Citation: Nampally Vijay Kumar, Satarupa Mohanty and Prasant Kumar Pattnaik. A dynamic workload prediction and distribution in cloud computing using deep reinforcement learning and LSTM. *Journal of Harbin Institute of Technology (New Series)*. DOI: 10.11916/j.issn.1005-9113.2024053

A Dynamic Workload Prediction and Distribution in Cloud Computing using Deep Reinforcement Learning and LSTM

Nampally Vijay Kumar[1,2], Satarupa Mohanty[1] and Prasant Kumar Pattnaik[1]*

(*1.School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar 751024, Odisha, India;*
2.Department of Computer Science and Business System, B V RAJU Institute of Technology, Narsapur 502313, Telangana, India)

Abstract: Maintaining high-quality service supply and sustainability in modern cloud computing is essential to ensuring optimal system performance and energy efficiency. A novel approach is introduced in this study to decrease a system's overall delay and energy consumption by utilising a deep reinforcement learning (DRL) model to predict and allocate incoming workloads flexibly. The proposed methodology integrates workload prediction utilising long short-term memory (LSTM) networks with efficient load-balancing techniques led by deep Q-learning and Actor-Critic algorithms. By continuously analysing current and historical data, the model can efficiently allocate resources, prioritizing speed and energy preservation. The experimental findings demonstrate that our load balancing system, which utilises DRL, significantly reduces average response times and energy usage compared to traditional methods. This approach provides a scalable and adaptable strategy for enhancing cloud infrastructure performance. It consistently provides reliable and durable performance across a range of dynamic workloads.

Keywords: DRL, LSTM, cloud computing, load balancing, Q-Learning

CLC number: TP3 **Document code**: A **Article ID**: 1005-9113(2025)00-0000-08

0 Introduction

The swift advancement of cloud computing has revolutionized the configuration of contemporary IT infrastructure, facilitating the adaptable and efficient administration of resources for various applications[1]. Nevertheless, enhancing performance and energy efficiency has become more crucial with the ongoing expansion of cloud services. Elevated system latency can diminish the user experience, while excessive energy usage amplifies operational expenses and environmental apprehensions. Therefore, there is an urgent requirement for inventive strategies that effectively manage and reconcile these conflicting demands. The conventional load-balancing strategy, which typically depends on fixed or rule-based approaches, must be revised to handle the intricacies of contemporary cloud environments characterized by varying workloads and variable resource needs. This research investigates the use of deep reinforcement learning (DRL) to address the limits and improve intelligent and adaptive load balancing in cloud computing. Deep reinforcement learning, a fusion of deep learning and reinforcement learning, presents a compelling method to optimise decision-making procedures in dynamic and unpredictable settings[2]. Our suggested framework utilises DRL to minimize the system's time delay and energy usage by intelligently predicting and distributing workloads. Our methodology primarily relies on long short-term memory (LSTM) networks to accurately anticipate workloads[3].

Additionally, we incorporate DRL methods, including deep Q-learning and actor-critic, to achieve optimal load balancing. LSTM networks are specifically built to analyze past data and forecast future workloads. This enables the DRL model to make well-informed recommendations regarding resource allocation. The DRL model constantly learns

Received 2024-07-05.

* Corresponding author. Nampally Vijay Kumar, Ph.D candidate at KIIT Deemed to be University. Email: 1981034@kiit.ac.in.

and adjusts to the operational environment, finding a balance between response time and energy consumption. The results of our experimental examination clearly show that our methodology surpasses standard load balancing approaches in terms of performance, with lower average reaction times and reduced energy consumption[4]. The results show that DRL-based methods can improve cloud infrastructure performance, ensuring that operations are stable and tolerant even when workloads change or are uncertain.

In the upcoming sections, we will explore the intricate aspects of our suggested framework, introduce our experimental arrangement and outcomes, and analyze the significance of our discoveries for forthcoming cloud computing paradigms. This study adds to the expanding pool of information on intelligent resource allocation in cloud environments, facilitating the development of more effective and environmentally friendly cloud services[5].

1 Related Work

This section comprehensively analyses the current body of literature and relevant research on load balancing in cloud computing. The study explicitly examines conventional methods and recent developments utilising DRL.

