Research on Watershed Runoff Forecast Based on Deep Reinforcement Learning

Xuhong Fang[1], Jiaye Li[2*], Qunfeng Liu[1], Hongguan Chen[2] and Zihao Luo[2]

[1] School of Computer Science and Technology, Dongguan University of Technology, Dongguan, Guangdong, 523808, China
[2] School of Environment and Civil Engineering, Dongguan University of Technology, Dongguan, Guangdong, 523808, China

*Corresponding author's e-mail: lijiaye@dgut.edu.cn

Abstract. With the drastic changes in global climate, the frequent occurrence of extreme weather events has led to an increasing number of flood events, highlighting the crucial importance of accurate watershed runoff forecasting for disaster prevention, water resource management, and environmental protection. However, conventional machine learning methods have often struggled to establish precise models when facing complex hydrological environments, resulting in significant forecast deviations. In contrast, deep reinforcement learning methods have shown remarkable performance in handling complex problems and achieved substantial success in various domains. In this context, our study takes the pioneering step of applying deep reinforcement learning methods to watershed runoff forecasting, aiming to enhance forecast accuracy and reliability to address the challenges posed by global climate change and frequent floods. We conducted comparative experiments on commonly used machine learning methods, namely Long Short-Term Memory (LSTM) and Deep Q-Network (DQN), for watershed runoff forecasting. The experimental results demonstrate that DQN, with a forecast accuracy of approximately 91.3%, outperforms LSTM significantly in terms of forecast accuracy and reliability. Even when DQN encounters forecast errors, the deviation does not exceed one runoff level.

Keywords: Watershed runoff forecast, Deep reinforcement learning, Deep Q-Network (DQN), Long Short-Term Memory (LSTM)

1 Introduction

In the context of ongoing global climate change and the frequent occurrence of extreme weather events and floods worldwide, watershed runoff forecasting plays a crucial role in flood control and disaster reduction, water resources management, and ecological conservation [1]. Accurate forecast of watershed runoff is of paramount importance for timely response to floods and waterlogging disasters, rational utilization of water resources, and maintenance of ecological balance [2].

In the past, many methods have been employed for runoff forecasting, encompassing statistical methods, physical models, and neural network models [3]. While these methods can provide predictions for watershed runoff to a certain extent, practical applications reveal some challenges and shortcomings [4]. These methods exhibit a notable reliance on historical observation data. However, inadequate historical data or poor data quality can adversely impact their prediction performance. Moreover, when confronted with intricate hydrological environments, terrains, and soil types, these methods often encounter challenges in establishing precise models [5], leading to significant deviations in prediction results. Simultaneously, the determination of parameters necessitates expertise and professional knowledge, making the selection and adjustment of parameters challenging for complex watershed systems.

In recent years, machine learning and deep learning have demonstrated remarkable capabilities in approximating nonlinear systems and handling high-dimensional data [6], [7], and in hydrological forecasting, it has also achieved many remarkable achievements. For example, Xu et al. propose an LSTM and PSO-based deep learning model, using PSO to optimize LSTM hyperparameters to improve learning sequence features, to predict flooding in specific watersheds from rainfall and runoff data[8]. Xiang et al. propose the Graph Neural Rainfall-Runoff Model, a deep learning model fully utilizing spatial data that, compared to baselines, has less overfitting and significantly improves performance[9]. Moreover, deep reinforcement learning (RL), an emerging artificial intelligence technology, has exhibited superior abilities in addressing complex problems and optimizing decision-making when compared to conventional machine learning methods [10]. As a result, it has achieved considerable success across diverse domains, including robotics [11], the Internet of Things [12], gaming [13], and autonomous driving [14]. Despite the superior capabilities of RL, to the best of our knowledge, there have been no reports on the utilization of RL methods for runoff forecasting.

This paper introduces a pioneering approach for watershed runoff forecasting, utilizing deep reinforcement learning methods [15], to forecast future runoff from historical precipitation, runoff, and other data [16]. In this study, we utilized the Deep Q-Network (DQN) in the domain of deep reinforcement learning and made specific modifications to adapt it to our research. Firstly, we replaced the traditional Q network with an LSTM neural network [17]. Secondly, our DQN architecture comprises two neural networks with identical structures: one serves as the Q network, and the other as the Q-Target network. Subsequently, we conducted performance experiments for both LSTM and DQN, comparing accuracy and loss during the training process, as well as accuracy on the test dataset. The test results demonstrate a significant improvement in accuracy for DQN compared to LSTM.Formatting the title, authors and affiliations

2 Method for watershed runoff forecast

2.1 Deep Q-Learning network model

DQN is a Q-learning algorithm based on deep neural networks and is also a form of value function approximation. In the DQN algorithm, the agent does not directly derive a comprehensive behavioral strategy from historical experience data. Rather, it learns from the reward values conveyed by the environment. Q-learning algorithm approximates the value function for state-action pairs (Q function) using deep neural networks. The Q function estimates the cumulative reward obtained by executing a certain action in the current state.

