single hidden layer and supervised reverse tuning training for the whole structure. Moreover, it can also extract abstract features and solve the problem of over fitting of backward propagation and slow learning speed of a single DAE.

In this work, we adopt the Stack Denoising Autoencoder (SDAE) (Vincent et al., 2010) to process the filtered variables, as depicted in Fig. 4(b). The input layer of SDAE receives the noisy data $\overline{X}$ and encodes it to obtain signal Y, and the output layer reconstructs the signal Y to obtain the deconstructed data T. In this paper, the filtered dredging data will be processed by dimension and noise reduction, and the main feature vector encoded by SDAE only contains 7 elements. In this way, the original multi-dimensional data can be compressed into fewer dimensions without losing the information of the original data.



(a) Denoising Autoencoder (DAE).



(b) Stack Denoising Autoencoder (SDAE).

Fig. 4. The structure comparison between SDAE and DAE.

This can facilitate the establishment of the dynamics prediction module that will be discussed later. If necessary, the compressed feature vector can also be decoded so as to restore to the form of the original data with physical meanings.

3.2. *Dynamics prediction module*

As mentioned before, in order for a DRL agent to learn the optimal control policy, a virtual environment or simulator must be established beforehand. Trial-and-error interactions are essential for an agent to collect experiences. However, it is unsafe and expensive to directly use a real CSD to train the agent from scratch. Thus, we seek to construct a network that can work as a virtual environment or a simulator to inform the agent about the coming state. Specifically, in any state, if the agent takes an action, the network can predict the coming new state. In this way, the agent can evaluate whether it should take this action or the other possible options. In this work, we will construct a CNN structure, which contains multi-dimensional temporal feature vectors, to be responsible for predicting state transitions.

Traditional CNN is a type of artificial neural network that requires deep structure and convolution computation. The CNN structure includes an input layer, hidden layers and an output layer. Based on the functionalities, the hidden layers can also be divided into nonlinear, pooling and full connection layers. The input layer of CNN can be multidimensional data. For example, if the input data are pixels of an image, the input features need to be standardized and the original pixel values distributed in [0,255] should be normalized to [0,1]. The functionality of convolutional layers is to extract features from input data. In image process, the first convolutional layer may only extract some low-level features, such as edges, lines and angles, while more layers can continuously extract complex features iteratively based on low level features.

Each convolutional layer contains multiple kernels that are used to obtain various features. To this end, the convolution operation is carried out with the input data of convolutional check, and then the nonlinear activation function is applied to each result of this calculation. The pooling layer is responsible for compressing the amount of data processed by convolutional layers, as well as retaining main features and reducing the amount of network parameters and computation complexity. At the same time, dimension reduction realized by pooling layers can also allows CNN to extract a wider range of features. The common pooling operations includes mean and max pooling. After several convolutional and pooling operations, CNN can achieve feature extraction and compression of input signals. Afterwards, one or more full connection layers can be used to obtain global information, and the output of the last full connection layer is passed to the output layer. Finally, softmax logistic regression function can classify the outputs, which calculates the probability distribution of various categories.

In the CNN structure, a number of hyper-parameters determines the network configuration. We can use 1D convolution kernels to extract spectral features, and 2D convolution kernels to extract spatial features. The default inputs of traditional CNN are images, which can be easily establish 3D structure by combining spatial and spectral information in feature cubes. In Section 4.2, we will detail our CNN model for predicting the state transitions.

3.3. *Deep reinforcement learning model*

Deep Reinforcement learning (DRL) provides an interactive paradigm in which an artificial agent can interact with the environment to obtain experiences so as to improve its behaviours. Performing an action, the agent can receive positive or negative rewards from its environment. A RL task is a sequential decision making problem modelled by Markov Decision Process (MDP), and the agent needs to select action $a_t \in A$ to perform based on the current state $s_t \in S$ at each time-step t. Consequently, the action executed by the agent will also change the environment, and the agent receives a scalar immediate reward r_t. The agent's behaviour is determined by policy $\pi(a_t|s_t)$, which maps states to probability distribution over actions. The state transition function is denoted by $P(s_{t+1}|s_t, a_t)$, indicating the transition probability from state s_t to s_{t+1} by performing action a_t. For an episodic problem, a trajectory τ is formed when the agent reaches a terminal state, and the accumulated reward is defined by

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{5}$$

where $\gamma \in (0, 1]$ denotes the discount factor. Thus, the objective of the agent is to maximize the expectation of this long-term accumulated return, rather than an immediate reward.

Meanwhile, in order to evaluate the expected return of the policy π, two value functions are introduced: state value function $V^{\pi}(s)$ and state–action value function (or called q-value function) $Q^{\pi}(s, a)$. The state value function represents the expected total return from executing policy π starting from state s,

$$V^{\pi}(s) = \mathbb{E}_{\tau}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, \pi]. \tag{6}$$

Comparatively, the q-value function represents the expected total return of performing action a in the initial state s and then executing policy π

$$Q^{\pi}(s, a) = \mathbb{E}_{\tau}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi]. \tag{7}$$

Therefore, the goal of the agent is to find the optimal control policy to maximize the expected total future discounted return, $J(\pi) = \mathbb{E}\left[R_t|\pi\right]$. The policy π is parametrized by θ, and we can use policy gradient theorem to adjust the parameters,

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{s\sim\rho^{\pi}, a\sim\pi_{\theta}} \left[\nabla_{\theta} \log \pi_{\theta}(a|s) Q^{\pi}(s, a)\right], \tag{8}$$

where ρ^{π} means the state distribution, and it also depends on the policy parameters θ.

However, many well-known DRL algorithms still have some limitations, such as the need of long-term interaction with the environment to collect experiences for finding the optimal policy. In realistic applications, we cannot afford the cost of trial-and-error interactions with real entities, and it is also difficult to construct a perfect simulator to replace the entities. With regard to the excavating control of a CSD, even well-trained professionals are afraid to fully explore the state space, as this may lead to uncertain safety accidents. Therefore, our proposed offline learning approach based on DRL should address this problem.

