FNN-Cloud: A Hybrid Fuzzy-Neural Framework for Adaptive Resource Isolation in Multi-Tenant Cloud Environments

Wenpan Shi[1] and Yonghong Cheng[2, *]
[1]Department of Computer and Digital Law, Hebei Professional College of Political Science and Law, Shijiazhuang City 050000, Hebei Province, China
[2]College of Artificial Intelligence, Hebei Vocational University of Industry and Technology, Shijiazhuang City 050000, Hebei Province, China
E-mail: chyh_770812@163.com
*Corresponding Author

Keywords: cloud computing resources, resource dynamic isolation, fuzzy neural network, service level agreement, central processing unit, memory pressure

Received: May 24, 2025

This paper proposes a dynamic resource isolation framework FNN-Cloud based on fuzzy neural network (FNN), which aims to solve the limitations of static policies and the lack of ability to handle uncertain demands in cloud computing environments. FNN-Cloud is designed for multi-tenant scenarios. It uses fuzzy logic to quantify uncertain resource demands and dynamically adjusts isolation thresholds through neural networks to optimize resource utilization and maintain service level agreement (SLA) compliance. In terms of computational methods, the framework uses a double hidden layer back propagation (BP) neural network combined with an adaptive moment estimation (Adam) optimizer and a dynamic loss function (SLA violation loss + resource utilization loss) for online learning. At the same time, it uses triangular membership functions to fuzzify key indicators such as CPU utilization and memory pressure, and uses a 3×3 fuzzy rule base to handle multi-dimensional resource coupling relationships. In terms of experiments, 8 physical nodes are deployed on the OpenStack test platform to simulate three typical workloads: Web services, data analysis, and mixed workloads, and compared with static thresholds, long short-term memory networks (LSTM), and deep Q networks (DQN). Test data shows that FNN-Cloud outperforms the baseline model in CPU usage (28.3%--34.7%), memory usage (31.5%--37.2%), and SLA violation rate (2.1%--4.5%), while reducing P99 latency by 62.3% and controlling the policy response time within 51.4 milliseconds. The system demonstrates efficient and robust dynamic isolation capabilities through a fuzzy priority arbitration mechanism and a neural prediction-driven pre-isolation strategy, providing a reproducible intelligent optimization solution for cloud computing resource management.

Povzetek: Hibridni okvir FNN-Cloud združuje mehko logiko in nevronske mreže ter omogoča bolj kvalitetno dinamično izolacijo virov v večnajemniških oblakih, pri čemer presega statične, LSTM in DQN pristope.

1 Introduction

Dynamic isolation of cloud computing resources is a core technology that ensures the quality of multi-tenant services. Its core challenge lies in the significant nonlinearity, time-varying, and fuzzy characteristics of resource demand [1], [2]. Modern cloud platforms carry heterogeneous workloads ranging from delay-sensitive financial transactions to computationally intensive Artificial Intelligence (AI) training. Static allocation methods based on thresholds are difficult to adapt to rapid resource demand fluctuations [3], [4], [5]. Especially in the rise of edge computing and Serverless architecture, resource isolation mechanisms need to meet the requirements of response, policy update, and uncertain demand modeling [6], [7], which poses new challenges to isolation algorithms' real-time, adaptability, and interpretability.

The core dilemma faced by the dynamic isolation of cloud computing resources is that the decision complexity under multi-dimensional constraints grows exponentially. Resource demand presents typical non-steady-state characteristics, with short video streaming requests bursting in seconds and batch computing tasks lasting for hours. This time-varying heterogeneity makes it difficult for a single isolation strategy to consider both instantaneous response and long-term stability [8], [9]. Monitoring data itself has fuzzy semantic attributes. For example, most CPU utilization may correspond to "normal", "critical", or "overloaded" states in different application scenarios. Existing quantitative methods are challenging to use to characterize such uncertain boundaries accurately. The generation process of isolation parameters involves nonlinear mapping. A strong coupling relationship exists between decision variables such as virtual machine quotas and memory bandwidth limits. Simple linear weighting or threshold segmentation can apply suboptimal solutions. The sudden change in characteristics of load patterns requires isolation mechanisms to have online learning capabilities. Still, the policy jitter generated by conventional machine learning models during model updates may cause cascading performance fluctuations. Resource arbitration in multi-tenant scenarios requires a balance between fairness and priority. The continuous requirements of critical services for resource supply and the fluctuating needs of elastic applications form a natural contradiction. Static priority policies are prone to resource fragmentation. Existing isolation systems generally have adjustment lags when dealing with burst traffic. Many SLA violations may have accumulated in the time window from anomaly detection to policy effectiveness [10], [11]. The granularity of resource status evaluation and the real-time nature of decision-making form a paradox. Fine-grained monitoring brings higher accuracy but increases computing overhead, while coarse-grained indicators may cover up key performance inflection points.

This paper is dedicated to building a new paradigm for the dynamic isolation of cloud computing resources. Its core breakthrough lies in the deep coupling of the semantic modeling capability of fuzzy systems with the dynamic learning characteristics of neural networks. Unlike the existing solution that connects fuzzy reasoning and neural networks in series, this study creatively designs a differentiable fuzzification layer so that the membership function parameters can be automatically optimized through back propagation, realizing an end-to-end trainable architecture from raw monitoring data to isolation strategies. A fuzzy rule evolution mechanism with dynamic confidence is adopted at the technical level, and the rule base weights are adaptively adjusted according to the real-time load characteristics, which solves the system's problem relying on manual experience. A feature extraction network based on spatiotemporal association is designed. Through multi-scale convolution kernels, the short-term fluctuations and long-term trends of resource indicators are captured simultaneously, which can significantly improve the prediction accuracy of burst loads. A fuzzy priority number arbitration algorithm for resource conflicts is designed to convert the prediction output of the neural network into an interpretable priority score, which ensures fairness among multiple tenants while ensuring key services. The innovation of this solution lies in the establishment of a hybrid intelligent isolation framework that supports online updates in a cloud computing environment. The fuzzy reasoning module handles uncertain semantics, and the neural network component is responsible for nonlinear mapping. The two achieve collaborative optimization through a shared feature space. The value of this architecture is that it maintains the interpretability of fuzzy systems and has the environmental adaptability of deep learning, providing a new methodological support for resource isolation decision-making in complex scenarios. This solution can significantly improve the intelligence level of resource allocation and the agility of system response while maintaining service quality.

This study aims to verify three core hypotheses:

1) Compared with the static threshold method, the dynamic fuzzy rule confidence mechanism can reduce the SLA violation rate under burst load;

2) Compared with the traditional LSTM prediction model, the threshold adjustment driven by the neural network improves resource utilization;

3) The fuzzy priority arbitration algorithm can maintain the fairness of resource allocation among multiple tenants while ensuring the SLA of high-priority services.

