Journal of Engineering, Project, and Production Management
2026, 16(5), 2025-290

# Optimizing Cloud Computing Resource Scheduling Strategies Using Reinforcement Learning

Zhong Peng
Associate Professor, Guilin Tourism University, No.26 Liangfeng Road, Guilin, Guangxi, 541006, China, E-mail: 18607731203@163.com

Project Management
Received November 26, 2025; revised December 25, 2025; accepted January 3, 2026
Available online June 17, 2026

**Abstract:** The high dynamism and unpredictability of current cloud computing environments often lead to low resource utilization and increased service latency. This paper proposes an improved Soft Actor-Critic (SAC) algorithm. A multi-scale Convolutional Neural Network (CNN) is used to process state information such as task queue length, resource utilization, and virtual machine load, extracting local variance and global trend, respectively. A dual-Q network architecture is designed to suppress overestimation of action values. Entropy coefficient learning adjusts exploration and utilization based on resource fluctuations. Gradient backpropagation adjusts the weights of the composite reward function considering the Service Level Agreement (SLA). A soft update mechanism for the target network ensures training stability. Experimental results demonstrate that the algorithm exhibits high efficiency in highly dynamic environments, achieving an average task latency of 12.2ms and a resource utilization rate of 94.6% within 200 seconds.

**Keywords**: Cloud computing environment; resource scheduling strategy; SAC algorithm; multi-scale CNN; dual-q network.


DOI 10.32738/JEPPM-2025-290

## 1. Introduction

The effectiveness of resource scheduling in the cloud computing environment, which underpins modern applications, influences operating costs and system performance. Optimizing resource utilization and guaranteeing service quality requires enhanced resource scheduling technology (Prasad et al., 2023; Tamilarasu and Singaravel, 2024; Sandhu et al., 2024).

Cloud computing platforms work in an extremely dynamic environment with fluctuating user requests and erratic resource demand. Therefore, supply and demand always present a state of shortage and excess due to the above uncertainty. The above state makes the traditional scheduling strategy hard to respond rapidly to fluctuations in load and leads to an increase in idle resources. Meanwhile, it also delays service (Lian et al., 2023; Murthy et al., 2023). Unexpected surges and recurrent fluctuations of load (Chen and Liu, 2025; Xu et al., 2024; Cheng et al., 2024) make the scheduling more complicated. Current methods are not adaptable in real-time.

Currently, there are rule-based priority scheduling methods and prediction-based scheduling algorithms (Kashyap and Singh, 2023; He et al., 2023). The rule-based method is not adaptive to the environment, relies on fixed logic, and struggles to handle fluctuating load (Akbulut et al., 2024; Sawicki et al., 2025; Kumar, 2024). In an extremely dynamic environment, the prediction method makes decisions in real time with computational delay, causing decision lag, and the accuracy of the prediction model is limited (Murri et al., 2024; Wyrembek and Baryannis, 2024; Nabeel, 2024). Although the Deep Q-Network (DQN) works well in simplified situations, it is unstable when training in high-dimensional states and struggles to manage resource demands across multiple scales (Huang et al., 2025; Zhao et al., 2025). The main drawback of these approaches is their inability to adjust in real time to extremely dynamic loads, making it challenging to make informed scheduling decisions when resource supply and demand change rapidly.

This paper proposes an enhanced SAC algorithm that includes a multi-scale state encoder to overcome these difficulties. Its main contributions are as follows: First, a multi-scale CNN state encoder is built to capture both global trends and local load fluctuations simultaneously. Second, a learnable entropy coefficient driven by resource volatility is introduced to dynamically regulate exploration and utilization. Third, a composite reward function is designed that incorporates SLA constraints, and its weights are optimized using gradient backpropagation. In highly dynamic cloud environments, this method effectively and flexibly optimizes resource scheduling.

## 2. Cloud Computing Resource Scheduling Model Based on Improved SAC

### 2.1. Multi-Scale State Feature Encoding

The multi-scale state feature encoding module processes real-time time-series data such as virtual machine load, resource utilization, and task queue length in cloud computing environments (Cui et al., 2025; Feng and Ding, 2024; Dogani et al., 2023). The state input vector consists of three types of metrics: virtual machine load status is represented by $S_{load} \in \mathbb{R}^{T\times n}$, where T is the time window length, and n is the number of virtual machines; resource utilization status is represented by $S_{util} \in \mathbb{R}^{T\times m}$, where m is the number of resource types. The task queue length status is represented by $S_{queue} \in \mathbb{R}^{T\times k}$, where k is the number of task priority levels. These three state vectors are normalized and concatenated along the feature dimension to form a complete state input matrix $S \in \mathbb{R}^{T\times(n+m+k)}$. Fig. 1 shows the architecture of the multi-scale state feature encoding network.