4. Implementation of PPLC

In this section, we will detail how to realize and implement the proposed offline learning approach, i.e., PPLC. Since the operation of the excavating process is continuous, the agent modelled by DRL should also work in continuous state and action spaces. With regard to continuous control, many DRL algorithms still have the problem of overestimation caused by function approximation error, and the nature of time-difference learning further exaggerates this problem. The popular Deep Deterministic Policy Gradient (DDPG) algorithm is no exception. As DDPG contains the critic and the actor networks, the functionality of the actor networks is to continuously find the maximum Q value, regardless of whether the Q value is overestimated. Thus, in order to address the overestimation problem caused by function approximation, this paper employs the Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018), which is modified from DDPG and can greatly improve both learning speed and the performance.

4.1. State and action space

As discussed in Section 3.3, the agent needs to select an action to perform at each state. Thus, we need to define the state and action spaces for the excavating process of a CSD. Since there are more than 100 types of measured data collected by the monitoring system of a CSD, we need to pick some of the main feature variables so as to reduce the dimensionality of the state space. In this work, we do not manually select the elements of the state space via domain knowledge. In Section 3.1, we have detailed how to process the raw collected data by dimensionality and noise reduction. The purpose of this procedure is to identify the feature vector that contains enough information about the excavating process. The main feature vector only contains 7 elements that can retain enough information about the dredging dynamics, and can also eliminate redundant variables.

With regard to the action space of the excavating process, the agent needs to perform an action to response to the dynamics of the environment. Here the agent needs to adjust the swing speed that should be continuously manipulated, since it determines the amount of the dredged materials by the cutter head. It should be noted that the exploration of the state and action spaces should be regulated because of safety concerns in the excavating process.

4.2. Transition function

When the agent performs an action at a state, the transition function should specify the coming state. In this work, the transition function is realized by a CNN predicting model. Since the main feature vector only contains 7 elements, we construct a matrix with 7 × 7 elements as the input layer, as depicted in Fig. 5. Thus, the input layer includes the feature vectors of the past 7 time-steps, and the predicted coming state will be generated based on time-series observations.

The pooling layer of the conventional CNN model may ignore the correlation between local and the whole information, and the gradient descent algorithm may converge to local minimum. On the basis of the conventional CNN structure, we have made three main improvements. Specifically, this work employs soft pooling and Adaptive Moment Estimation (ADAM) optimization, and we also use residual convolution layers to replace traditional convolution layers.

Soft pooling can retain important information of feature maps while maintaining efficient computing. The core idea lies in the use of softmax to calculate the eigenvalue weight of region O,

$$w_i = \frac{e^{h^i}}{\sum_{j\in O} e^{h^j}}, \tag{9}$$

where w_i is the weights of the minimum unit kernel h of the ith term in region O, and it can ensure the transmission of important features. Meanwhile, the output of the pooling layer is obtained by the sum of all weighted activations in the kernel neighbourhood O. Compared to other methods based on maximum pooling or average pooling, soft pooling uses softmax to produce normalized results with a probability distribution proportional to each activation values relative to the adjacent activations value.

The ADAM method differs from traditional stochastic gradient descent method that keeps a constant learning rate. In comparison, ADAM can compute adaptive learning rates by calculating the first and second order exponential weighted average estimates of gradients,

$$\begin{cases} g_t \leftarrow \Delta_{\theta} f_t(\theta_{t-1}) \\ m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t \leftarrow \dfrac{m_t}{1-\beta_1^t} \\ \hat{v}_t \leftarrow \dfrac{v_t}{1-\beta_2^t} \\ \theta_t \leftarrow \theta_{t-1} - \alpha \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}, \end{cases} \tag{10}$$

where θ indicates the model parameters, f is the loss function, and g_t means the initialization gradient. β_1 denotes the first order exponential weighted decay rate, while m_t means the first order exponential

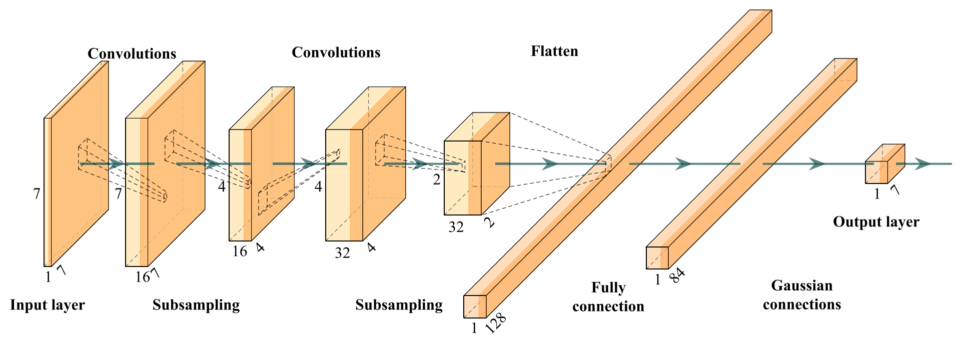


Fig. 5. The structure of CNN for predicting the coming state.

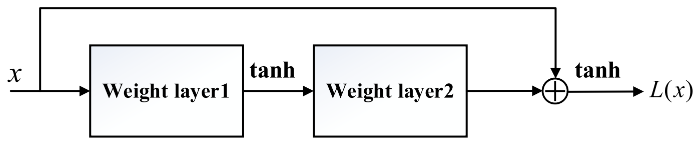


Fig. 6. The structure of residual convolutional layer.

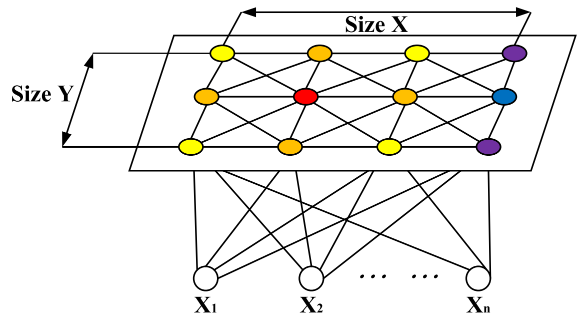


Fig. 7. The basic structure of the SOM networks for indicator extraction.

weighted average estimates. Besides, β_2 indicates the second order exponential weighted decay rate, while v_t denotes the second order exponential weighted average estimates, i.e., the mean values of the squares of the components of g_t. Thus, $\hat{m}_t$ and $\hat{v}_t$ indicate the bias-corrected first and second moment estimates after deviation correction, and the correction coefficients are $\frac{1}{1-\beta_1^t}$ and $\frac{1}{1-\beta_2^t}$, respectively.