In the context of DQN, there exist two networks with identical structures: one is referred to as the Q-Target network, while the other is denoted as the Q network. In this study, we employed LSTM networks to approximate the Q function, using them as both the Q-Target and Q networks. LSTM networks offer significant advantages for handling time series data due to their unique memory cell structure, which effectively captures long-term dependencies in temporal data. As illustrated in figure 1, we utilized a 3-layer LSTM network, with each LSTM layer containing 64 neurons. The role of this architecture is to process sequential data and learn complex patterns within it. Following the output from the LSTM network, we utilized a fully connected layer to output the Q-value function Q(s, a), thereby accomplishing the forecast of runoff in the watershed.

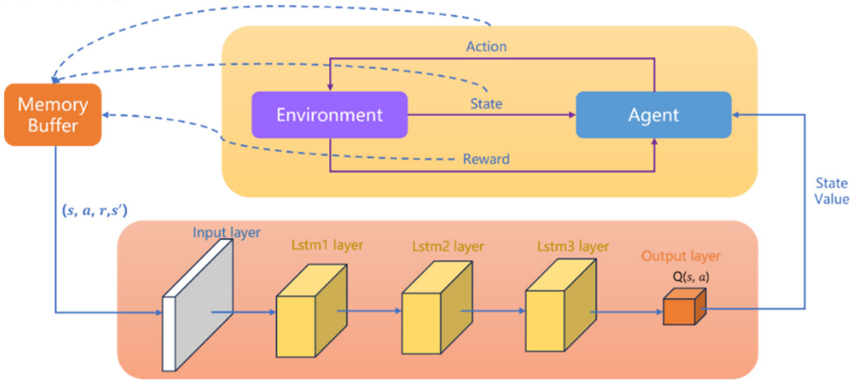

Fig. 1. The framework flow chart of DQN

Due to the high correlation between the previous state and the current state during the iterative interaction between the DQN agent and the environment, a memory buffer, as illustrated in figure 1, is introduced to store training samples over time. This buffer serves to break temporal correlations and smooth out changes in the data distribution, preventing the neural network from overfitting and failing to converge. During each learning iteration, DQN randomly selects a batch of samples from the memory buffer, inputs them into the neural network, and conducts gradient descent. As new training samples are generated, a mixture of old and new samples is updated in batches, disrupting the correlation between adjacent training samples and enhancing sample utilization.

2.2 Model training and updating

At the beginning of the training process, the agent utilizes perceivers to collect the current environment's state information and employs the ε-greedy algorithm to select the action to be executed. The ε-greedy algorithm is a strategy used to balance exploration and exploitation. Specifically, when the agent makes action selections, the ε-greedy algorithm randomly selects an action with a probability of ε ($0 < \varepsilon < 1$) for exploration and selects the action with the highest known Q-value with a probability of (1-ε) for exploitation [18]. By adjusting the value of ε, the agent can strike a balance between exploring unknown actions and exploiting known optimal actions, enabling it to better explore the environment and gradually optimize its decision-making strategy.

In the DQN algorithm, the Q network takes states as input and outputs Q-values for various actions. The more accurate these Q-values are, the better the Q network is trained. The Q-Target network is solely used for calculating the Target Q values, while the current Q values are predicted solely by the current Q network. The difference between the Q network and the Q-Target network is that the Q network is updated in the experience replay buffer with each step, while the Q-Target network periodically performs a hard copy (a complete replication) of the Q network's parameters to update itself. This delayed update is designed to ensure stability during the training of the Q network. Consequently, the agent can converge faster to superior strategies during the learning process. The loss function formula is computed by formula (1).

$$Loss = (r + \gamma \cdot maxQ_{a'}(s', a'; \omega^{-})) - Q(s, a{:}\, \omega)^2 \tag{1}$$

During the training phase, firstly, the current state s is input into the Q network, resulting in the Q value denoted as $Q(s, a; \omega)$, where a and ω denote the current action and network parameters, respectively. Subsequently, the next state s', is input into the Q-Target network, yielding Q values for various actions, denoted as $Q_{a'}(s', a'; \omega^{-})$. The action with the maximum Q value is then selected. Finally, $Q(s, a; \omega)$ serves as the network's prediction, while $r + \gamma \cdot maxQ_{a'}(s', a'; \omega^{-})$ serves as the actual value for the network. The loss function is chosen as the variance for error backpropagation. In this study, the parameter γ ($0 \leq \gamma \leq 1$) was set to 0.3, representing the discount factor. It balances immediate and future rewards. Lower values of γ prioritize short-term gains, while higher values emphasize long-term rewards, allowing the agent to optimize its strategy.