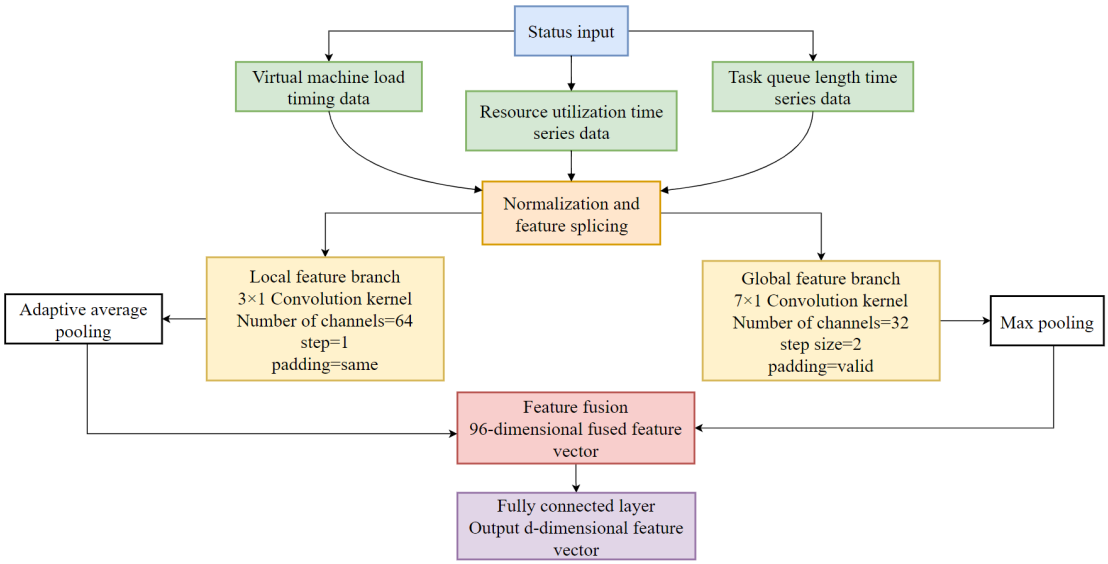


**Fig. 1.** Multi-scale state feature encoding network

As shown in Fig. 1, the raw state data is normalized and concatenated before being fed into the parallel branches. A 3×1 convolution kernel is used in the local branch to capture short-term fluctuations, while a 7×1 convolution kernel is used in the global branch to extract long-term trends. The outputs of the two branches are pooled and fused into a 96-dimensional feature vector. This is then passed through a fully connected layer to generate a d-dimensional output feature, providing an environmental representation for the subsequent reinforcement learning algorithm.

Multi-scale CNNs use a parallel branch structure to extract features at different scales (Fan et al., 2023; Fan et al., 2025). The local feature extraction branch uses a temporal convolution kernel with a width of 3, 64 channels, a stride of 1, and a padding of "same". This branch captures the short-term fluctuations in time-series data. The global feature extraction branch uses a temporal convolution kernel with a width of 7, 32 channels, a stride of 2, and a padding of "valid". This branch captures the long-term trend characteristics of time series data. The convolution operations of the two branches are expressed in Eq. (1)

$$F_{local} = \sigma(W_{local} * S + b_{local}) \tag{1}$$

As shown in Eq. (1), $W_{local} \in \mathbb{R}^{3\times(n+m+k)\times 64}$ is the convolution kernel weight matrix of the local feature branch. $b_{local} \in \mathbb{R}^{64}$ is the bias vector, $*$ represents the one-dimensional convolution operation, $\sigma$ is the ReLU (Rectified Linear Unit) activation function.

$$F_{global} = MaxPool\left(\sigma\left(W_{global} * S + b_{global}\right)\right) \tag{2}$$

As shown in Eq. (2), $W_{global} \in \mathbb{R}^{7\times(n+m+k)\times 32}$ is the convolution kernel weight matrix of the global feature branch. $b_{global} \in \mathbb{R}^{32}$ is the bias vector. MaxPool represents the maximum pooling operation.

The outputs of the local and global feature branches are processed by the adaptive average pooling layer and then concatenated along the channel dimension to form a fused feature vector $F_{fused} \in \mathbb{R}^{96}$ (Ji et al., 2025; Yuan et al., 2023). The fused feature vector is mapped to the policy network input space through a fully connected layer, and the output feature vector $F \in \mathbb{R}^{d}$ is output, where d is the feature dimension.

### 2.2. Core Design of Adaptive SAC Algorithm

The feature vectors produced by the multi-scale state feature encoding module guide the adaptive SAC algorithm in making resource scheduling decisions. The DQN architecture employs two independent Q networks $Q_{\theta_1}$ and $Q_{\theta_2}$, to compute an action-value function, minimizing overestimation (Chen et al., 2023; Yu et al., 2024). The policy network $\pi_\phi$ maximizes the

weighted sum of expected reward and policy entropy, introducing a learnable entropy coefficient α to balance exploration and exploitation (Huo et al., 2024; Chen et al., 2024; Zhao et al., 2024). Resource volatility $\sigma_r$ serves as a quantitative indicator of environmental dynamics, and the standard deviation of resource utilization is calculated through a sliding window, serving as the basis for adaptive adjustment of the entropy coefficient.