Conventional convolution layers are often plagued by information loss or attenuation during feature sample information processing and transmission due to convolution operations. However, the residual convolution layer addresses this issue by establishing a bypass branch that directly connects feature sample information to the subsequent weight layer in the input process, thereby preserving the integrity of the feature information. Additionally, the input bypass branch ensures that the activation function *tanh* does not suffer from gradient disappearing problems when deep networks perform differential operations multiple times, thus ensuring the accuracy of deep convolution layers in processing feature data. Moreover, the entire network model only needs to learn the input and can still achieve end-to-end backward propagation without increasing the complexity of network calculation, thereby resolving the degeneration problem caused by deep convolution operation. This is illustrated in Fig. 6. With these three improvements, the revised CNN model can output a predicted state vector at each time-step for the agent to evaluate the possible actions.

4.3. Reward function

In RL task, as the objective of the agent is to maximize the accumulated reward, we need to specify the reward function so as to evaluate the performance of each step. Such an evaluation should take account of safety concerns of the excavating process. For instance, the current of the cutter motor cannot exceed the maximum value. On the other hand, we also hope that the main indicators of the excavating process can be kept at a desired state, and the excavating concentration can be maintained at a high level.

However, the feature vector used in the CNN model does not have any physical meanings, and we cannot directly evaluate the performance of each time-step based on the resulting state information. Therefore, when designing the reward function, we cannot directly utilize the main feature vector. Instead, we will first decoded the main feature vector by SDAE, and then select the key parameters, which are needed to construct the reward function, through the following Self-Organizing Map (SOM) network.

4.3.1. Indicator extraction by SOM

The basic structure of the SOM network is shown in Fig. 7, which consists of an input layer and a competition layer (an output layer). The number of neurons in the input layer is n, and the competition layer is a one-dimensional or two-dimensional plane array composed of $M = xy$ neurons. The network is fully connected, which means that each input node is connected to all output nodes.

In SOM, the neurons are connected with each other according to the input vector X and the weight vector w. The two vectors have to be normalized to $\widetilde{X}$ and $\widetilde{w}$ in the competition layer. In the learning process, the weight vector in the competition layer is compared with the current input vector $\widetilde{X}_i$. When the distance is the smallest, the neuron (weight vector) becomes the winner, and the distance is calculated based on the cosine similarity,

$$d_{som} = \widetilde{w}_{j*}^T \widetilde{X}_i = \max_j(\widetilde{w}_j^T \widetilde{X}_i) = \frac{w_j^T \widetilde{X}_i}{\|\widetilde{w}_j^T\| \|\widetilde{X}_i\|}, \tag{11}$$

where $\widetilde{w}_j$ indicates an neuron weight of the competition layer. At the same time, the weights of the nodes in the winning neighbourhood are updated iteratively. The updating strategy is that the closer nodes to the winning neighbourhood will get greater updating range. We also need to consider an updating constraint g for each node, which can be obtained by Gaussian function as follows,

$$g(i) = e^{-\frac{(c_x-a)(c_y-b)}{2\delta^2}} e^{-\frac{(c_x-a)(c_y-b)}{2\delta^2}}, \tag{12}$$

where a, b are the horizontal and vertical coordinates of the input data, and c_x, x_y are the horizontal and vertical coordinates of the active nodes of the contention layer.

In addition, the purpose of node updating is to make the winning node closer to the samples, and thus the following formula is adopted to update the weights,

$$\begin{cases} w_{j*}(t+1) = \widetilde{w}_{j*}(t) + \alpha g(i)(\widetilde{X} - \widetilde{w}_{j*}) \\ w_j(t+1) = \widetilde{w}_j(t), \end{cases} \tag{13}$$

where $j \neq j^*$, and α denotes the learning rate.

4.3.2. Rewards with safety concerns

According to our investigation of manual operation of the CSD studied in this work, the monitoring system will alert the operator on the screen when some indicators exceed the permitted values. For human operation, such warning messages can prompt the operator to take immediate corrective actions. However, due to the nature of manual operation, it is impossible for humans to consistently concentrate on monitoring multiple indicators, resulting in occasional occurrences where certain indicators may exceed the warning limits of safety constraints. Therefore, some situations in the historical data have actually exceeded safety concerns of human operators.

In this work, since we utilize the historical data generated by human operators to train the state transitions model, the state space certainly includes those situations that have exceeded safety concerns. In order for the RL agent to be aware of the safety consciousness that the human operators also care about, we take account of safety concerns for the reward function.

Here we will use the indicator extracted in Section 4.3.1 to design an evaluation strategy for designing the reward function. Since the measurement scales of different parameters are various, we have to normalize them so as to eliminate the differences between absolute values,

$$\hat{h}_t^i = \frac{h_t^i - h_t^{\min}}{h_t^{\max} - h_t^{\min}} \quad (i = 1, 2, \dots, m), \tag{14}$$

where m indicates the dimension of the evaluation vector, and $\hat{h}_t^i$ is the normalized value of the ith term of the evaluation vector at time t. $h_t^{\min}$ and $h_t^{\max}$ represents the minimum and maximum values of the evaluation vector, respectively.