$$\omega_{t+1} = \omega_t + \alpha \cdot Loss\, \nabla\, Q(s, a; \omega) \tag{2}$$

By solving for the parameter ω, we can obtain the update formula for the Q network, as shown in formula (2). The parameter α represents the learning rate, a critical factor controlling how large each parameter update is during training. After each round of training, the parameter ω of the Q network is updated in real-time, while the parameter ω^{-} of the Q-Target network is updated with a delay. In our study, after several rounds of training, all parameters of the Q network are fully transferred to the Q-Target network.

3 Experiments and analysis of results

3.1 Experimental platform and dataset

In this paper, the experimental setup used a Lenovo workstation with the following specifications: OS - Ubuntu 20.04.6 LTS, CPU - Intel(R) Xeon(R) Gold 6128 CPU @ 3.40GHz, RAM - 128GB, GPU - RTX 2080Ti, and Deep Reinforcement Learning Framework - PyTorch.

The experimental dataset, CAMELS (Catchment Attributes and Meteorology for Large-sample Studies) [19], was sourced from multiple American institutions and research labs, providing high-quality, multi-spatial scale daily forcing data, daily runoff data, and fundamental metadata for 671 watersheds, including location, elevation, size, and shapefiles delineating watershed boundaries.

For our study, we utilized the Daymet data from the CAMELS dataset, focusing on the Fish River near Fort Kent, Maine watershed, at approximately 68.58° W longitude and 47.24° N latitude, covering 2252.70 km^2. This dataset offers daily 1 km x 1 km gridded coverage data for the entire continental United States, encompassing seven surface parameters: precipitation, shortwave radiation, vapor pressure, minimum temperature, maximum temperature, day length, and runoff.

Our training dataset spanned from January 1, 2011, to December 31, 2013, aligning with common practice in runoff forecasting, which often uses recent years' data for training. Since daily runoff data was unavailable or replaced with predicted data after September 30, 2014, the test dataset covered the period from January 1, 2014, to June 30, 2014.

3.2 Analysis of experimental results

In our DQN model, the input state consists of continuous data from the past 7 days, with a size of (7, 7). The model's output represents the average runoff level of the entire watershed for the next day, making the action space discrete with a size of (1, 5), where each item corresponds to the value of a runoff level category. The reward function is set to 1 for correctly forecasting the next day's runoff level category, and 0 otherwise. During the DQN model initialization phase, multiple parameters were configured to construct the model. These parameters include the maximum training iterations, maximum capacity of the experience replay buffer, learning rate, difference in training steps for target network updates, and the probability factor for employing the ε-greedy algorithm. The detailed model initialization parameters are presented in Table 1.

Table 1. DQN model initialization parameter table.

Parameter	Value	Annotation
TOTAL_STEP	100	Maximum training iterations
MEMORY_SIZE	20	Maximum capacity of the experience replay buffer
REPLACE_TARGET_ITER	10	The number of training rounds required for an update of the target network
DISCOUNT_FACTOR	0.3	The discount factor
LEARNING_RATE	0.001	Learning rate
E_GREEDY	0.9	The probability factor for the ε-greedy algorithm.

In this study, the DQN model employs a cyclic iterative training approach, where the neural network incrementally optimizes its parameters through repeated agent-environment interactions. By cycling through episodes of taking actions, observing results, and updating parameters, the network is able to continuously learn and adapt to the complex variation patterns of watershed runoff over time. This iterative reinforcement learning method enhances the model's predictive capabilities, enabling it to better capture intricate hydrological dynamics and runoff generation processes.

Additionally, we conducted comparative experiments using LSTM networks for direct runoff forecasting in the research watershed. The LSTM architecture and initialization parameters were designed to match the Q-network within the DQN model, including identical network structure, maximum training iterations, and learning rate. This ensures a fair comparison between the capabilities of the models. The training iteration process for the standalone LSTM model is illustrated in figure 2, while the training iteration process for the full DQN model is depicted in figure 3.