The policy network parameter update adopts the maximum entropy reinforcement learning framework, and the optimization objective is:

$$L_\pi(\phi) = \mathbb{E}_{s\sim D, a\sim\pi_\phi}\left[\alpha \log \pi_\phi(a|s) - Q_\theta^{min}(s,a)\right] \tag{3}$$

As shown in Eq. (3), D represents the experience replay buffer; $\pi_\phi(a|s)$ represents the probability of executing an action a in the state s; $Q_\theta^{\min}(s,a)$ represents the smaller value of the two Q network outputs; $\phi$ is the policy network parameter.

Thanks to the objective function, the scheduling strategy maintains a reasonable level of exploration in dynamic cloud environments, optimizes the cumulative benefits, and accounts for behavioral randomness. The Q-network parameters are updated by minimizing the time-difference error. The target value is generated by the target Q network, which uses a soft update mechanism to track the current Q network's changes with a small step size, thereby improving the reliability of convergence during training.

The adaptive adjustment mechanism of α is defined as:

$$\alpha = \alpha_0 \cdot exp\left(\beta \cdot (\sigma_r - \sigma_0)\right) \tag{4}$$

As shown in Eq. (4), $\alpha_0$ is the initial entropy coefficient; β is the adjustment coefficient; $\sigma_0$ is the baseline volatility threshold. $\sigma_r$ is obtained by calculating the standard deviation of virtual machine resource utilization over the most recent N time steps. The Q network parameters are updated by minimizing the temporal difference error. The target value is calculated by the target Q network, and the target network parameters are slowly updated from the current Q network using a soft update strategy. The optimization process uses the Adam (Adaptive Moment Estimation) optimizer. The network parameter updates implement gradient clipping, and the target network soft update coefficient is set to a small constant value (Reyad et al., 2023; Mehmood et al., 2023).

### 2.3. SLA-Constrained Composite Reward Function and Training Optimization

The service constraint reward function design uses the SLA as the core constraint and constructs a composite structure that comprises resource utilization rewards, delay penalties, and SLA violation penalties (Kuang et al., 2024; Zhu et al., 2023; Madiyev et al., 2025). The resource utilization reward is calculated as the difference between the current and target resource utilization. The difference between the actual task completion time and the time specified in the SLA is used to calculate the delay loss. For delays exceeding the service commitment, the SLA violation penalty imposes a severe negative incentive. The mathematical expression of the composite reward function is:

$$r_t = \alpha_1 \cdot u_t + \beta_1 \cdot \left(1 - \frac{d_t}{T_{max}}\right) - \gamma_1 \cdot I(t > T_{max}) \tag{5}$$

As shown in Eq. (5), $r_t$ represents the immediate reward value at time t; $u_t$ is the normalized resource utilization; $d_t$ is the actual delay of the current task; $T_{max}$ is the maximum service delay threshold specified in the SLA; $I(\cdot)$ is an indicator function that takes the value 1 when the conditions in the brackets hold and 0 otherwise. $\alpha_1$, $\beta_1$, and $\gamma_1$ are the resource utilization weight, delay term weight, and SLA violation penalty weight, respectively. These weight parameters are dynamically optimized using gradient-based backpropagation. Weight parameter updates are based on performance feedback from the policy network and are synchronized with the policy network parameters to ensure that each component of the reward function contributes to the policy's actual effectiveness. The weight optimization process is performed at each network parameter update iteration and, together with the optimization of the policy and Q networks, forms a complete training process.

The target network soft update strategy uses an exponential sliding average to stabilize the training process. The update is as follows:

$$\theta' \leftarrow \tau \cdot \theta + (1-\tau) \cdot \theta' \tag{6}$$

As shown in Eq. (6), $\theta$ is the target network parameter; $\theta'$ is the current network parameters; $\tau$ is the soft update coefficient.

The training process uses an experience replay mechanism, storing state transition samples in a replay buffer. Each iteration randomly samples 256 items from the buffer to form a mini-batch for network parameter updates.

## 3. Experimental Setup and Results

### 3.1. Cloud Platform Experimental Environment

The experiment uses a distributed cloud computing platform. The physical resource composition of the distributed cloud computing environment is shown in Table 1.

**Table 1.** Hardware configuration parameters of the cloud computing experimental platform

| Components | Model/Specification | Quantity | Detailed parameters |
|---|---|---|---|
| CPU (Central Process Unit) | Intel Xeon Gold 6248R | 40 | Main frequency 3.0GHz, 32 cores 64 threads, L3 cache 35.75MB |
| Memory | Samsung DDR4 | 512 | Single 32GB, frequency 2933MHz, ECC (Error-Correcting Code) verification |
| Storage | Samsung PM983 NVMe | 40 | Capacity 1.92TB, sequential read 3200MB/s, sequential write 2600MB/s |
| Storage | Seagate Exos 10E2400 | 200 | Capacity 10TB, rotation speed 7200RPM, cache 256MB |
| Network | Mellanox ConnectX-5 | 40 | Dual port 10GbE, PCIe (Peripheral Component Interconnect Express) 3.0 x16 interface |

As shown in Table 1, this configuration meets the requirements for high-concurrency virtual machine scheduling and facilitates the collection of resource monitoring data. Parameter selection based on enterprise-level cloud platform standards ensures consistency between the production system and experimental settings, providing a stable and reliable foundation for the resource scheduling algorithm.