In order for the agent to move towards the desired state, we also need to evaluate the movement of the current state. Thus, we define r_t^j to reflect the tendency towards the desired state,

$$r_t^j = \begin{cases} 0 & (d_t < d_{t-1}) \\ -(d_t - d_{t-1}) & (d_t \geq d_{t-1}), \end{cases} \tag{15}$$

where d_t is the Euclidean distance at time-step t between the current evaluation vector $\hat{h}_t$ and the target evaluation vector $\hat{h}_{\text{goal}}$, and it can be calculated by

$$d_t = \|\hat{h}_t - \hat{h}_{\text{goal}}\| = \sqrt{\sum_{i=1}^{m} (\hat{h}_t^i - \hat{h}_{\text{goal}}^i)^2}. \tag{16}$$

Then, the rewards at each time-step can be defined by

$$r_t^{\text{total}} = r_t^j + r_t^{\text{safe}}, \tag{17}$$

where the second term r_t^{safe} indicates the safety concerns, and it is defined as follows,

$$r_t^{\text{safe}} = \begin{cases} 0 & (\text{safe}) \\ -d_{\text{safe}} & (\text{otherwise}). \end{cases} \tag{18}$$

Here d_{safe} indicates the Euclidean distance between each parameter and the corresponding safety threshold,

$$d_{\text{safe}} = \|\hat{h}_t - \hat{h}_{\text{safe}}\| = \sqrt{\sum_{i=1}^{n} (\hat{h}_t^i - \hat{h}_{\text{safe}}^i)^2}. \tag{19}$$

The selection of the safety thresholds is obtained by using the confidence degree of Gaussian distribution combined with the professional knowledge and recommendations of human operators. During the training process, the agent may exceed the permitted values as it needs to try different possibilities. Similarly to how manual operation may receive warnings from the monitoring system of a CSD, if the agent's actions cause the safety constraints to be exceeded during the training process, it will receive a negative penalty signal. Eventually, through multiple rounds of iterative training, we expect that the learning agent can obtain the optimal control policy that will not trigger these safety constraints like an exceptional human operator who rarely triggers safety warnings. In this work, the safety thresholds consider the slurry concentration, the motor current of cutter head, the motor current of the underwater pump, the degree of suction vacuum, and the pipeline flow rate.

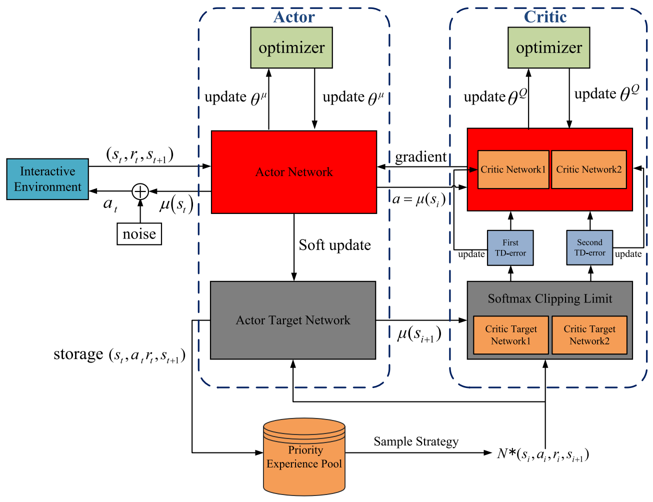


Fig. 8. A schematic diagram of the TD3 network structure.

4.4. Data-driven offline DRL algorithm

The classic DDPG algorithm contains four networks, i.e., two actor networks and two critic networks. The target actor network will select action a_t to perform according to the current state s_t and the current policy π. The next coming state s_{t+1} is predicted by the transition function, and then we can calculate the reward r_t (see Equ. (17)) of performing the action a_t. Then, the experience (a_t, s_t, r_t, s_{t+1}) can be stored in the replay buffer B, and the current critic network is updated by minimizing the loss,

$$L(\theta^Q) = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i | \theta^Q)^2), \tag{20}$$

where y_i can be obtained by

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}). \tag{21}$$

Here $\theta^{Q'}$ and $\theta^{\mu'}$ represent the parameters of the target critic network and the target actor network, respectively. The current actor network is updated by the gradient methods as follows,

$$\nabla_{\theta^\mu} J(\theta^\mu) \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=a_i} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s^i}. \tag{22}$$

Although DDPG sometimes can achieve excellent performance, it is often vulnerable to hyper-parameters and other types of adjustments. For instance, Q functions of DDPG are generally overestimated. To address this issue, this paper adopts the TD3 algorithm to serve as the control mechanism for the swing speed manipulation. While the conventional TD3 algorithm involves the implementation of a double network to mitigate the overestimation of Q value, it also leads to an underestimation of this value. Additionally, the conventional TD3 algorithm employs a random sampling strategy from the memory experience pool to train the model, which may result in the inclusion of irrelevant or negative samples. To address these issues, we consider two improvements to enhance the conventional TD3 algorithm, as depicted in Fig. 8.

1. To reduce the influences of invalid and negative samples on model training, this paper employs the prioritized replay buffer

mechanism for sampling,

$$p_i = \frac{(K_i)^\zeta}{\sum_{i=1}^{N}(K_i)^\zeta}, \tag{23}$$

where p_i represents sampling probabilities, K_i denotes the priority, and ζ is the power index converting the importance of TD-error to priority.

2. The softmax operator is introduced to limit the smaller Q-values generated by the double target evaluation network as follows,

$$y_t = r_t + \gamma \mathrm{softmax}(Q_{\theta_i'}(s_{t+1}, \pi_{\phi_1}(s_{t+1}))). \tag{24}$$

In conclusion, the overall data-driven offline learning algorithm based on TD3 for the excavating operation is shown in Algorithm 1. Based on this algorithm, we can allow the agent to interact with the state transition prediction model to learning the optimal policy.