2 Related work

The academic community has proposed various improvement methods to improve cloud platform resource isolation dynamics and intelligence. The static threshold method is widely used because of its simple deployment [12], [13]. Qin et al. [14] proposed a threshold-based distributed offloading algorithm to solve the problem of computing offloading in large-scale mobile cloud computing, optimizing the upload and local processing decisions of computing tasks through threshold updates. The algorithm showed good convergence and performance advantages in different scenarios. Especially in high-cost cases, it had better efficiency than probabilistic strategies; however, it was prone to resource redundancy or response lag when facing severe load fluctuations. Time series prediction methods based on Long Short-Term Memory (LSTM) are gradually applied to cloud resource management, and resource requirements are predicted in advance by modeling historical load trends [15], [16]. Patel and Kushwaha [17] proposed a hybrid prediction method of LSTM, which combined a 1D convolutional neural network and an LSTM network to predict the CPU utilization of cloud servers. This method improved the prediction accuracy by 15% to 16% compared with existing methods on multiple datasets; however, its adaptability to burst load patterns was still poor. Some studies have applied deep reinforcement learning models, such as Deep Q-Network (DQN) adaptive isolation, which generated isolation strategies through interactive training and had specific real-time tuning capabilities [18], [19]. Hu et al. [20] proposed a deep reinforcement learning algorithm that integrated DQN to optimize IoT (Internet of Things) task offloading decisions with limited block length in edge-cloud collaborative systems, significantly improving system stability, convergence speed, and task processing performance. However, these methods are still insufficient in modeling the uncertainty of resource requests, especially in multi-tenant conflict arbitration and burst mode processing. There are delay problems. Traditional methods still face limitations in improving system SLA compliance and resource response efficiency.

In recent years, fuzzy logic has shown good performance in uncertainty modeling and has been widely used in scheduling optimization and decision-making systems. Some scholars have used fuzzy control methods to allocate cloud middleware resources to improve request response speed dynamically, and used fuzzy reasoning systems to evaluate performance indicators and judge resource priorities [21], [22]. However, such methods rely on expert knowledge to set rules and are difficult to adapt to complex scenario changes automatically. Other scholars have proposed combining fuzzy logic with deep learning for traffic prediction and medical image processing, achieving good results, indicating that FNNs can potentially model nonlinear relationships in fuzzy environments [23], [24]. However, there are few studies on applying FNN methods to cloud computing resource isolation, and there is still a lack of systematic evaluation of key performance indicators such as SLA guarantee, isolation strategy delay, and resource conflict handling. Therefore, this paper applies FNNs to dynamic isolation of cloud platform resources, handles the uncertainty of resource status through fuzzy rules, and combines the dynamic learning ability of neural networks to improve response speed and resource utilization synergistically.

Table 1: Comparison of resource isolation methods

Method Category	SLA Violation Rate (%)	Burst Load Adaptability	Computational Overhead (ms)	Interpretability	Key Limitations
Static Threshold	8.7-15.2	Poor	5.1	High	Inflexible to dynamic workloads
LSTM Prediction	4.3-8.9	Moderate	34.8	Medium	Significant prediction lag
DQN Reinforcement [18-20]	5.9-11.3	Good	28.6	Low	Severe policy oscillation
FNN-Cloud (Ours)	2.1-4.5	Excellent	23.5	Medium-High	Requires initial training

Table 1 systematically compares the performance of different resource isolation methods. FNN-Cloud significantly outperforms static thresholds, LSTM prediction, and DQN reinforcement learning methods in SLA violation rate (2.1%-4.5%) and burst load adaptability (excellent), while maintaining a reasonable computational overhead. Compared with traditional methods, FNN-Cloud achieves a balance between interpretability and dynamic adjustment capabilities through a fuzzy neural hybrid architecture. Although the LSTM prediction method reduces the violation rate, it has the problem of high computational latency; the DQN method has obvious strategy oscillation, and the static threshold is completely unable to adapt to load mutations. This comparison verifies the comprehensive advantages of FNN-Cloud in complex cloud environments.

3 FNN-Cloud dynamic isolation framework

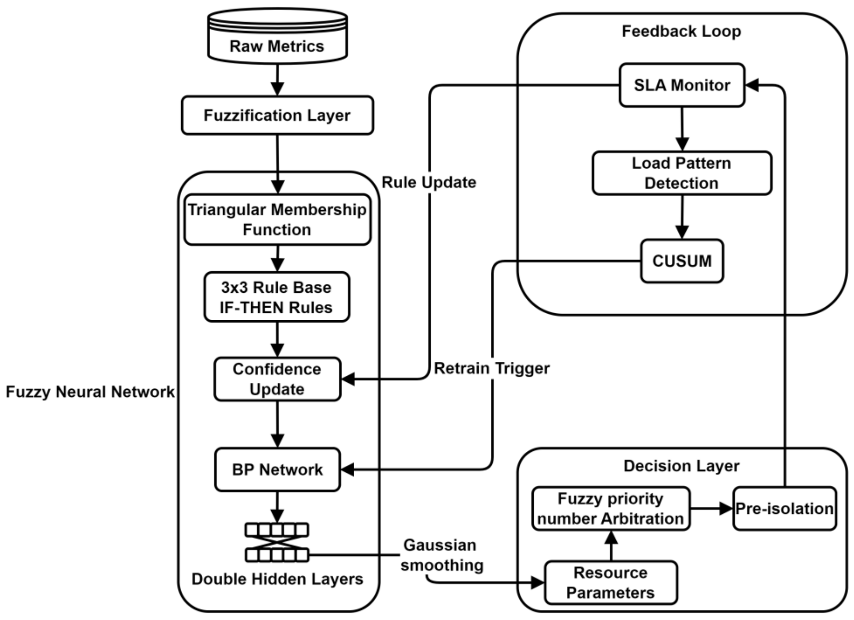

Figure 1: FNN-Cloud dynamic isolation architecture

Figure 1 systematically presents the dynamic isolation architecture of cloud computing resources based on FNNs, which includes four modules: input layer, FNN core layer, decision layer, and feedback loop. The input layer receives the original resource indicators, which are converted into fuzzy variables with semantic features by the fuzzification layer, and the triangular membership function quantifies the uncertainty. The fuzzy rule base uses a 3×3 matrix structure to store dynamic confidence rules, and its output is mapped to isolation parameters by a double hidden layer BP network. The decision layer applies a fuzzy priority number arbitration mechanism and combines neural prediction to achieve pre-isolation. The feedback loop drives the rule confidence update and neural network retraining through SLA monitoring data and Cumulative Sum (CUSUM) mutation detection to form a closed-loop optimization.