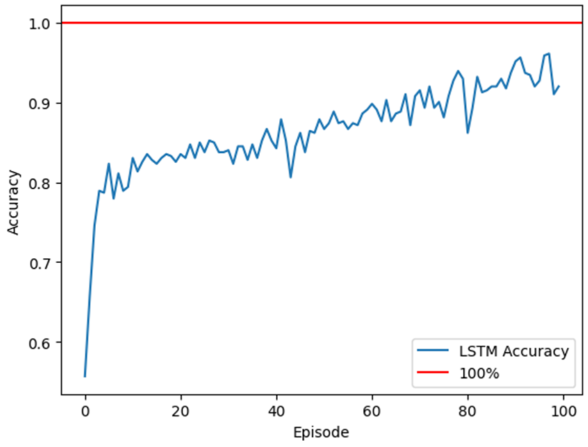

Fig. 2. LSTM training process

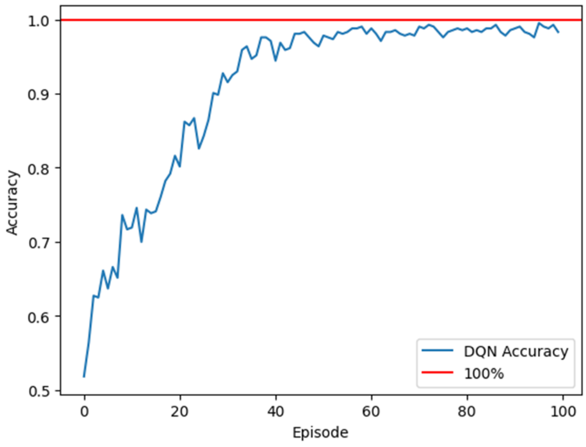

Fig. 3. DQN training process.

From the results in Figure 2 and Figure 3, it is evident that DQN outperforms LSTM during the training process, achieving higher accuracy with the same number of training iterations. Furthermore, DQN achieves convergence with fewer training iterations compared to LSTM. This observation is further supported by the loss curves shown in figure 4, where the blue line represents LSTM and the orange line represents DQN. Throughout the training process, the descending slopes of the loss curves for both LSTM and DQN are similar, but DQN exhibits smaller loss fluctuations and demonstrates a more stable behavior, consistently moving towards reducing the loss. In contrast, LSTM experiences more significant fluctuations in the later stages of the training iterations.

During the testing phase, we implemented a sliding window approach to enhance our daily runoff predictions. Within each interval, we inputted data from the preceding 7 days into both the LSTM and DQN models to make forecasts regarding the runoff level for the subsequent day. This systematic approach enabled us to forecast daily runoff for the first half of 2014. We achieved this by incrementally advancing the 7-day input window one day at a time, which allowed us to continuously generate predictions. The forecast results on the test set are shown in figure 5 and figure 6. The Y-axis shows the model's predicted runoff level category from 0 to 4 for the next day, based on quantiles of the historical maximum daily runoff. The X-axis shows the progression of test days in the first half of 2014. Each prediction uses the prior 7 days as input to forecast the next day's runoff level.

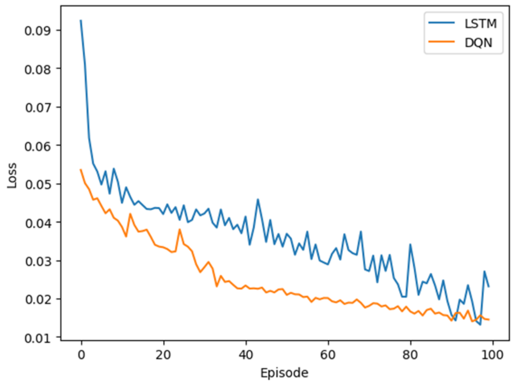

Fig. 4. Loss curves of LSTM and DQN during training iterations.

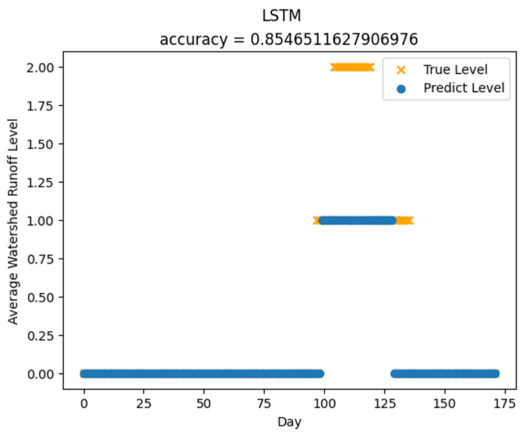

Fig. 5. Results of the LSTM on the test dataset.