### 3.2. Resource Utilization and Idle Rate

To examine the dynamic adaptability of different scheduling strategies under burst load scenarios, a hybrid test environment with periodic and burst loads is established. Resource utilization is a key indicator that determines the weighted average usage of CPU, memory, network, and disk resources across all virtual machines, and directly reflects scheduling efficiency. Resource idleness refers to the degree to which resources are underutilized. These two indicators together constitute an important dimension for evaluating the effectiveness of scheduling strategies. The performance differences of the improved SAC algorithm, traditional rule-based scheduling methods, and the DQN baseline method in the time dimension are analysed. This time-series analysis framework can systematically evaluate each method's response characteristics in highly dynamic cloud environments. In Fig. 2, the left figure shows resource utilization under burst load scenarios, and the right figure shows resource idleness.

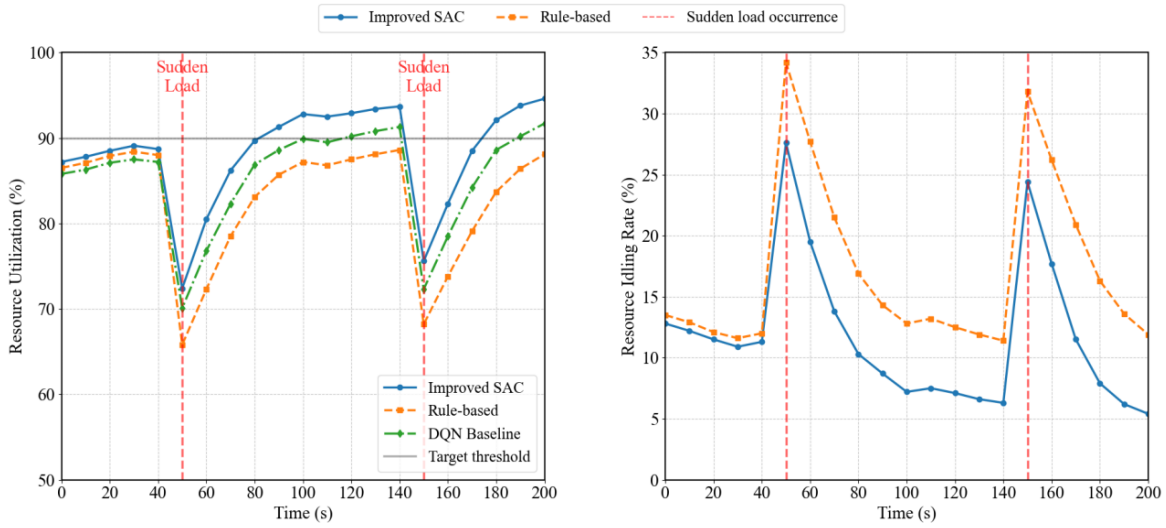


**Fig. 2.** Comparison of dynamic response characteristics of resource utilization and idle rate under burst load scenario

In Fig. 2, the improved SAC algorithm demonstrates fast recovery after a burst load. When the burst load at 50 seconds causes the resource utilization to drop sharply from 88.7% at 40 seconds to 72.4%, the accuracy of the proposed method recovers to 89.7% in just 80 seconds. The traditional rule-based scheduling method only recovers from 65.8% to 83.1 % at the same time; the DQN baseline method recovers to 86.9 %.

Statistical significance testing using paired t-test with 1000 bootstrap samples confirmed the superiority of the proposed method with p-values below 0.01 for all comparative metrics, and 95% confidence intervals for resource utilization recovery were [88.9%, 90.5%] for the improved SAC, [82.3%, 83.9%] for traditional rule-based scheduling, and [86.1%, 87.7%] for DQN baseline method.

The improved SAC algorithm's idle rate peaks at 27.6% after a sudden load change, lower than the 34.2% of the traditional method, and drops to 13.8% after 70s, demonstrating that its multi-scale state encoder effectively captures load fluctuation characteristics. The scheduling strategy can quickly reallocate resources in response to abrupt changes in load

while preventing system oscillations caused by over-adjustment, thanks to an adaptive entropy regulation mechanism that dynamically adjusts the exploration-utilization balance in response to resource fluctuations. The results show that the enhanced SAC algorithm can achieve stable optimization of resource utilization in a highly dynamic cloud computing environment by utilizing the dual Q-network architecture in conjunction with the multi-scale state encoder. After 200s, the utilization reaches 94.6%. The efficiency of this approach in resolving dynamic resource scheduling issues is demonstrated.