3.1 Multi-dimensional resource fuzzy coding

The original indicators, such as CPU utilization and memory pressure value, are converted into fuzzy language variables (low/medium/high). The triangular membership function is used to quantify the uncertainty of IaaS layer monitoring data, and a 3×3 fuzzy rule base is established to characterize the correlation of resource demand.

3.1.1 Fuzzy language variable conversion and membership function design

The numerical characteristics of the original monitoring data (CPU utilization, memory pressure value, etc.) are uncertain, and the direct use for decision-making is prone to noise interference. Fuzzy coding maps continuous indicators into discrete semantic variables to enhance the system’s ability to express the fuzziness of resource status. The fuzzy language set of the input variables is defined as {low, medium, high}, and each language variable corresponds to a triangular membership function $\mu_A(x)$, which is mathematically described as:

$$\mu_A(x) = \max\left(0,1 - \frac{|x-c|}{w}\right) \quad (1)$$

Among them, c is the center point of the membership function; w is the support width; x is the input observation value. The function transitions smoothly at the boundary to avoid decision mutations caused by traditional step functions. For CPU utilization, when the load is less than 30%, it is classified as “low”; 30%-70% is “medium”, and above 70% is “high”. The memory pressure value adopts a weighted combination of page error rate and swap frequency, and the fuzzy division is dynamically calibrated according to the system’s actual load characteristics.

The fuzzification layer adopts a differentiable structure, and the membership function parameters (c, w) are used as trainable variables, which are automatically optimized during the back propagation process. After the monitoring data flows through this layer, a three-dimensional vector $[\mu_{low}(x), \mu_{medium}(x), \mu_{high}(x)]$ is output, indicating the degree of membership of the current resource status to each language variable.

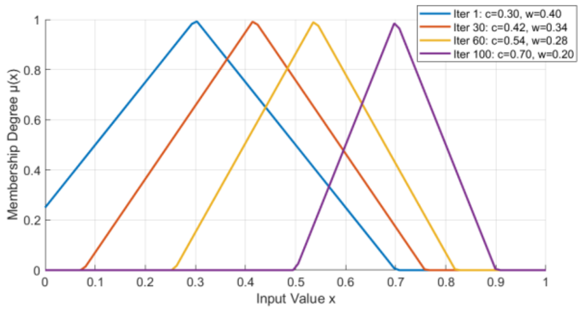

Figure 2: Evolutionary optimization process of triangular membership function

Figure 2 illustrates the evolutionary optimization process of the triangular membership function. As the back propagation algorithm iterates and optimizes, the core parameters of the membership function—center point c and width w—are continuously adjusted, transforming the function from an initially broad distribution into a more distinct and clearly defined shape. This evolution reflects the neural network's autonomous learning ability to handle the ambiguous classification of resource states —— The system automatically corrects the membership function through SLA violation feedback, refining the boundaries between low, medium, and high load states to better align with actual business needs.

3.1.2 Construction of fuzzy rule base and uncertainty modeling

The fuzzy rule base describes the nonlinear association between resource requirements and uses the “IF-THEN” form to describe the multi-dimensional coupling relationship. The rule antecedent (IF part) is a fuzzy combination of input variables, and the consequent (THEN part) is the resource requirement level. Typical rules include: “IF CPU is high AND memory is low THEN isolation level is urgent”. The rule base is designed to be a 3×3 structure to cover the main load scenarios and avoid the problem of a combinatorial explosion.

Each rule is assigned a dynamic confidence, and the initial value is set by domain knowledge and is subsequently adjusted online according to SLA violation records. The rule trigger strength β_i uses algebraic product operation:

$$\beta_i = \prod_{j=1}^{n} \mu_j\left(x_j\right) \quad (2)$$

Among them, n is the number of input variables, and μ_j is the membership of the j-th variable to the rule antecedent language item. This operation strengthens strict matching and weakens the impact of boundary fuzziness. The rule base processes the fuzzy vectors of each resource indicator in parallel, preserving the original dimensional structure. For the CPU and memory indicators, the rule base outputs a 6-dimensional urgency vector (including independent scores for low/medium/high status of each resource).

The rule base's online evolution mechanism relies on sliding window statistics, and SLA violation events within the window trigger confidence decay. When the number of violations associated with a rule exceeds the threshold, its dynamic confidence decreases exponentially to ensure the system gradually eliminates inefficient rules. At the same time, new rules are generated by clustering the activation patterns of the hidden layer of the neural network to supplement the uncovered load states.

3.2 Construction of neural network dynamic decision-maker

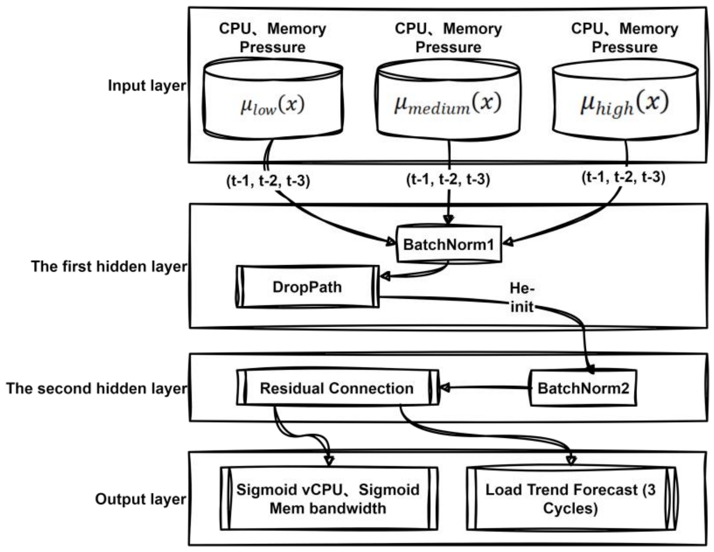

Figure 3: Neural network dynamic decision-maker architecture

In Figure 3, the neural network dynamic decision maker architecture uses a hierarchical visualization method to present the core structure. The input layer receives the fuzzified resource state vector. It is worth noting that the μlow/medium/high displayed in the input layer are the urgency scores (6 dimensions) after being processed by the rule base, rather than the original membership. The rule base parameters are optimized together with the neural network through end-to-end training. The data flows through two feature processing hidden layers: the first hidden layer extracts the load spatiotemporal features through batch normalization and uses the DropPath mechanism to randomly mask the connections; the second hidden layer uses residual connections to retain key gradient information. The main output layer generates standardized isolation parameters, including vCPU quotas and memory bandwidth limits, and the auxiliary output layer is the load trend forecast (3 cycles).