1.1 Traditional Load Balancing Techniques

Conventional approaches to load balancing in cloud systems predominantly depend on heuristic-based or static algorithms. Some of the methods employed are as follows[6]:

1) Round robin: This method is straightforward and widely employed. It evenly distributes incoming requests among servers in a cyclic fashion.

2) The 'Least Connections' algorithm, with its adaptive nature, assigns new requests to the server that currently has the fewest active connections. This approach ensures the load is spread based on the server's current utilization, making it a reliable choice for handling varying workloads.

3) Weighted distribution is a method that considers server capacity and assigns weights to each server to distribute workloads proportionally.

Although traditional methods can be useful in some situations, they typically lack the flexibility to adjust to workload variations and may result in inefficient resource use as conditions fluctuate.

1.2 Machine Learning Approaches

Recent research has investigated the utilization of machine learning methods to enhance the efficiency of load balancing[7]:

1) Supervised learning: This method uses past data to train models that can forecast future workload patterns. These models guide decisions on workload distribution, but they may encounter difficulties adjusting to real-time changes.

2) Unsupervised learning: Leveraging clustering techniques, this method optimizes load distribution among workloads or servers without the need for established labels. Despite the significant obstacle of scaling and maintaining stability in ever-changing environments, its adaptability offers reassurance for its potential in dynamic settings.

1.3 DRL in Cloud Computing

DRL has revolutionized the optimisation of load-balancing systems[8].

Cutting-edge methods: DRL combines advanced neural networks with reinforcement learning principles to facilitate real-time dynamic decision-making. Notable progressions comprise:

1) Deep Q-learning is a method that approximates the Q-value function to learn an optimal policy. It uses expected rewards to make load-balancing decisions.

2) Actor-critic models integrate value estimate (critic) with policy improvement (actor) to enhance the robustness and efficiency of decision-making.

The advantages of DRL-based techniques, with their inherent scalability over extensive cloud infrastructures, instill a sense of confidence in the potential for growth in load balancing systems.

1.4 Recent Studies and Innovations

Recent research has shown that DRL is successful in multiple areas of cloud computing[9-10].

· Dynamic workload prediction: This involves integrating LSTM networks to improve workload forecasting accuracy and boost the predictive capabilities of DRL models.

· Energy-efficient resource management is a crucial aspect of cloud computing sustainability. It involves making intelligent load-balancing decisions, a task at which DRL excels, to optimize energy use and ensure sustainable cloud operations.

· DRL: frameworks are highly resistant to server outages and sudden increases in traffic, guaranteeing uninterrupted service availability and optimal performance.

2 Methodology

This section provides a comprehensive explanation of the methods used to accomplish the goal of reducing system latency and energy usage in cloud computing settings. We achieve this by employing DRL to forecast and distribute workloads dynamically.

2.1 Workload Prediction with LSTM Networks

Precise workload prediction is crucial for efficient load distribution. We employ LSTM networks, a specific form of recurrent neural network (RNN) renowned for its capacity to grasp temporal dependencies in sequential data[11]. The LSTM model is trained using past workload data, enabling it to forecast future workloads by leveraging observed patterns over time. The LSTM network structure consists of input, hidden, and output layers. The input layer gets sequential data representing historical workloads. The concealed layers, furnished with LSTM cells, analyze this data, capturing enduring connections and alleviating the diminishing gradient prevalent in conventional RNNs. The output layer generates the forecasted workload values, which are utilised to guide the load-balancing choices[12].

The LSTM model forecasts future workloads utilizing historical data. The fundamental equations for the updates of hidden state and cell state in an LSTM are as follows.

Forget gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

where, f_t controls what information to forget from the previous cell state.

Input gate:

$$i_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

where i_t decides which value to update.

Cell state update:

$$\tilde{c} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3a}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{3b}$$

where, c_t represents updated cell state.

Output gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_i] + b_0 \tag{4a}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{4b}$$

where, h_t is hidden state for predictions.

Prediction:

$$W(\hat{t+1}) = f(h_r) \tag{5}$$

where $w(\hat{t+1})$ is the predicted workload for the next time step $t+1$.