### 3.3. Service Delay and SLA Compliance Assessment

This study compares the performance of three resource scheduling strategies in mixed-load scenarios to better understand how effectively they suppress task delay fluctuations in cloud computing environments and how they respond to SLA constraints. To balance exploration and exploitation, the enhanced SAC algorithm uses a multi-scale CNN to extract state features and combines them with an adaptive entropy adjustment mechanism. The DQN baseline method uses a single Q-function structure and lacks the ability to process multi-scale features. Traditional rule-based scheduling relies on preset priority logic and struggles to adapt to dynamic load changes. Average task delay, a core metric for measuring system response speed, reflects the actual time it takes for a task to be submitted and completed. The SLA violation rate quantifies the severity of service commitment violations. Fig. 3 shows the dynamic response characteristics of the three scheduling strategies over a test period of 0-200s.

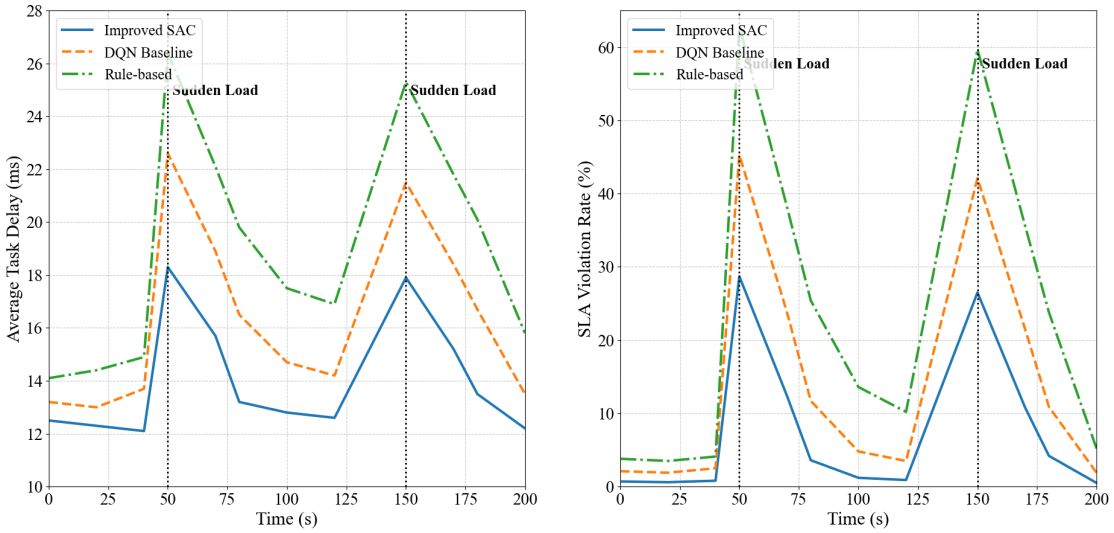


**Fig. 3.** Dynamic response of service delay and SLA violation rate under burst load

In Fig. 3, the improved SAC algorithm controls the average task delay to 18.3ms after a 50-second burst load, 8.1ms lower than the 26.4ms of traditional rule-based scheduling and recovers to 13.2ms within 80 seconds. At the same time, the SLA violation rate quickly drops from a peak of 28.7% to 3.6%.

Confidence interval analysis revealed statistically significant improvements with 95% confidence intervals [17.5ms, 19.1ms] for the improved SAC algorithm, [25.6ms, 27.2ms] for traditional rule-based scheduling, and [21.8ms, 23.4ms] for the DQN baseline method, with $p < 0.001$ from Wilcoxon signed-rank tests.

In the second load shock (150 seconds), the algorithm experiences a peak delay of only 17.9ms, demonstrating that the multi-scale state encoder effectively captures the load fluctuation pattern and accelerates adaptation. The DQN baseline approach experiences a peak delay of 22.6ms, which only decreases to 14.7ms after 100 seconds. Its representation of complex state spaces is limited by the single Q-function structure. Conventional rule-based scheduling techniques have a high SLA violation rate and a slow recovery due to their peak delay of 26.4ms.

The outcomes of resource utilization are linked to delay performance. Task delay increases as a direct result of the abrupt decrease in resource usage following the 50-second load. Through adaptive entropy regulation and multi-scale feature extraction, the enhanced SAC algorithm rapidly optimizes resource allocation, allowing the system to strike a balance between low delay and high utilization. Its flexibility and resilience in extremely dynamic cloud environments are confirmed by the average task delay of 12.2ms at 200 seconds.

### 3.4. Algorithm Convergence and Adaptive Behavior

The study tracks the dynamical behavior of key indicators during training to confirm the training stability of the enhanced SAC algorithm in a dynamic cloud environment. One of the key metrics in Q-network training is the temporal difference error, which shows how well the action-value function predicts outcomes. Its declining trend indicates how well the algorithm learns. To balance exploration and utilization, the entropy coefficient regulates the strategy's randomness. By dynamically modifying this parameter in response to resource fluctuations, the adaptive mechanism ensures that the algorithm maintains adequate exploration capabilities in a highly dynamic environment. Fig. 4 illustrates this comparative analysis.