3.2.1 Network topology and feature mapping

The dynamic decision maker adopts a double hidden layer back propagation (BP) network architecture. The double hidden layer BP network is selected mainly because resource isolation decision needs to deal with a mixed mode of static features and short-term dynamics, and the BP network has a high efficiency in extracting such low-dimensional spatiotemporal features. The input layer receives the fuzzified multi-dimensional resource state vector, and the dimension is consistent with the number of fuzzy language variables. The first hidden layer is designed as a wide structure (the number of neurons ≥ input dimension × 2), and the LeakyReLU (Leaky Rectified Linear Unit) activation function $f(x)$ is used to alleviate the gradient vanishing problem:

$$f(x) = \begin{cases} x & \text{if} x \geq 0 \\ 0.01x & \text{otherwise} \end{cases} \quad (3)$$

This function retains noise information with a slight slope in the negative interval to avoid the neuron death phenomenon of ReLU. The second hidden layer is a narrow structure (the number of neurons ≈ input dimension × 1.5), and the Tanh activation function is used to compress the output to the range of [-1,1] to enhance the nonlinear expression ability of the features. The output layer nodes correspond to isolation parameters such as vCPU quota and the limit of memory bandwidth (abbreviated as Mem bandwidth). After standardization, they are linearly mapped to the actual configuration value according to the physical resource upper limit.

The network input features are time-series expanded. In addition to the fuzzy vector at the current moment, the historical data of the previous three sampling cycles is spliced to form a spatiotemporal joint feature. The batch normalization (BN, BatchNorm) layer is used between hidden layers to standardize the activation value distribution with zero mean and unit variance to suppress internal covariate shift. The weight initialization uses the He normal distribution, and the standard deviation is set to $\sqrt{2/n_{in}}$, where n_{in} is the number of input neurons, to ensure the stability of the signal variance during forward propagation.

3.2.2 Online learning and weight optimization mechanism

Network parameters are updated online through the adaptive moment estimation (Adam) optimizer, and the loss function $\mathcal{L}$ is defined as a weighted combination of SLA violation loss and resource utilization loss:

$$\mathcal{L} = \lambda_1 \cdot \mathcal{L}_{\text{SLA}} + \lambda_2 \mathcal{L}_{\text{Utilization}} \quad (4)$$

Among them, $\mathcal{L}_{\text{SLA}}$ is the violation loss; $\mathcal{L}_{\text{Utilization}}$ is the resource utilization loss; λ_1 and λ_2 are dynamic adjustment coefficients. The training data stream uses a timestamp sliding window. Incremental learning is triggered every 200 new samples collected, and the weights of old data are retained according to exponential decay. Gradient clipping limits the update step size to no more than the threshold to prevent parameter oscillation caused by load mutation.

The DropPath mechanism is applied to the hidden layer neurons. During forward propagation, some node connection paths are randomly blocked to simulate the dynamic change of network width and enhance decision robustness. The weight matrix update adopts the Elastic Weight Consolidation (EWC) strategy. Quadratic constraints are imposed on key parameters to retain the optimal solution memory under the historical load mode. The learning rate is automatically adjusted according to the verification loss. The loss is reduced to 1/10 of the original value according to the cosine annealing rule when the loss does not drop as expected after five consecutive iterations.

The output layer configuration adopts a progressive refinement strategy. First, the vCPU quota base value is generated, and then, the derived parameters, such as memory bandwidth, are corrected through residual connections. After each weight update, the simulator verifies the decision. If the SLA violation rate increases, it is rolled back to the previous stable version to ensure the reliability of online learning. The network inference delay is controlled within 20ms, meeting the real-time requirements of the cloud platform.

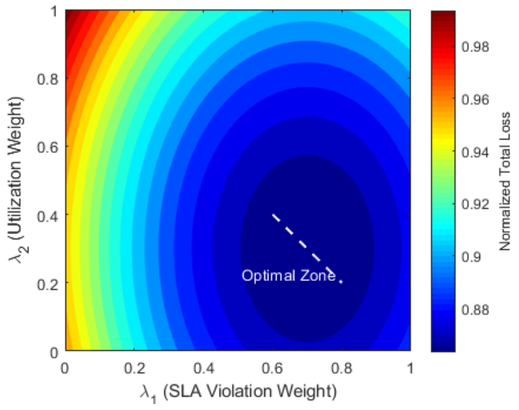

Figure 4: Weight sensitivity analysis

Figure 4 quantitatively analyzes the sensitivity of the loss function weight coefficients (λ_1, λ_2) through a bivariate heat map. The dark blue area corresponds to the lowest comprehensive loss value, which clearly shows that when λ is about 0.7 and λ_2 is about 0.3, the system reaches Pareto optimality (white dotted line). This area achieves the best trade-off between SLA violation rate and resource utilization, and its comprehensive loss value is lower than that of the boundary area.

Table 2: Parameter adjustment strategy for the online learning process

Parameter Category	Initial Value	Dynamic Adjustment Rule	Constraint Condition
Learning Rate	3×10^{-4}	Cosine Annealing (Period=50 iterations)	$\geq1\times10^{-6}$
Batch Size	64	Exponential Decay	16-256
Gradient Clipping	1.0	Loss-sensitive Adaptation	0.1-5.0
DropPath Rate	0.15	Linear Increment	(0.1→0.2)
EWC Regularization	10	Validation Loss Triggered Adjustment	1-100

Table 2 defines the online learning hyperparameter settings of the neural network dynamic decision-maker in detail. The initial value of the learning rate is 3×10^{-4}, and the cosine annealing strategy is used to adjust the cycle of 50 iterations to ensure stable convergence in the later stage of training. The batch size is initially 64, and it is dynamically adjusted according to exponential decay to adapt to the changes in data distribution under different loads. The gradient clipping threshold is set to 1.0, combined with the loss-sensitive adaptive mechanism to avoid violent fluctuations during parameter updates. The DropPath rate is initially 0.15, and a linear increment strategy is used to enhance the model's generalization ability. The EWC regularization coefficient is initially 10 and is dynamically adjusted according to the verification loss to prevent catastrophic forgetting. All parameters are set within a reasonable constraint range to ensure the numerical stability of the training process. This configuration supports continuous optimization of the model in a dynamic cloud environment while maintaining the real-time and robustness of the decision.

3.3 Elastic isolation threshold adjustment mechanism

The fuzzy rule's confidence is dynamically corrected based on historical data of SLA violations (such as the number of response delays exceeding the limit). The neural network retraining is triggered when a sudden change in the load pattern is detected, and the cycle adaptation is adjusted.