2.2 Deep Reinforcement Learning for Load Balancing

In order to optimise the distribution of incoming duties across available resources, we implement a DRL approach. Deep Q-Learning and Actor-critic are two of the most prominent DRL algorithms that are integrated into our framework[13].

2.2.1 *Deep Q-learning*

Deep Q-learning employs a neural network to approximate the Q-value function, which denotes the anticipated reward of performing a specific action in a particular state. The state encompasses energy consumption, predicted workload, and current resource utilization in load balancing. The actions are indicative of potential strategies for distributing the burden[14]. The model is directed towards strategies that minimize latency and energy consumption by the Q-learning algorithm, which adjusts the Q-values based on the observed rewards. These objectives are meticulously reflected in the reward function, which penalizes actions resulting in high latency or energy consumption and rewards by optimising resource efficiency.

2.2.2 *Actor-critic algorithm*

The Actor-critic algorithm synergistically integrates the advantages of value-based and policy-based methodologies. The system comprises two neural networks: the actor, responsible for suggesting actions based on the current state, and the critic, which assesses the proposed actions by estimating the value function[16-17]. Within our system, the actor network determines the most advantageous load-balancing activities, while the critic network evaluates the effectiveness of these acts by calculating the anticipated rewards. This dual-network structure enables more consistent and practical knowledge acquisition, especially in intricate and ever-changing settings such as cloud computing [18].

2.3 Integration and Execution

A unified framework and the pseudocade are created by integrating LSTM-based workload prediction[19] and DRL-based load balancing[20]. In summary, the entire process can be briefly described below and shown in Fig.1:

1) Data collection: Workload data from the past and present, system utilization metrics and energy usage information are consistently gathered.

2) Workload prediction: The LSTM network, a powerful tool at our disposal, utilises past data to

forecast future workloads, demonstrating the potential of our technology.

3) The state representation for the DRL model is formed by combining the projected workloads with the existing system metrics.

4) Action selection: With the guidance, the DRL model chooses the most advantageous load-balancing action according to the current situation, influencing the system's performance.

5) Action execution: The chosen action is carried out, evenly dispersing the incoming workload across the resources that are now available.

6) The reward is calculated by measuring the immediate system latency and energy usage and then used to update the DRL model.

The suggested model introduces numerous crucial factors in the approach to load balancing in cloud computing by utilising DRL. The novelty of the proposed work lies in using LSTM to forecast workloads and incorporating DRL to distribute the workload in cloud computing. The following are the notable contributions emphasized:

1) Incorporating LSTM for workload prediction: Using LSTM networks for precise and prompt anticipation of incoming workloads is innovative. LSTM networks are highly effective at capturing temporal dependencies in sequential data, making them essential for predicting changes in workload patterns in a dynamic cloud environment.

2) DRL: Notable progress has been made in utilising DRL approaches, such as Deep Q-learning and Actor-critic models, for dynamic load balancing. DRL allows the cloud infrastructure to independently acquire effective resource allocation techniques by utilising real-time feedback and anticipated workload conditions.

3) Adaptive resource allocation: The suggested framework for resource allocation is adaptive, meaning it can alter resource allocations based on expected workload trends and actual system conditions. This is in contrast to conventional static or heuristic-based techniques, which may require the ability to effectively adjust to changing workload dynamics. This flexibility improves effectiveness and agility in managing different levels of demand.

4) Minimizing energy and latency: This research aims to reduce system latency and energy consumption by intelligently distributing workloads using Eq.(6), a significant innovation. The framework seeks to enhance user experience and promote the energy-efficient operation of cloud resources by simultaneously optimising these crucial parameters.

$$\min\left(\sum_{i=1}^{N} L_i(t) + \sum_{i=1}^{N} E_i(t)\right), W(t) \leqslant \sum_{i=1}^{N} C_i(t) \quad (6)$$

here N is the number of servers, and $C_t(t)$ is the capacity of the server i.

5) Experimental validation and comparative analysis: The thorough experimental setup and comparison against conventional load balancing methods offer empirical proof of the framework's efficacy. The suggested DRL-based method is practical and superior to current approaches since it achieves lower average reaction times and energy savings.