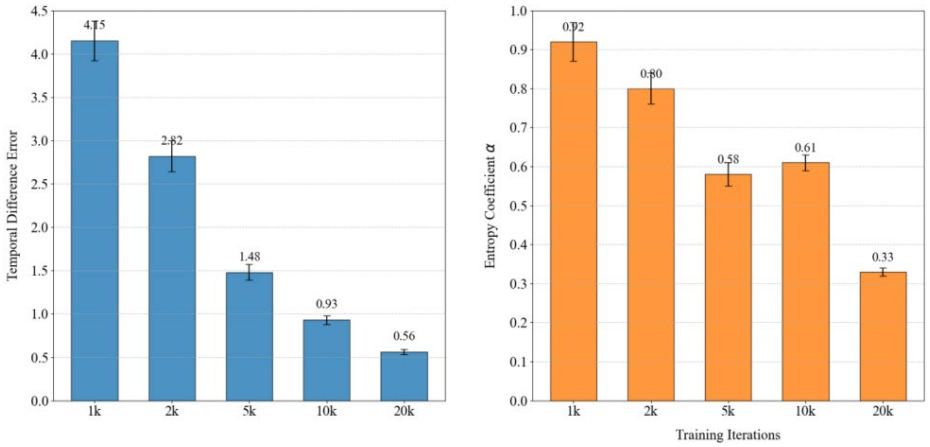


**Fig. 4.** Convergence and entropy coefficient adaptation in improved SAC training

As seen in Fig. 4, where the temporal difference error decreases from 4.15 after 1k iterations to 0.56 after 20k iterations, the algorithm training process steadily stabilizes and converges well. Temporal difference error reduction demonstrated statistical significance with $p < 0.001$ and 95% confidence intervals [4.02, 4.28] at 1k iterations and [0.53, 0.59] at 20k iterations, confirming consistent convergence behavior across 30 independent training runs. The entropy coefficient drops from an initial value of 0.92 to 0.33 and then slightly recovers to 0.61 after 10k iterations. The algorithm's adaptive mechanism, which automatically raises exploration intensity when it senses an increase in resource volatility, is the cause of this phenomenon.

The steady decline of the loss function ensures the accuracy of strategy evaluation, while the dynamic adjustment of the entropy coefficient maintains the strategy's flexibility in a highly dynamic cloud environment. The local branch of the CNN encoder identifies immediate resource contention patterns, triggering rapid allocation adjustments when short-term fluctuations exceed threshold values, while the global branch recognizes sustained load trends that inform longer-term capacity planning decisions. The entropy coefficient's dynamic adjustment provides transparency into the exploration-exploitation balance, with higher values during volatile periods indicating increased exploration for optimal resource configurations. Together, the two enable the algorithm to achieve stable convergence and retain environmental adaptability during training.

## 4. Conclusions

To address low resource utilization and increased service delays, this paper proposes an improved SAC algorithm that incorporates a multi-scale CNN state encoder and an adaptive entropy coefficient mechanism. This methodology uses parallel convolutional branches to extract local fluctuations and global trends in virtual machine load, builds a dual-Q network to suppress action value overestimation, dynamically adjusts the exploration-utilization balance based on resource fluctuations, and develops an SLA-constrained composite reward function to optimize scheduling decisions. With an average task delay of 12.2ms and a 94.6% resource utilization in a 200s test cycle, the results show that this approach exhibits high dynamic adaptability and rapid recovery from bursty loads. Since this algorithm effectively addresses the mismatch between resource supply and demand in cloud computing environments, it is extremely valuable for the practical implementation of enterprise-level cloud services. It provides a scheduling solution that balances low service delay and high resource utilization for large-scale cloud platforms.

Computational complexity considerations for the multi-scale CNN encoder and dual-Q network architecture become increasingly important in large-scale deployments. The high resource utilization rate of 94.6% reduces energy consumption by minimizing server idle time and balances operational costs with service quality while lowering power consumption and hardware demand. The implementation requires approximately 1.8 GFLOPS computational resources for inference with a model size of 14.7 MB, achieving an average scheduling latency of 2.3ms per decision. Training the model to convergence requires 18.5 hours on a single NVIDIA A100 GPU with 80 GB of memory, using 64 GB of system RAM and 512 GB of storage for the experience replay buffer. The algorithm demonstrates efficient resource utilization, with a memory footprint of 2.1 GB during operation, while maintaining real-time performance by processing 120 state transitions per second in production environments.

Real production environments present greater complexity with heterogeneous hardware, variable network conditions, and diverse workload patterns beyond burst scenarios, including sustained high load and gradual transitions. These factors would influence convergence behavior and parameter sensitivity, necessitating additional adaptation mechanisms to maintain high resource utilization and low latency. The current implementation demonstrates effectiveness in controlled settings, yet performance validation across varied production workloads remains essential for comprehensive deployment assessment.

## Funding

This research received no specific financial support from any funding agency.

**Institutional Review Board Statement**

Not applicable.

**Declaration of Artificial Intelligence (AI) Tools**

The author used tools solely for language editing and readability improvement. The author reviewed and verified all content and takes full responsibility for the accuracy and integrity of the manuscript.