Algorithm 1 The data-driven offline learning algorithm based on TD3.

```
1: Inputs: $\theta_1$ parameters of first initial critic network $Q_{\theta_1}$; $\theta_2$ parameters of second initial critic network $Q_{\theta_2}$; $\theta_1'$ parameters of first target critic network $Q_{\theta_1'}$; $\theta_2'$ parameters of second target critic network $Q_{\theta_2'}$; $\phi$ parameters of initial actor network $Q_\phi$;$\phi'$ parameters of target actor network $Q_{\phi'}$; $d$ is the update frequency of actor network.
2: Initialize priority replay buffer $B$
3: for $episode$ $e = 1$ to $M$ do
4:     Initialize environmental state $s_0$
5:     for $step$ $t = 1$ to $T$ do:
6:         Select action with exploration noise $a_t \leftarrow \pi_{\phi'}(s_t) + \epsilon_t, \epsilon_t \leftarrow \mathcal{N}(0, \sigma)$
7:         Obtain virtual state $s_t^*$ by reducing the dimension of real state $s_t$
8:         Get new virtual state $s_{t+1}^*$
9:         Decode and get new real state $s_{t+1}$
10:        Get reward $r_t$
11:        Store transition tuple $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{B}$
12:        If the capacity is full, the oldest transformation is overwritten.
13:        Sample a minibatch of transitions $(s, a, r, s')$ from $B$ with prioritization.
14:        $a \leftarrow \pi_{\phi'} + \epsilon, \epsilon \sim \mathrm{clip}(\mathcal{N}(0, \sigma), -c, c)$
15:        Set $y_i = r_t + \gamma \mathrm{softmax}(Q_{\theta_i'}(s_{t+1}, \pi_{\phi_1}(s_{t+1})))$
16:        Update the critics $\theta_i \leftarrow \min_{\theta_i} \frac{1}{N} \sum (y - Q_{\theta_i}(s, a))^2$ $(i = 1, 2)$
17:        if $t$ mod $d$ then
18:            Update the actor $\phi$ by the deterministic policy gradient:
19:            $\nabla_{\theta\phi} J(\theta^\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
20:            Update the target critic networks $\theta_i'$ and the target actor network $\phi_i'$
21:        end if
22:    end for
23: end for
```

5. Evaluation and results

In this section, we will first present the results of data preprocessing, and then discuss the prediction accuracy of the state transitions. Then, we will demonstrate the performance of the DRL agent in controlling the excavating process.

In this work, in order to measure the accuracy of neural network models, we adopt the Mean Absolute Error (MAE), Root Mean Square Error (RMSE), R-Square (R^2) and Explained Variance Score (EVS) as the evaluation indicators. The calculation of them are summarized as follows,

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m} |(y_i - y_i')|, \tag{25}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - y_i')^2}, \tag{26}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - y_i')^2}{\sum_{i=1}^{m}(y_i - \bar{y}_i)^2}, \text{and} \tag{27}$$

$$\mathrm{EVS} = 1 - \frac{\mathrm{Var}\{y - y'\}}{\mathrm{Var}\{y\}}. \tag{28}$$

Among them, MAE and RMSE are the indicators for longitudinal error evaluation, which mainly measure the overall performance and deviation of the model in long-term operation. In comparison, R^2 and EVS are the indicators for lateral error evaluation, which mainly measure the overall fitting performance of the model. The smaller the MAE and RMSE indicators are, the smaller the deviations between the real values and the estimations are. The value ranges of R^2 and EVS are between [0,1]. When they are closer to 1, the prediction and fitting ability of the model is stronger, and the model performance is better.

5.1. Data preprocessing

The raw dataset was collected by the central monitoring system of a CSD, and the historical data were generated by the manipulations of well-trained operators. The raw dataset contains 43200 observations (12 h), and the original dataset contains more than 100 measured signals. It should be noted that when a CSD completes a swing movement, it needs to be pushed forward a step with the help of the spud poles. In addition, the anchors for the swing movement also need to be moved irregularly. In these situations, the operator does not need to adjust the swing speed. According to our analysis of historical data, these intermittent stops occupy a relatively small amount of time in continuous operation, and the dredge pumps are usually still in normal working condition, as shown in Fig. 9. We can find that when the swing operation stops, the response of the slurry concentration in the pipeline is relatively lagging behind.

Therefore, we believe that all these collected data are useful for describing the dynamic characteristics of the dredging system. The raw dataset is divided into training and testing sets, where 34560 observations (80% of the dataset) have been used for training the networks, and the remaining 8640 observations (20% of the dataset) are used for evaluating the performance.

Correlation analysis is conducted between the raw dataset and the dredging concentration, including the Pearson linear analysis and the MIC nonlinear analysis. To filter out the data that are not relevant to the dredging concentration, we set the threshold as 0.25, as shown in Fig. 10. We combine the results of linear and nonlinear correlation analysis and finally obtain 25 feature variables that are highly correlated with the dredging concentration.

Although we can directly remove the irrelevant variables independent of the dredging concentration, the remaining variables may still be redundant to construct the networks for state transition prediction. At the same time, the noisy dataset should also be processed. Therefore, the filtered feature variables related to dredging concentration are fed into the SDAE model for dimensionality reduction and denoise processing, and we finally obtain a feature vector with 7 elements that are independent and can contain enough information of the dredging dynamics.

Here we also verify the accuracy of the SDAE model, since the output feature vectors should contain enough information about the original dredging dynamics. In this work, we decode the feature vectors and restore the dredging dataset to evaluate the accuracy of the SDAE model. Table 1 shows the performance metrics of the SDAE model.

We can see that the deviations of MAE and RMSE are only 1.67 and 2.11, respectively, and R^2 and EVS values both reach to 0.99. The above results prove that, with respect to the encoding and decoding process of the SDAE model, the errors between the original dataset and the restored data are very small. In addition, the fitting performance of the SDAE model is acceptable for dimensionality reduction and denoising.

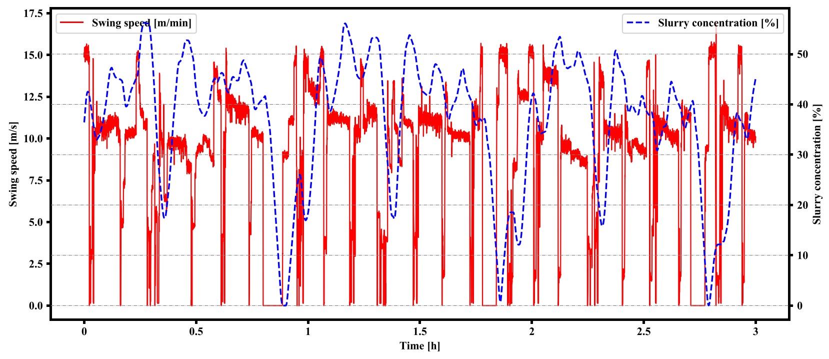
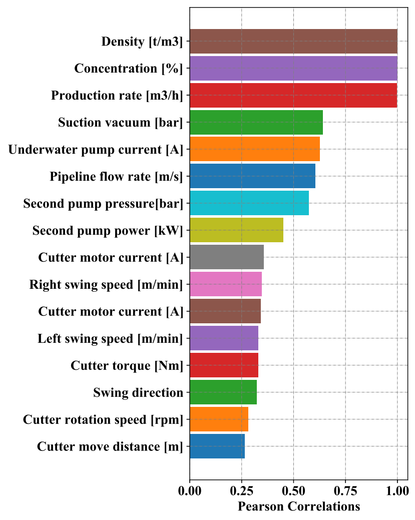
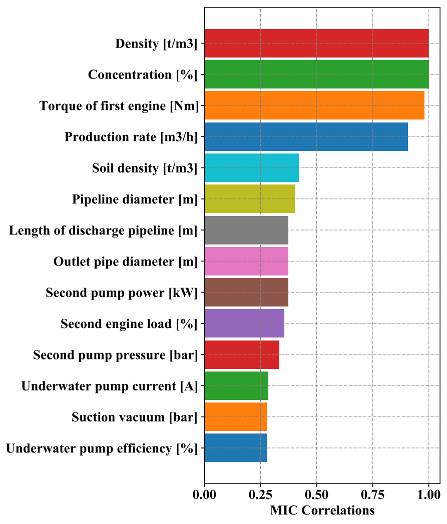