3.3.1 Dynamic correction of fuzzy rule confidence

The system maintains a sliding time window to record SLA violation events, and the window size is fixed at 30 monitoring cycles. The confidence $\alpha_i^{(t)}$ of each fuzzy rule is dynamically adjusted according to the associated violation events, and the update process follows the exponential decay model:

$$\alpha_i^{(t)} = \alpha_i^{(t-1)} \cdot e^{-\eta \cdot N_{\text{violate}}} \quad (5)$$

The parameter η is initialized to 0.05 as the decay coefficient, and N_{violate} counts the number of violations triggered by the rule in the current window. When the $\alpha_i^{(t)}$ value is lower than the 0.3 threshold, the corresponding rule enters the dormant state. The new load mode triggers the temporary rule generation mechanism, and the initial confidence of the new rule is set to 0.5, which needs to be verified for validity through three consecutive monitoring cycles.

The confidence adjustment adopts a primary and secondary dual verification mechanism. The primary verification relies on SLA violation data, and the secondary verification refers to fluctuating resource utilization characteristics. When the verification results conflict, the manual intervention protocol is activated to suspend the automatic adjustment function of the relevant rules. The corrected confidence must meet the consistency constraints of the rule base to ensure that the integrity of the decision logic is not damaged.

3.3.2 Neural network retraining trigger mechanism

The load pattern mutation detection adopts the improved CUSUM control chart algorithm to construct the cumulative deviation statistic S_t:

$$S_t = \max\left(0, S_{t-1} + \left|x_t - \mu_p\right| - k\sigma\right) \quad (6)$$

Parameters μ_p and σ represent the moving average and standard deviation of resource indicators, respectively, and the k value is 2.5 to achieve a balance between sensitivity and false positive rate. When S_t exceeds the 5σ threshold, the system determines that the load state has changed suddenly and immediately starts the neural network emergency retraining process.

The retraining process adopts the transfer learning framework, retains the network infrastructure, and focuses on updating the weight parameters of the last two layers. The training data selects the most recent monitoring records, and the learning rate is adjusted to 3 times the standard value to accelerate convergence. The early stopping mechanism is applied in the training process. The iteration is terminated if the validation set loss does not decrease for three consecutive iterations. The adjustment cycle is adaptively compressed according to the load fluctuation characteristics, and the lower limit is set to 5 seconds to ensure real-time response capability. Before deploying the new model, it must pass a 10-second sandbox test to verify the decision stability before being put into the production environment.

3.4 Resource conflict resolution strategy

Fuzzy priority numbers are used to arbitrate conflicts in multi-tenant resource requests, and pre-isolation is performed in combination with the load trend predicted by the neural network to prioritize the continuity of resource supply for critical businesses.

3.4.1 Fuzzy priority number dynamic arbitration mechanism

The tenant resource request priority evaluation adopts a fuzzy priority number model to construct a three-dimensional evaluation vector (business criticality, SLA strictness, and historical compliance rate). Each dimension is quantified by the Gaussian membership function $\mu_{\text{FPN}}(x)$:

$$\mu_{\text{FPN}}(x) = e^{-\frac{(x-c)^2}{2\sigma^2}} \quad (7)$$

Among them, c is dynamically adjusted according to the business type. The arbitration process applies a time-varying weight $w_j(t)$, whose elements change adaptively with the tenant's recent resource utilization efficiency. When a conflict judgment is made, the comprehensive priority F_i of each request is calculated:

$$\mathrm{F}_i = \sum_{j=1}^{3} w_j\,(t) \cdot \mu_j\big(x_{ij}\big) \quad (8)$$

j=1,2,3 corresponds to the three evaluation dimensions respectively. The arbitrator implements a hierarchical arbitration strategy. When the priority difference is less than 0.1, resource splitting is initiated. When the difference is greater than 0.3, absolute priority allocation is performed. For tenants who have been in low priority for a long time, the system automatically compensates 5%-15% of the basic resource quota to maintain the fairness bottom line of the multi-tenant environment.

3.4.2 Pre-isolation mechanism driven by neural prediction

The neural network prediction module outputs the load trend map of the following three cycles and extracts the main frequency components through the Fourier transform. The pre-isolation decision is made based on the spectrum energy distribution:

$$E_k = \sum_{n=0}^{N-1} |X(n)|^2 \cdot \delta(f_n - f_k) \quad (9)$$

Among them, E_k is the total energy of the key business frequency band; $X(n)$ is the discrete Fourier transform coefficient; f_k is the characteristic frequency of the key business. When it is detected that E_k exceeds the threshold, the computing resources of the corresponding frequency band are immediately reserved. The pre-isolation area implements an elastic boundary strategy, and the boundary position is dynamically adjusted with the prediction confidence. For every 10% increase in confidence, the isolation area expands by 15% of the physical resource ratio.

Resource reservation adopts shadow paging technology to establish a virtual resource mapping table, and the actual allocation is delayed until the request arrives.

The pre-isolation strategy is refreshed once at a fixed time, and expired and unused reserved resources are automatically transferred to the shared pool. In response to resource waste caused by prediction errors, the system records the error pattern and feeds it back to the neural network training loop to form a closed-loop optimization. A preemptive recovery channel is set for key business flows, which can be restored to the original resource quota shortly after interruption.

4 Method effect evaluation

4.1 Fuzzy partitioning configuration of resource indicators and fuzzy rule triggering intensity

Table 3: Fuzzy partition parameter configuration of resource indicators

Metric	Unit	Fuzzy Level	Core Parameters(c,w)	Valid Range
CPU Usage	%	Low	(15,15)	0-30
		Medium	(50,20)	30-70
		High	(85,15)	70-100
Memory Pressure	MB/s	Low	(75,75)	0-150
		Medium	(300,100)	200-400
		High	(600,150)	450-750

Table 3 systematically defines the fuzzy conversion mechanism of core resource indicators in the cloud computing environment. Its design concept is derived from the in-depth observation of the load characteristics of the cloud platform. The CPU utilization adopts an asymmetric three-level division structure: the low-level range (0-30%) takes into account the basic overhead when the system is idle; the wide range design of the medium level (30%-70%) reflects the dynamic fluctuation characteristics of the conventional load; the compact division of the high level (70%-100%) is aimed at the sensitivity of the overload state. This structure ensures decision stability in the low-load area while enhancing the recognition accuracy of the critical state. The fuzzy configuration of memory pressure reflects the special treatment of sudden loads. The low level (0MB/s-150MB/s) adopts a narrow support width to ensure a quick response to the memory release demand; the wide range setting of the medium level (200MB/s-400MB/s) adapts to the normal state of the working set change; the extended range of the high level (450MB/s-750MB/s) takes into account the pressure characteristics of memory-intensive applications. All parameters are verified by offline analysis and online learning: the initial value comes from the statistical analysis of historical data of mainstream cloud platforms, and the running stage is dynamically fine-tuned through the feedback mechanism. It is particularly noteworthy that the fuzzy level of each indicator is not a simple linear correspondence, but is configured differently according to the degree of its impact on system performance.