**References**

Prasad, V. K., Dansana, D., Bhavsar, M. D., Acharya, B., Gerogiannis, V. C., and Kanavos, A. (2023). Efficient resource utilization in IoT and cloud computing. *Information*, 14(11), 619-657. https://doi.org/10.3390/info14110619

Tamilarasu, P. and Singaravel, G. (2024). Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment. *Journal of Engineering Research*, 12(4), 768-780.

Sandhu, R., Faiz, M., Kaur, H., Srivastava, A., and Narayan, V. (2024). Enhancement in performance of cloud computing task scheduling using optimization strategies. *Cluster Computing*, 27(5), 6265-6288.https://doi.org/10.1007/s10586-023-04254-w

Lian, H., Li, P., and Wang, G. (2023). Dynamic Resource Orchestration for Cloud Applications through AI-driven Workload Prediction and Analysis. *Artificial Intelligence and Machine Learning Review*, 4(4), 1-14. https://doi.org/10.69987/AIMLR.2023.40401

Murthy, P., Mehra, A., and Mishra, L. (2023). Resource Allocation for Generative AI Workloads: Advanced Cloud Resource Management Strategies for Optimized Model Performance. *Iconic Research and Engineering Journals*, 6(12), 1428-1437.

Chen, H. and Liu, J. (2025). Burst load scheduling latency optimization through collaborative content caching in edge-cloud computing. *Cluster Computing*, 28(3), 166.https://doi.org/10.1007/s10586-024-04891-9

Xu, J., Yu, H., Fan, G., Zhang, J., Li, Z., & Tang, Q. (2024). Adaptive edge service deployment in burst load scenarios using deep reinforcement learning. *Journal of Supercomputing*, 80(4), 5446.10.1007/s11227-023-05656-8

Cheng, Y., Cao, Z., Zhang, X., Cao, Q., and Zhang, D. (2024). Multi objective dynamic task scheduling optimization algorithm based on deep reinforcement learning. *The Journal of Supercomputing*, 80(5), 6917-6945.10.1007/s11227-023-05714-1

Kashyap, S. and Singh, A. (2023). Prediction-based scheduling techniques for cloud data center's workload: a systematic review. *Cluster Computing*, 26(5), 3209-3235.https://doi.org/10.1007/s10586-023-04024-8

He, C., Dong, Y., Li, H., and Liew, Y. (2023). Reasoning-based scheduling method for agile earth observation satellite with multi-subsystem coupling. *Remote Sensing*, 15(6), 1577-1604.https://doi.org/10.3390/rs15061577

Akbulut, O., Cavus, M., Cengiz, M., Allahham, A., Giaouris, D., and Forshaw, M. (2024). Hybrid Intelligent Control System for Adaptive Microgrid optimization: integration of rule-based control and deep learning techniques. *Energies*, 17(10), 2260-2282. https://doi.org/10.3390/en17102260

Sawicki, P., Sawicka, H., Karkula, M., and Zajda, K. (2025). Combined Rough Sets and Rule-Based Expert System to Support Environmentally Oriented Sandwich Pallet Loading Problem. *Energies*, 18(2), 268-315. https://doi.org/10.3390/en18020268

Kumar, A. (2024). Building Autonomous AI Agents based AI Infrastructure. *International Journal of Computer Trends and Technology*, 72(11), 116-125.https://doi.org/10.14445/22312803/IJCTT-V72I11P112

Murri, S., Bhoyar, M., Selvarajan, G. P., and Malaga, M. (2024). Transforming Decision-Making with Big Data Analytics: Advanced Approaches to Real-Time Insights, Predictive Modeling, and Scalable Data Integration. *International Journal of Communication Networks and Information Security*, 16(5), 506-519.

Wyrembek, M. and Baryannis, G. (2024). Using MCDM methods to optimize machine learning decisions for supply chain delay prediction: A Stakeholder-centric approach. *Logforum*, 20(2), 175-189.10.17270/J.LOG.001019

Nabeel, M. Z. (2024). AI-enhanced project management systems for optimizing resource allocation and risk mitigation: Leveraging big data analysis to predict project outcomes and improve decision-making processes in complex projects. *Asian Journal of Multidisciplinary Research & Review*, 5(5), 53-65.