Fig. 9. The curves of swing speed and the slurry concentration of the historical data with 3 h manual operation.



(a) Pearson correlations.



(b) MIC correlations.

Fig. 10. Pearson and MIC correlation analysis of the raw dataset with respect to the dredging concentration.

Table 1
The accuracy evaluation of the SDAE model for data preprocessing.

MAE	RMSE	R^2	EVS
1.67	2.11	0.99	0.99

Table 2
The accuracy evaluation of our CNN model for state transition prediction.

MAE	RMSE	R^2	EVS
0.846	1.045	0.98	0.98

5.2. State transition prediction

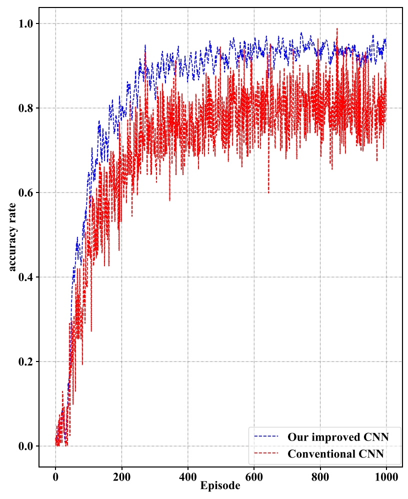
As mentioned before, the state transition function of DRL should output the coming state, and in this work we employ a CNN model to predict the coming state. Here we also need to demonstrate the accuracy of our CNN model that will be compared with a baseline, i.e., the conventional CNN model. Fig. 11 depicts the accuracy and loss curves during the training episodes. We can see that the accuracy of the our improved CNN model quickly converges to 0.95 with less fluctuation. However, the convergence speed of the conventional CNN model is slow, and it can only converge around 0.8 with large fluctuation. At the same time, with the increase of training episodes, the loss value of our improved CNN model can be reduced to 2.5, which is only one third of the loss value of the conventional CNN model. Thus, we can say that the accuracy of our optimized CNN model is higher than that of the conventional CNN structure.

Moreover, we also quantify the accuracy of our CNN model, and the results are listed in Table 2. The deviations of MAE and RMSE are only 0.846 and 1.045, respectively. In addition, R^2 and EVS are 0.98. Thus, we can claim that the improved CNN model has high accuracy and fitting ability to provide state transitions.

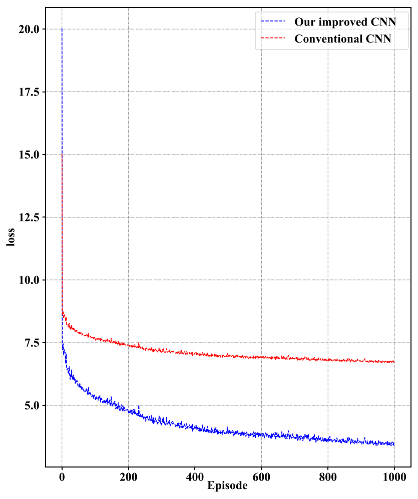
5.3. Control performance of DRL agent

In order to evaluate the control performance of the proposed TD3 based DRL algorithm, we have also implemented the classic DDPG algorithm as a baseline. As mentioned before, we have to consider the safety concerns in designing the reward function. We employ the SOM network to screen the historical data and find out the variables that contribute to the dredging concentration. In the reward function, those variables are selected to impose the safety concerns. The specific contribution degrees of those variables are listed in Table 3.

In dredging process, the dredging concentration is reflected by the slurry density, which is usually measured by a nuclear-based gamma densitometer. We should also define a safety threshold for the measured density, since the pipeline can be blocked if the slurry density is too high. The motor current of the cutter can also reflect the cutting force of soil, and it cannot exceed the permitted value. The dredged materials need to be sucked up by the underwater pump, and, thus, the motor



(a) Accuracy.



(b) Loss.

Fig. 11. Comparison of accuracy and loss between the conventional CNN and our improved CNN.

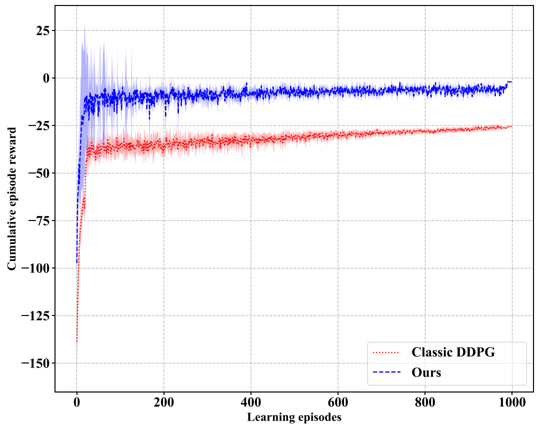


Fig. 12. Comparison of cumulative rewards of the classic DDPG and our algorithm during the learning episodes.

Table 3
Selected variables by SOM for safety concerns in the reward function.