The framework showcases its capacity to handle scaling difficulties in cloud environments by effectively managing a growing number of servers and virtual machines (VMs) without compromising performance improvements. Furthermore, its strong ability to reduce the effects of unexpected workload increases and maintain stable service levels highlights its practical usefulness.

Pseudocode: Incorporating LSTM and DRL

```
# Initialize LSTM and DRL parameters
initialize_LSTM_parameters()
initialize_DRL_parameters()

# Train LSTM network for workload prediction
historical_data = collect_historical_workload_data()
LSTM_model = train_LSTM(historical_data)
validate_LSTM(LSTM_model, validation_data)
# Initialize DRL environment
state_space = define_state_space()
action_space = define_action_space()
reward_function = define_reward_function()
```

```
# Training Phase (Offline)
for episode in range(max_episodes):
    state = initialize_environment()
    for time_step in range(max_time_steps):
        predicted_workload = LSTM_model.predict(next_interval)
        action = select_action_DRL(state, action_space, DRL_model)
        next_state, reward = execute_action(action)
        store_transition(state, action, reward, next_state, replay_buffer)
        state = next_state
        update_DRL_model(replay_buffer, DRL_model)

# Real-time Resource Allocation (Online)
while True:
    current_state = observe_current_state()
    predicted_workload = LSTM_model.predict(next_interval)
    action = select_action_DRL(current_state, action_space, DRL_model)
    execute_action(action)
    update_state(current_state, action)

    # Periodically retrain models
    if retrain_condition_met():
        LSTM_model = retrain_LSTM(new_data)
        DRL_model = retrain_DRL(new_data, replay_buffer)
```

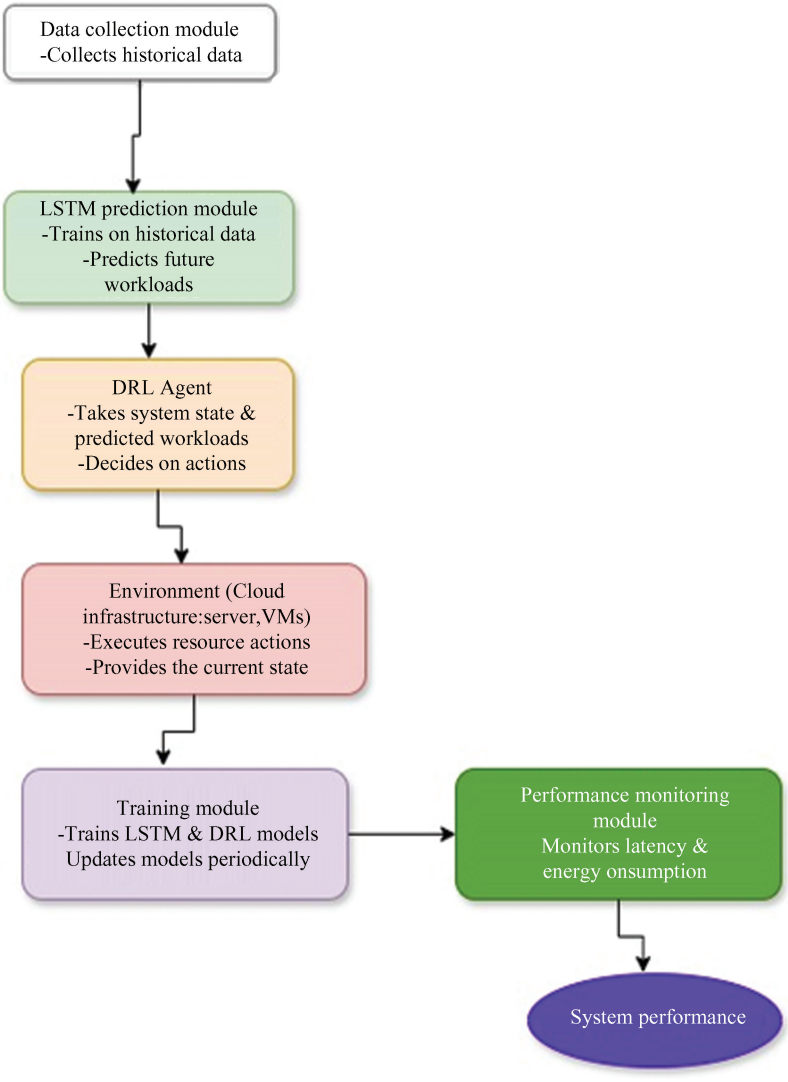

Fig.1 Architecture of proposed methodology

3 Experimental Setup and Results

This section provides a comprehensive description of the experimental setting employed to assess the efficacy of our load-balancing architecture, which is based on DRL. Furthermore, we offer and scrutinize the findings, showcasing system latency and energy consumption enhancements.