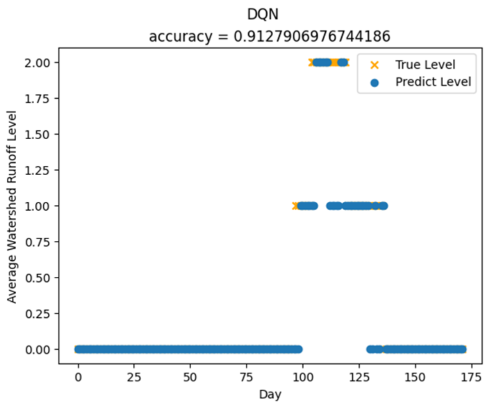

Fig. 6. Results of the DQN on the test dataset.

The experimental results indicate that the DQN model achieved approximately 91.3% accuracy on the test dataset, while the LSTM model had about 85.5% accuracy. This demonstrates DQN significantly outperformed LSTM, with a 6.8% accuracy improvement in runoff level forecasting. Analyzing the errors reveals insights. Most of DQN's errors occurred during transitions between runoff level categories but remained within one category of deviation. In contrast, LSTM had lower accuracy in stable and transitional periods with larger errors. Compared to DQN, LSTM struggled more to capture runoff level patterns, trends, and transitions.

4 Conclusion

In this study, we applied the DQN deep reinforcement learning method to watershed runoff forecasting, utilizing an LSTM network within the DQN architecture. Comparative experiments between LSTM and DQN models revealed that DQN achieved convergence with fewer training iterations, displayed smaller and more stable errors during training, and outperformed LSTM with a significantly higher test accuracy of 91.3% compared to 85.5%. Analysis of DQN's errors indicated they primarily occurred during runoff level transitions, with deviations staying within a single level. These results highlight the potential benefits of employing deep reinforcement learning techniques like DQN for runoff forecasting, suggesting their promise in advancing hydrological model and prediction in watershed contexts.

References

1. Abbass K, Qasim M Z, Song H, et al. (2022) A review of the global climate change impacts, adaptation, and sustainable mitigation measures. Environmental Science and Pollution Research, 29(28): 42539-42559.
2. Nkwunonwo U C, Whitworth M, Baily B. (2020) A review of the current status of flood modelling for urban flood risk management in the developing countries. Scientific African, 7: e00269.
3. Javadinejad S, Dara R, Jafary F. (2022) Difference of rainfall-runoff models and effect on flood forecasting: A brief review. Resources Environment and Information Engineering, 4(1): 184-199.
4. Peel M C, McMahon T A. (2020) Historical development of rainfall-runoff modeling. Wiley Interdisciplinary Reviews: Water, 7(5): e1471.
5. Molina J L, Zazo S, Martín-Casado A M, et al. (2020) Rivers' temporal sustainability through the evaluation of predictive runoff methods. Sustainability, 12(5): 1720.
6. Mohammadi B. (2021) A review on the applications of machine learning for runoff modeling. Sustainable Water Resources Management, 7(6): 98.
7. Bishop C M, Nasrabadi N M. (2006) Pattern recognition and machine learning. New York: Springer.
8. Xu Y, Hu C, Wu Q, et al. (2022) Research on particle swarm optimization in LSTM neural networks for rainfall-runoff simulation. Journal of hydrology, 608: 127553.
9. Xiang Z, Demir I. (2021) High-resolution rainfall-runoff modeling using graph neural network. arXiv preprint arXiv:2110.10833.

10. Li Y. (2017) Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.
11. Brunke L, Greeff M, Hall A W, et al. (2022) Safe learning in robotics: From learning-based control to safe reinforcement learning. Annual Review of Control, Robotics, and Autonomous Systems, 5: 411-444.
12. Chen W, Qiu X, Cai T, et al. (2021) Deep reinforcement learning for Internet of Things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 23(3): 1659-1692.
13. Vinyals O, Babuschkin I, Czarnecki W M, et al. (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782): 350-354.
14. Kiran B R, Sobh I, Talpaert V, et al. (2021) Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems, 23(6): 4909-4926.
15. Arulkumaran K, Deisenroth M P, Brundage M, et al. (2017) A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866.
16. Gao S, Zhang S, Huang Y, et al. (2022) A new seq2seq architecture for hourly runoff prediction using historical rainfall and runoff as input. Journal of Hydrology, 612: 128099.
17. Yu Y, Si X, Hu C, et al. (2019) A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31(7): 1235-1270.
18. Rodrigues Gomes E, Kowalczyk R. (2009) Dynamic analysis of multiagent Q-learning with ε-greedy exploration. In: Proceedings of the 26th annual international conference on machine learning. 369-376.
19. Livneh B, Rosenberg E A, Lin C, et al. (2013) A long-term hydrologically based dataset of land surface fluxes and states for the conterminous United States: Update and extensions. Journal of Climate, 26(23): 9384-9392.