Variables	Contribution degree
Density	0.1376
Motor current of cutter	0.0643
Motor current of underwater pump	0.0626
Suction vacuum	0.0481
…	…
Outlet pressure of second pump	0.0013

current of the underwater pump can also reflect the amount of slurry in the pipeline. Based on the selected variables and suggestions of experienced operators, we consider five safety concerns: concentration C_w, motor current of cutter I_c, motor current of underwater pump I_p, suction vacuum value D_v and pipeline flow rate V_f.

Table 4 lists the hyper-parameters used in our learning algorithm, where α denotes the learning rate of the actor network, β indicates the learning rate of the critic network, and γ is the discount factor of future rewards. τ indicates the soft update rate, and z represents the delay update rate of target action network.

We compare the cumulative reward curves of our learning algorithm with the classic DDPG in Fig. 12. It can intuitively reveal how the two algorithms converge to the optimal desired state. At the beginning of the learning episodes, the cumulative rewards of each episode are negative, which means that the trajectory of each episode is not optimal and has violated the safety concerns according to the reward function. Thus, we can conclude that our learning algorithm outperforms the classic DDPG algorithm with respect to convergence.

Table 4
Hyper-parameters of our learning algorithm.

Hyper-parameter	Value
Maximum Steps	500
Maximum episodes	1000
Batch size	32
Convolution size 1	64
Convolution size 2	64
α	0.001
β	0.002
γ	0.99
τ	0.001
z	5

In order to demonstrate the effectiveness of our proposed approach to the excavating control of a CSD, we compare the performance of a well-trained human operator and the classic DDPG learning algorithm with our proposed approach. As shown in Fig. 13, the initial states for all the approaches are the same, and the state prediction model used in the classic DDPG is the same as that used in our learning approach. From the initial states, we also add random noise of state transitions so as to simulate the environmental disturbances. In dredging process, since the objective is to maintain the slurry density in the pipeline at a high level, and also ensure that all the safety concerns cannot been violated. Thus, we have depicted the curves of the slurry concentration in the pipeline, as well as the other variables related to safety concerns.

We can see that all the approaches seek to improve the slurry concentration from the initial states, and, in general, our proposed approach can quickly maintain the concentration above 60% in comparison with other two approaches. We can say that our approach can stabilize the slurry density, and, at the same time, it can also maintain all the safety concerns at the reasonable levels. In contrast, we can find that, the motor current of the underwater pump of the classic DDPG exceeds the maximum permitted value (180 A) between 147 s and 162 s. In the classic DDPG, the motor current of the cutter head also exceeds the safety threshold at 406 s to 412 s. Those cases also occurs in human operations occasionally. Thus, we can conclude that our proposed learning approach can provide quicker responses to the dynamics of the excavating process. Most importantly, all the safety concerns have been taken into consideration during stabilizing the slurry concentration of a CSD.

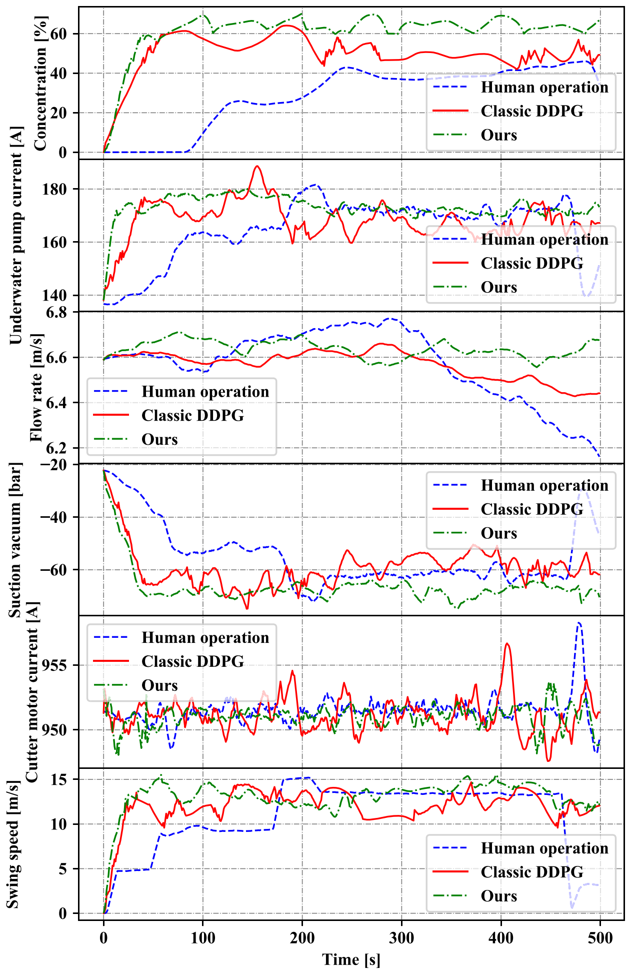


Fig. 13. Performance comparison between human expert operation, classic DDPG and our learning approach.

6. Conclusions

In order to propose an automated control approach for the excavating operation of a CSD, we have presented a data-driven offline learning approach, named Preprocessing-Prediction-Learning Control (PPLC). Specifically, the preprocessing module is responsible for filtering out irrelevant variables from raw collected data through correlation analysis and dimensionality reduction. We have also constructed a Convolutional Neural Network (CNN) model that works as the state transition function to resolve the problem of the lack of an accurate simulator. We employ the TD3 algorithm to design a deep reinforcement learning agent to control the swing speed of the excavating operation. In order to demonstrate the effectiveness of the proposed approach, we have compared the performance of a well-trained human operator and the classic DDPG algorithm with our approach. The results show that the proposed approach surpasses the others, with respect to the maintenance of slurry density at a high level and the satisfaction of safety concerns. Our approach also opens up the possibility of developing a learning approach to other complex control problems, where it is unaffordable to train the agent in real scenarios from scratch, but offline historical data are available to adopt our proposed approach.

CRediT authorship contribution statement

Changyun Wei: Conceptualization, Methodology, Formal analysis, Writing – original draft. **Hao Wang:** Visualization, Data curation. **Haonan Bai:** Software, Visualization, Data curation, Resources. **Ze Ji:** Writing – review & editing, Validation, Formal analysis. **Zenghui Liu:** Investigation, Methodology.