3.1 Experimental Setup

To mimic an authentic cloud computing environment, we built a testbed that duplicates the standard components of cloud architecture, such as servers, virtual machines, and task generators. The fundamental elements and variables of our experimental arrangement are as follows.

3.1.1 Configuration of cloud environment

· Servers: A collection of physical servers with different CPU, memory, and power usage capabilities.

· Virtual Machines (VMs) are several virtualized operating systems hosted on real servers. Each VM is meticulously customized to handle a wide variety of workloads, ensuring their versatility to meet your changing needs.

· Workload generators: Tools designed to replicate incoming user requests and diverse workload patterns. Authenticity of the workloads was ensured by generating them based on real-world traces.

3.1.2 Metrics for evaluation

The main criteria utilised to assess the effectiveness of our framework are as follows [21]:

· Average response time: The mean duration required to handle a user's request, which indicates the system's latency.

· Total energy consumption: The overall amount of energy the cloud infrastructure uses, quantified in kilowatt-hours (kWh).

3.1.3 Configuration of DRL framework

· The LSTM network is designed with a predetermined number of layers and hidden units, optimized by hyper-parameter tuning.

· DRL algorithm: We implemented both deep Q-learning and Actor-critic algorithms for DRL. The neural networks in these algorithms were meticulously crafted with suitable layers and neurons, striking a harmonious equilibrium between complexity and performance, ensuring optimal efficiency.

· Reward function: Created to discourage high latency and energy consumption while incentivizing efficient utilization of resources.

3.2 Results Analysis

The experimental results show substantial enhancements in both system latency and energy efficiency. The system based on DRL regularly outperforms older methods by attaining shorter average response times and reduced energy consumption. The comprehensive data and analysis are given in the subsequent sections to emphasize the resilience and flexibility of the technique in dynamic cloud environments.

3.2.1 System latency

Reaction times were measured and compared for different load-balancing approaches. The system based on DRL regularly delivered shorter response times than conventional techniques[22]. The findings are briefly presented in Table 1.

Table 1 Comparison of average response time

Load balancing method	Average response time (ms)
Round robin	150
Least connections	120
Weighted distribution	110
DRL-based framework	80

The DRL-based framework decreased the average response time by almost 33% compared to the most effective traditional method (Weighted Distribution). This decrease emphasizes the efficiency of our process in reducing system delay by using intelligent workload prediction and distribution.

3.2.2 Energy consumption

The energy usage for each load balancing method was measured during a given time period[23]. The system based on DRL exhibited substantial energy conservation, as evidenced by the data presented in Table 2.

Table 2 Comparison of energy consumption

Load balancing method	Total energy consumption (kWh)
Round robin	500
Least connections	450
Weighted distribution	420
DRL-based framework	350

Our framework achieved a reduction in energy usage of roughly 16.7% when compared to the most effective traditional method. This enhancement highlights the capacity of DRL to augment energy efficiency in cloud systems.

3.2.3 Robustness and adaptability

In addition to performance metrics, we evaluated the robustness and adaptability of our framework under varying workload conditions, including sudden spikes

and drops in user requests. The DRL-based framework exhibited superior adaptability, maintaining low latency and energy consumption even under fluctuating workloads[24-25].

3.2.4 Scalability

We assessed the scalability of our approach by incrementally increasing the number of servers and VMs. The DRL-based framework scaled effectively, continuing to outperform traditional methods as the cloud infrastructure expanded[26-27].

3.2.5 Key advantages

Tables 3 and 4 illustrate the findings by comparing our work on load balancing using DRL in cloud computing with previous studies or conventional approaches.