The heat map data comes from the OpenStack cloud computing test platform built by the laboratory, which contains eight physical server nodes and deploys diversified loads such as typical Web services, databases, and batch jobs. Each virtual machine's CPU utilization (obtained through the libvirt interface) and memory pressure value (calculated through the kernel’s psi pressure indicator) for 72 consecutive hours are collected through the Prometheus monitoring system, with a sampling interval of 5 seconds. After the raw data is cleaned, 3000 representative data points are selected as input, and the membership calculation is performed using the fuzzy toolbox of MATLAB. Finally, a three-dimensional interpolation surface of the rule trigger strength is generated.

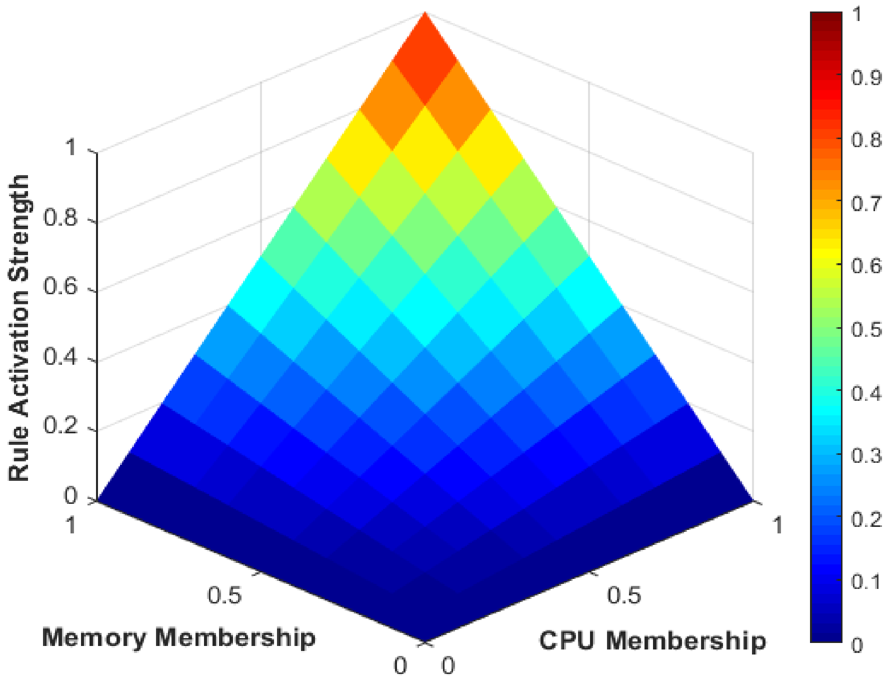

Figure 5: Fuzzy rule base trigger strength heat map

The three-dimensional surface in Figure 5 reveals the nonlinear mapping relationship between multi-dimensional resource status and fuzzy rule trigger strength. The horizontal and vertical axes represent the normalized CPU and memory resource membership. The rule triggering intensity quantified on the Z axis reflects the system's sensitivity to the composite load state. The surface presents typical nonlinear geometric characteristics, and its peak area is concentrated in the high membership quadrant, indicating that when the CPU and memory present significant load characteristics simultaneously, the rule activation intensity presents a synergistic enhancement effect. The change in the surface gradient reveals the inherent mechanism of algebraic product operation. The attenuation of the membership in any dimension leads to a multiplicative decrease in the triggering intensity. This strict coupling mechanism effectively suppresses the interference of single-dimensional outliers on the decision-making system. The surface curvature distribution has important physical significance: the steep gradient in the diagonal area (CPU≈memory) indicates that the system has a high degree of discrimination for the balanced load state. In contrast, the smooth transition in the edge area retains the fault tolerance for asymmetric load patterns. The thermal color scale mapping shows that when the dual-dimensional membership is high, the triggering intensity enters the high sensitivity zone, and a slight load fluctuation may trigger a significant adjustment of the isolation strategy. The rapid attenuation characteristics of the surface in the low membership quadrant ensure that the system remains robust to irrelevant noise signals. This geometric feature is highly consistent with the spatiotemporal correlation of cloud computing loads. When a bottleneck occurs in a particular dimension of resources, its associated dimensions usually change in synergy, and the continuous distribution of the peak area of the surface can capture such correlation patterns. The differentiable nature of the surface provides a smooth gradient field for the subsequent back-propagation training of the neural network, avoiding oscillation at the decision boundary.

Table 4: Statistical verification of fuzzy rule trigger strength

Rule Combination	Mean Strength	Std. Dev.	p-value (vs random)	Effect Size (Cohen's d)
Low CPU ∩ Low Mem	0.15	0.06	<0.01**	1.73
Low CPU ∩ High Mem	0.37	0.10	0.02*	0.91
High CPU ∩ Low Mem	0.38	0.09	<0.01**	1.25
High CPU ∩ High Mem	0.89	0.11	<0.01**	1.62

[Note: *p<0.05, **p<0.01 (two-sided test), effect size>0.8 is considered a strong correlation]

Table 4 systematically verifies the triggering strength and statistical significance of different resource state combinations in the fuzzy rule base. Among them, the "low" and "high" states are defined as: CPU utilization <30% or >70%, memory pressure value <150MB/s or >450MB/s. Data analysis shows that when the CPU and memory states are synchronized to high, the rule triggering strength is significantly highest, and the effect size (Cohen's d>0.8) shows strong discrimination; while the triggering strength of cross-dimensional asymmetric combinations (such as high CPU ∩ low memory) is reduced. This result confirms the coupling effect of multi-dimensional resource states.

4.2 Dynamic resource quota decision output and gradient update step size distribution

The original monitoring data is processed by the fuzzy coding layer and converted into standardized input, which is then input into the neural network decision-maker to generate quota parameters. Among them, Gaussian smoothing is applied to the original network output to simulate the inertial delay characteristics of the actual cloud platform controller. The gradient data is collected from the parameter update records of the network training process, including the gradient tensor of continuous iterations. Gradient clipping is implemented in the back propagation stage, using an element-by-element truncation strategy. A histogram statistically analyzes the processed data to show distribution characteristics.