Declaration of competing interest

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.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61703138, and the International Science and Technology Cooperation Research Funds of Changzhou under Grant CZ20210027.

References

Anderlini, E., Husain, S., Parker, G.G., Abusara, M., Thomas, G., 2020. Towards real-time reinforcement learning control of a wave energy converter. J. Mar. Sci. Eng. 8 (11), 845.

Bai, S., Li, M., Kong, R., Han, S., Li, H., Qin, L., 2019. Data mining approach to construction productivity prediction for cutter suction dredgers. Autom. Constr. 105, 102833.

Dorrity, M.W., Saunders, L.M., Queitsch, C., Fields, S., Trapnell, C., 2020. Dimensionality reduction by UMAP to visualize physical and genetic interactions. Nature Commun. 11 (1), 1–6.

Evangelidis, G.D., Horaud, R., 2017. Joint alignment of multiple point sets with batch and incremental expectation-maximization. IEEE Trans. Pattern Anal. Mach. Intell. 40 (6), 1397–1410.

Fujimoto, S., Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. PMLR, pp. 1587–1596.

García-Gil, D., Luque-Sánchez, F., Luengo, J., García, S., Herrera, F., 2019. From big to smart data: Iterative ensemble filter for noise filtering in big data classification. Int. J. Intell. Syst. 34 (12), 3260–3274.

Han, S., Li, H., Li, M., Tian, H., Qin, L., Yu, Y., Ma, J., 2022. Intelligent short-term forecasting for mud concentration in CSD dredging construction. Ocean Eng. 266, 113151.

Kuhnle, A., Kaiser, J.-P., Theiß, F., Stricker, N., Lanza, G., 2021. Designing an adaptive production control system using reinforcement learning. J. Intell. Manuf. 32, 855–876.

Li, M., Kong, R., Han, S., Tian, G., Qin, L., 2018. Novel method of construction-efficiency evaluation of cutter suction dredger based on real-time monitoring data. J. Waterw. Port Coast. Ocean Eng. 144 (6), 05018007.

Li, S., Li, W., Wen, S., Shi, K., Yang, Y., Zhou, P., Huang, T., 2021. Auto-FERNet: A facial expression recognition network with architecture search. IEEE Trans. Netw. Sci. Eng. 8 (3), 2213–2222.

Li, W., Wen, S., Shi, K., Yang, Y., Huang, T., 2022. Neural architecture search with a lightweight transformer for text-to-image synthesis. IEEE Trans. Netw. Sci. Eng. 9 (3), 1567–1576.

Li, Y., Xiong, B., Vilathgamuwa, D.M., Wei, Z., Xie, C., Zou, C., 2020. Constrained ensemble Kalman filter for distributed electrochemical state estimation of lithium-ion batteries. IEEE Trans. Ind. Inform. 17 (1), 240–250.

Lin, C.-C., Deng, D.-J., Chih, Y.-L., Chiu, H.-T., 2019. Smart manufacturing scheduling with edge computing using multiclass deep q network. IEEE Trans. Ind. Inform. 15 (7), 4276–4284.

Liu, C., Chu, X., Wu, W., Li, S., He, Z., Zheng, M., Zhou, H., Li, Z., 2022. Human–machine cooperation research for navigation of maritime autonomous surface ships: A review and consideration. Ocean Eng. 246, 110555.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.

Reshef, D.N., Reshef, Y.A., Finucane, H.K., Grossman, S.R., McVean, G., Turnbaugh, P.J., Lander, E.S., Mitzenmacher, M., Sabeti, P.C., 2011. Detecting novel associations in large data sets. Science 334 (6062), 1518–1524.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al., 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science 362 (6419), 1140–1144.

Su, J., Huang, J., Adams, S., Chang, Q., Beling, P.A., 2022. Deep multi-agent reinforcement learning for multi-level preventive maintenance in manufacturing systems. Expert Syst. Appl. 192, 116323.

Tang, J.-Z., Wang, Q.-F., 2008. Online fault diagnosis and prevention expert system for dredgers. Expert Syst. Appl. 34 (1), 511–521.

Tang, J.-Z., Wang, Q.-F., Bi, Z.-Y., 2008. Expert system for operation optimization and control of cutter suction dredger. Expert Syst. Appl. 34 (3), 2180–2192.

Tang, J., Wang, Q., Zhong, T., 2009. Automatic monitoring and control of cutter suction dredger. Autom. Constr. 18 (2), 194–203.

Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A., 2008. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning. pp. 1096–1103.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., Bottou, L., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. 11 (12).

Wang, B., Fan, S., Jiang, P., Xing, T., Fang, Z., Wen, Q., 2020. Research on predicting the productivity of cutter suction dredgers based on data mining with model stacked generalization. Ocean Eng. 217, 108001.

Wei, C., Bai, H., Wei, Y., Ji, Z., Liu, Z., 2022. Learning manipulation skills with demonstrations for the swing process control of dredgers. Ocean Eng. 246 110545.

Wu, X., Zhu, X., Wu, G.-Q., Ding, W., 2013. Data mining with big data. IEEE Trans. Knowl. Data Eng. 26 (1), 97–107.

Xu, J., Hou, Z., Wang, W., Xu, B., Zhang, K., Chen, K., 2018. Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks. IEEE Trans. Ind. Inform. 15 (3), 1658–1667.

Yue, P., Zhong, D., Miao, Z., Yu, J., 2015. Prediction of dredging productivity using a rock and soil classification model. J. Waterw. Port Coast. Ocean Eng. 141 (4), 06015001.

Zhang, S., Li, X., Zong, M., Zhu, X., Wang, R., 2017. Efficient kNN classification with different numbers of nearest neighbors. IEEE Trans. Neural Netw. Learn. Syst. 29 (5), 1774–1785.

Zhang, J., Yu, J., Tao, D., 2018. Local deep-feature alignment for unsupervised dimension reduction. IEEE Trans. Image Process. 27 (5), 2420–2432.

Zhou, S.K., Le, H.N., Luu, K., Nguyen, H.V., Ayache, N., 2021. Deep reinforcement learning in medical imaging: A literature review. Med. Image Anal. 73, 102193.