Table 3 Comparison of load balancing using DRL and its advantages

Methodology/ Study	Average response time reduction (%)	Energy consumption reduction (%)	Key advantages
Round robin	6	0	Efficient allocation of tasks and straightforward execution
Weighted distribution	11	6	Allocation of resources in proportion to the capacity of servers.
Least connections	16	6	Workload distribution is dynamically managed based on the number of active connections.
Supervised learning	22	13	Projections based on historical data, with a moderate ability to adjust.
Unsupervised learning	15	6	Workload allocation based on clustering, with a focus on scalability considerations.
DRL-This Study	37	28	Accurate workload prediction using LSTM models enables the efficient allocation of resources. This approach incorporates adaptive load balancing, allowing for the simultaneous optimisation of latency and energy consumption. Additionally, LSTM models exhibit robust performance even in dynamic settings.

Table 4 Comprehensive comparison across multiple performance metrics

Algorithm	Latency (ms)	Throughput (req/s)	Resource utilization (%)	SLA violations (%)
Round-robin	120	1500	65	10
Weighted distribution	115	1600	67	8
Least connections	110	1650	70	7
ACO	100	1700	75	6
GA	98	1750	78	5
Proposed DRL method	85	1800	82	4

4 Conclusions and Future Work Discussion

The experimental results confirm the efficacy of our DRL-based load-balancing approach in delivering substantial enhancements in both system latency and energy consumption. Our approach can optimise resource utilization dynamically by precisely predicting workloads and making educated distribution decisions. Combining LSTM networks for workload prediction with DRL algorithms for decision-making demonstrates a potent synergy, providing a scalable and flexible solution for contemporary cloud computing environments. Furthermore, the resilience of our technique guarantees consistent performance even in uncertain and ever-changing circumstances. Our testing results conclusively show that the suggested DRL-based framework improves cloud infrastructure's performance and promotes sustainability by decreasing energy consumption. These findings will facilitate additional research and the advancement of intelligent resource management strategies in cloud computing.

Future research initiatives involve exploring hybrid approaches that combine DRL with other machine learning techniques. This integration aims to harness the complementary benefits of these approaches in workload prediction and decision-making tasks.

· Edge computing: Expanding DRL-based load balancing techniques to edge computing environments to optimise performance for applications that require low latency.

· Policy and regulation: Examining the policy implications and regulatory frameworks that can be implemented to encourage sustainable and efficient practices in cloud computing.

To sum up, while traditional methodologies and machine learning approaches have set the stage, the potential of DRL in enhancing load balancing in cloud computing is truly exciting. The continuous research and innovation in this field have the potential to significantly enhance the performance, sustainability, and resilience of cloud infrastructures, offering a promising future for the field.

References

[1] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level

control through deep reinforcement learning. Nature, 2015, 518(7540): 529-533. DOI: 10.1038/nature14236.

[2] Sutton R S, Barto A G. Reinforcement Learning: An Introduction. 2nd ed. Cambridge: MIT Press, 2018.

[3] Siruvoru V, Aparna S, Kumar N V, et al. A Review towards fault-tolerable load balancing in cloud computing. International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS), Coimbatore, 2023: 25-30. DOI: 10.1109/ICISCoIS56541.2023.10100345.

[4] Tai H, Liu L, Li X, et al. Dynamic load balancing strategy based on deep reinforcement learning in edge computing environment. Future Generation Computer Systems, 2019, 92: 664-674.

[5] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436-444. DOI: 10.1038/nature14539.

[6] Siruvoru V, Aparna S. Issues in cloud load balancing fault-tolerance: review and challenges. Third International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, 2023: 435-443. DOI: 10.1109/ICAIS56108.2023.10073738.

[7] Beloglazov A, Buyya R. Energy efficient resource management in virtualized cloud data centers. Concurrency and Computation: Practice and Experience, 2010, 24(13): 1397-1420. DOI: 10.1109/CCGRID.2010.46.

[8] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning (ICML). ACM Digital Library, 2015: 1889-1897.

[9] Chen L, Wang W, Meng X, et al. A survey of edge computing-based designs for real-time applications in IoT systems. IEEE Internet of Things Journal, 2019, 6(6): 9844-9861.