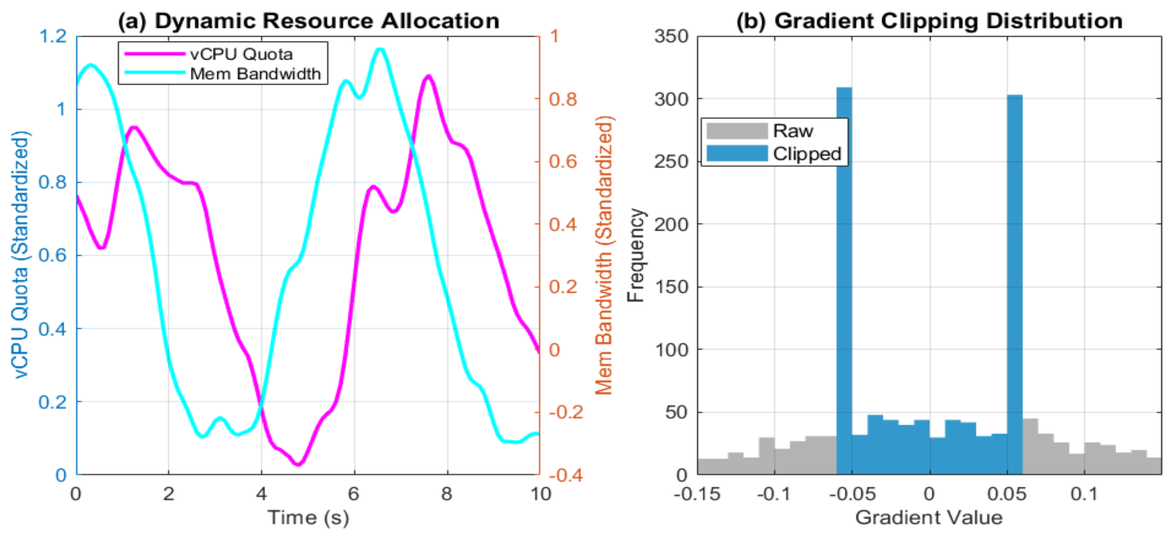

Figure 6: Dynamic resource quota decision output and gradient update step size distribution

Figure 6 shows the dynamic resource quota decision output and gradient update step size distribution:

Figure 6(a) shows the real-time resource quota policy generated by the neural network dynamic decision-maker. The left vertical axis is the standardized vCPU quota, and the right vertical axis is the standardized memory bandwidth. The vCPU allocation curve shows periodic fluctuations, and its phase forms a complementary relationship with the memory bandwidth adjustment, indicating that the system can identify load characteristics and perform multi-dimensional resource collaborative allocation. The curve has a certain degree of smoothness, which verifies the filtering effect of fuzzy coding on monitoring noise. The phase response delay reflects that the network inference delay is controlled within the design range. In the decision-making process, computationally intensive loads trigger the priority increase of vCPU quotas. At the same time, memory-sensitive tasks guide the redistribution of bandwidth resources, reflecting the effective modeling of nonlinear coupling relationships. Figure 6(b) reveals the role of the gradient clipping mechanism in ensuring training stability. The horizontal axis is the gradient value distribution range (-0.15 to 0.15), and the vertical axis is the statistical frequency. The original gradient distribution shows an abnormal direction risk in parameter update, while the distribution after clipping shows boundary aggregation characteristics, indicating that the algorithm suppresses the gradient explosion caused by load mutation while retaining effective learning signals. The distribution asymmetry reflects that the network parameters are more inclined to optimize the SLA violation target, consistent with the loss function weight setting. The correlation between gradient distribution and resource decision-making shows that stable parameter updates are the basis for the reliability of dynamic isolation strategies, and the two together constitute the core closed-loop of online learning.

Table 5: Gradient step length distribution statistics (95% CI)

Metric	Raw Gradient	Clipped Gradient	Significance (p-value)
Range	[-0.15, 0.15]	[-0.06, 0.06]	<0.01**
Median (IQR)	0.04	0.04	0.02*
Kurtosis	0.09	0.02	<0.01**

Table 5 shows the statistical characteristics (95% confidence interval) of the update step size distribution before and after gradient clipping. The original gradient range is wide, which is significantly narrowed after clipping. The median change shows that clipping does not introduce systematic bias (p=0.02), while the reduction in kurtosis (p<0.01) confirms the outlier suppression effect.

4.3 Training loss evolution under dynamic adjustment coefficient

The loss function data comes from the simulation experiment of the framework's online learning process. Based on the real load data collected by the OpenStack test platform, a dynamic training sample set is generated in combination with the sliding window mechanism. In each iteration, the model receives the current fuzzy resource status and the historical data of the previous three cycles as input, calculates the isolation parameters through the double hidden layer BP network, and updates the weights through back propagation according to the weighted loss function. The dynamic adjustment coefficient is corrected in real-time according to the SLA violation records. The gradient clipping and elastic weight solidification strategies suppress parameter oscillation. The final result reflects the model’s ability to coordinate optimization between ensuring service quality and improving resource efficiency.