Huang, B., Lu, Y., Ma, H., Yin, C., Yang, R., Shi, Y., Toa, Y., Wen, Y., and Zhong, Y. (2025). Deep Reinforcement Learning-Based Deployment Method for Emergency Communication Network. *Applied Sciences*, 15(14), 7961-7977. https://doi.org/10.3390/app15147961

Zhao, J., Fan, S., Zhang, B., Wang, A., Zhang, L., and Zhu, Q. (2025). Research Status and Development Trends of Deep Reinforcement Learning in the Intelligent Transformation of Agricultural Machinery. *Agriculture*, 15(11), 1223-1247. https://doi.org/10.3390/agriculture15111223

Cui, G., Hu, T., Zhang, W., and Bao, H. (2025). MMTransformer: a multivariate time-series resource forecasting model for multi-component applications. *Scientific Reports*, 15(1), 29740-29755.https://doi.org/10.1038/s41598-025-07162-8

Feng, B. and Ding, Z. (2024). Application-oriented cloud workload prediction: A survey and new perspectives. *Tsinghua Science and Technology*, 30(1), 34-54. 10.26599/TST.2024.9010024

Dogani, J., Khunjush, F., Mahmoudi, M. R., and Seydali, M. (2023). Multivariate workload and resource prediction in cloud computing using CNN and GRU by attention mechanism. *Journal of Supercomputing*, 79(3), 3437.10.1007/s11227-022-04782-z

Fan, X., Li, X., Yan, C., Fan, J., Yu, L., Wang, N., and Chen, L. (2023). MARC-Net: Terrain classification in parallel network architectures containing multiple attention mechanisms and multi-scale residual cascades. *Forests*, 14(5), 1060-1085. https://doi.org/10.3390/f14051060

Fan, X., Kong, J., Wang, H., Huang, K., Zhao, T., and Li, L. (2025). Multi-Scale Electromechanical Impedance-Based Bolt Loosening Identification Using Attention-Enhanced Parallel CNN. *Applied Sciences*, 15(17), 9715-9736.https://doi.org/10.3390/app15179715

Ji, Y., Shi, W., Lei, J., and Ding, J. (2025). DBRSNet: a dual-branch remote sensing image segmentation model based on feature interaction and multi-scale feature fusion. *Scientific Reports*, 15(1), 27786-27801.https://doi.org/10.1038/s41598-025-13236-4

Yuan, M., Ren, D., Feng, Q., Wang, Z., Dong, Y., Lu, F., and Wu, X. (2023). MCAFNet: A multiscale channel attention fusion network for semantic segmentation of remote sensing images. *Remote Sensing*, 15(2), 361-381.https://doi.org/10.3390/rs15020361

Chen, Y., Zhang, H., Liu, M., Ye, M., Xie, H., and Pan, Y. (2023). Traffic signal optimization control method based on adaptive weighted averaged double deep Q network. *Applied Intelligence*, 53(15), 18333-18354.https://doi.org/10.1007/s10489-023-04469-9

Yu, S., Wang, X., Shen, Y., Wu, G., Yu, S., and Shen, S. (2024). Novel intrusion detection strategies with optimal hyper parameters for industrial internet of things based on stochastic games and double deep Q-networks. *IEEE Internet of Things Journal*, 11(17), 29132-29145. 10.1109/JIOT.2024.3406386

Huo, L., Mao, J., San, H., Zhang, S., Li, R., and Fu, L. (2024). Efficient and stable deep reinforcement learning: selective priority timing entropy. *Applied Intelligence*, 54(20), 10224-10241.https://doi.org/10.1007/s10489-024-05705-6

Chen, G., Huang, Z., Wang, W., and Yang, S. (2024). A novel dynamically adjusted entropy algorithm for collision avoidance in autonomous ships based on deep reinforcement learning. *Journal of Marine Science and Engineering*, 12(9), 1562-1582. https://doi.org/10.3390/jmse12091562

Zhao, Y., Ma, D., and Liu, W. (2024). Efficient detection of malicious traffic using a decision tree-based proximal policy optimization algorithm: a deep reinforcement learning malicious traffic detection model incorporating entropy. *Entropy*, 26(8), 648-667.https://doi.org/10.3390/e26080648

Reyad, M., Sarhan, A. M., and Arafa, M. (2023). A modified Adam algorithm for deep neural network optimization. *Neural Computing and Applications*, 35(23), 17095-17112.https://doi.org/10.1007/s00521-023-08568-z

Mehmood, F., Ahmad, S., and Whangbo, T. K. (2023). An efficient optimization technique for training deep neural networks. *Mathematics*, 11(6), 1360-1381. https://doi.org/10.3390/math11061360

Kuang, S., Zhang, J., and Mohajer, A. (2024). Reliable information delivery and dynamic link utilization in MANET cloud using deep reinforcement learning. *Transactions on Emerging Telecommunications Technologies*, 35(9), e5028. https://doi.org/10.1002/ett.5028

Zhu, L., Huang, K., Fu, K., Hu, Y., and Wang, Y. (2023). A priority-aware scheduling framework for heterogeneous workloads in container-based cloud. *Applied Intelligence*, 53(12), 15222-15245.https://doi.org/10.1007/s10489-022-04164-1

Madiyev, A., Bulegenov, D., Karzhaubayev, A., Murzabulatov, M., and Bui, D. M. (2025). Energy-efficient offloading framework for mobile edge/cloud computing based on convex optimization and Deep Q-Network: A. Madiyev et al. *The Journal of Supercomputing*, 81(11), 1182-1230.

Zhong Peng holds a master's degree. His key professional experience includes serving as Director of the Network and Information Center at Guilin Tourism University and as Head of the Human Resources Department at the same institution. His primary research focus is on information technology applications and development, and human resource management.