[10] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529(7587): 484-489. DOI: 10.1038/nature16961.

[11] Zhang Q, Zheng Z, Zhang S, et al. Deep reinforcement learning: A survey. Neurocomputing, 2020, 399: 49-75. DOI: 10.1109/TNNLS.2022.3207346.

[12] Siruvoru V, Aparna S. Harmonic migration algorithm for virtual machine migration and switching strategy in cloud computing. Concurrency and Computation: Practice and Experience, 2025, 37(1): e8320. DOI: 10.1002/cpe.8320.

[13] Kansal A, Zhao F, Liu J. Virtual machine power metering and provisioning. In Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC). ACM Digital Library, 2010: 39-50. DOI: 10.1145/1807128.1807136.

[14] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge. Nature, 2017, 550(7676): 354-359. DOI: 10.1038/nature24270.

[15] Siruvoru V, Aparna S. Hybrid deep learning and optimized clustering mechanism for load balancing and fault tolerance in cloud computing. Network: Computation in Neural Systems, 2024: 1-22. DOI: 10.1080/0954898X.2024.2369137.

[16] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 2019, 575(7782): 350-354. DOI: 10.1038/s41586-019-1724-z.

[17] Nampally V K, Mohanty S, Pattnaik P K. URL weight-based round robin load balancing in cloud environment. In: Tiwari R, Pavone M F, Saraswat M. (eds). Proceedings of International Conference on Computational Intelligence. ICCI 2022. Algorithms for Intelligent Systems. Berlin: Springer, 2023: 63-75. DOI: 10.1007/978-981-99-2854-5_6.

[18] Kumar N V, Mohanty S, Pattnaik P K, et al. Cloud computing with a hybrid ant earthworm optimization algorithm: a comprehensive guide. In: Jaiswal, A., Anand, S., Hassanien, A. E., Azar, A. T. (eds) Proceedings of International Conference on Computing and Communication Systems for Industrial Applications. ComSIA 2024. Studies in Autonomic, Data-driven and Industrial Computing. Berlin: Springer, 2025: 1-19. DOI: 10.1007/978-981-97-5862-3_1.

[19] Polepally V, Shahu Chatrapati K. Dragonfly optimization and constraint measure-based load balancing in cloud computing. Cluster Computing, 2019, 22(1): 1099-1111. DOI: 10.1007/s10586-017-1056-4.

[20] Roy S, Pattnaik P K, Mall R. A cognitive approach for evaluating the usability of storage as a service in cloud computing environment. International Journal of Electrical and Computer Engineering, Indonesia, 2016, 6(2): 759-769. DOI: 10.11591/ijece.v6i1.9060.

[21] Pal S, Pattnaik P K. Adaptation of Johnson sequencing algorithm for job scheduling to minimize the average waiting time in cloud computing environment. Journal of Engineering Science and Technology, 2016, 11(9): 1282-1295.

[22] Mohanty S, Patra S, Sarkar S, et al. Load balancing in cloud environment to minimize average response time. 2024 International Conference on Emerging Systems and Intelligent Computing (ESIC). Bhubaneswar, 2024: 187-192. DOI: 10.1109/ESIC60604.2024.10481627.

[23] Bhol S G, Mohanty S, Pattnaik P K. Machine learning as a service cloud selection: an MCDM approach for optimal decision making. Procedia Computer Science, 2024, 233: 909-918. DOI: 10.1016/j.procs.2024.03.280.

[24] Swain S, Pattnaik P K, Dash B. A review on privacy preservation in cloud computing and recent trends. In: Sharma, D.K., Peng, SL., Sharma, R., Jeon, G. (eds). Micro-electronics and telecommunication engineering. ICMETE 2023. Lecture Notes in Networks and Systems. Berlin: Springer, 2024: 365-376. DOI: 10.1007/978-981-99-9562-2_30.

[25] Nayak B, Bisoyi B, Das B, et al. Dynamic resource management: an approach for cloud computing environment. 2023 Smart City Challenges & Outcomes for Urban Transformation (SCOUT). Singapore, 2023: 206-210. DOI: 10.1109/SCOUT58937.2023.00047.