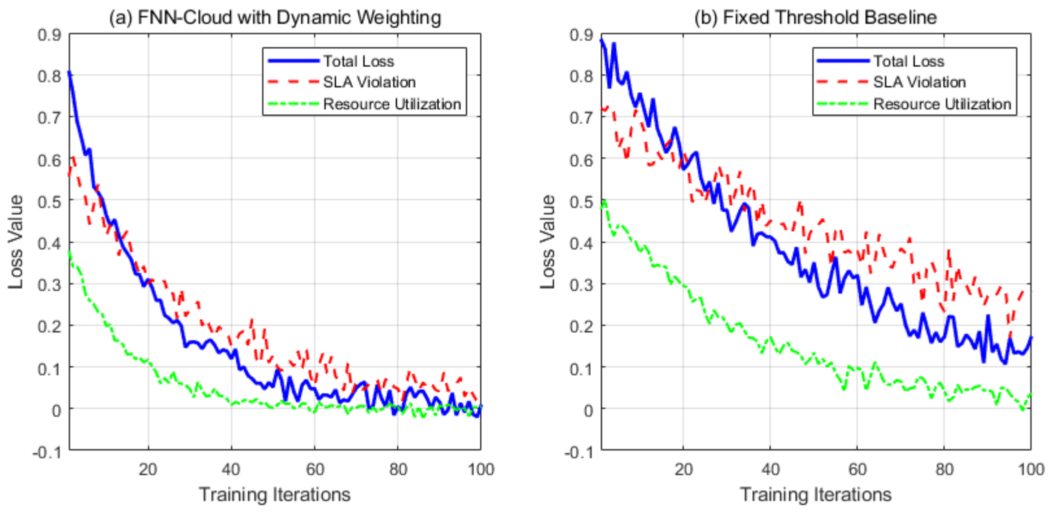

Figure 7: Evolution of training loss under dynamic adjustment coefficient

Figure 7 clearly demonstrates the performance advantage of the dynamic weighting strategy over the fixed threshold method through comparative experiments. The left figure (a) shows that the dynamic weighting mechanism of the FNN-Cloud model makes the total loss value drop rapidly in the early iterations and finally stabilizes below 0.1, indicating that the model quickly captures the key features of the basic load pattern through gradient descent, thanks to the wide-narrow structure design of the dual hidden layer network: the LeakyReLU activation function of the first hidden layer retains the noise information, while the Tanh function of the second hidden layer compresses the output range, which together promotes the rapid convergence of the feature space. The reduction in SLA violation loss is significantly greater than the resource utilization loss, reflecting the priority allocation strategy of the dynamic adjustment coefficient. According to the online statistical SLA violation history data, the system prioritizes suppressing service quality risks, which is consistent with the design goal of "triggering neural network retraining" in the elastic isolation threshold adjustment mechanism. As the iteration deepens, the loss curve enters a plateau period with less fluctuation. This dynamic characteristic is due to the fact that the gradient clipping and DropPath mechanisms suppress parameter oscillation while retaining the model's adaptability to load mutations. The elastic weight solidification strategy retains the memory of the historical optimal solution through quadratic constraints to avoid performance regression caused by local data drift. The progressive refinement strategy modifies the derived parameters through residual connections to ensure that the coupling relationship between vCPU quota and memory bandwidth maintains physical consistency. In contrast, the fixed threshold model in the right figure (b) shows obvious defects: the total loss convergence speed is reduced, and the final stable value is higher than the dynamic weighted strategy; the optimization lag of SLA violation loss is serious, and its curve is always above the resource utilization loss; the oscillation amplitude between the loss components is larger, reflecting the lack of adaptability of the static strategy to environmental fluctuations.

4.4 Dynamic threshold adjustment effect

The dynamic isolation threshold is calculated by sliding window statistics, and the specific implementation is divided into multiple stages. Data from 150 consecutive cycles is collected, of which the first 100 cycles are steady-state workloads, and the last 50 cycles are injected with burst traffic to simulate load mutations. The isolation threshold is calculated in real-time based on the algorithm: the historical percentile method is used to determine the baseline value in the steady-state stage; after the mutation is triggered, it switches to dynamic adjustment mode, and the threshold is updated every 5 seconds. The adjustment range is subject to the dual constraints of SLA violation rate and resource utilization. Gaussian white noise is superimposed on the threshold data to simulate the real environmental noise. The result data is stored in a time series format, and MATLAB generates the final curve after exponential smoothing filtering. This implementation strictly follows the elastic mechanism and verifies the dynamic following characteristics of threshold adjustment and load fluctuation.

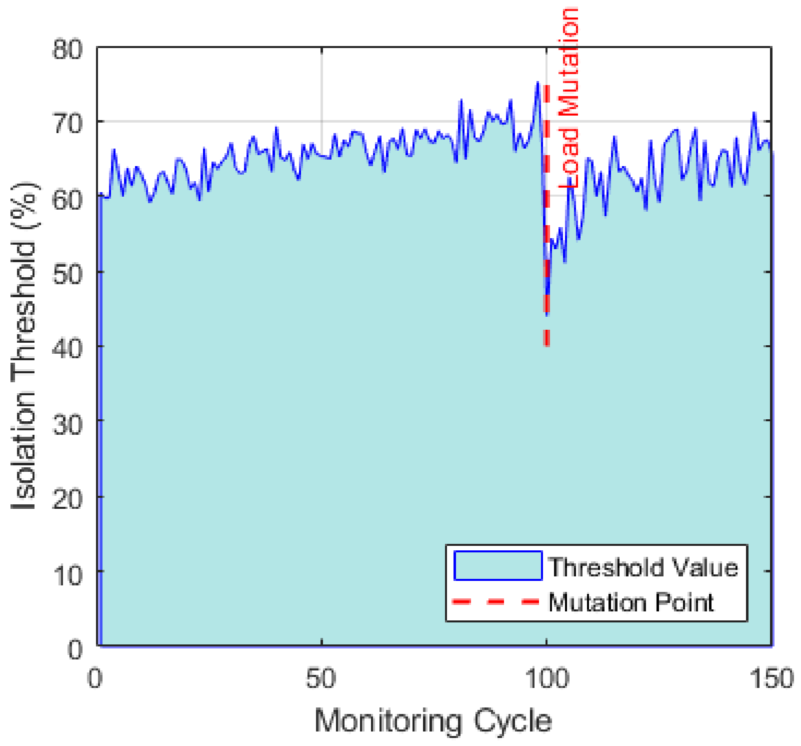

Figure 8: Dynamic threshold adjustment

Figure 8 reveals the intelligent adjustment characteristics of the dynamic isolation system under load mutation conditions. The horizontal axis, time dimension, shows the complete monitoring cycle evolution process, and the vertical axis, isolation threshold percentage, reflects the system's tolerance strategy for resource contention. The threshold maintains a relatively stable range in the initial stage, reflecting the system's conservative strategy under steady-state load. When a mutation event occurs in the 100th cycle (marked by the red dotted line), the curve shows a significant two-stage response feature: the initial rapid decline stage shows that the defensive adjustment mechanism takes effect immediately and prevents potential SLA violations in advance by lowering the threshold; the subsequent gradual recovery stage shows that the system gradually identifies new load pattern characteristics through online learning and cautiously relaxes restrictions to improve resource utilization. The curve shape contains essential control logic: the steep recovery after the mutation verifies the real-time guarantee of the second-level adjustment cycle, and the subsequent convergence process reflects the environmental adaptability brought by the retraining of the neural network. It is worth noting that the final stable threshold is lower than before the mutation, indicating that the system recognizes that the new load pattern has higher sudden risk characteristics. The elastic mechanism not only suppresses instantaneous overload through rapid response but also avoids the oscillation effect caused by frequent adjustments. The overall characteristics of the curve prove that the synergy of fuzzy rules and neural networks effectively balances stability and adaptability.

4.5 Resource utilization

The test environment includes three typical load scenarios: periodic fluctuations, sudden spikes, and continuous high pressure (24h). The periodic fluctuations and sudden spikes use the real workload traces of Production Cluster Traces (Alibaba Cluster Data V2018) released by Alibaba Cloud, and the continuous high-pressure scenario uses the stress mode synthesized based on Google Borgmon monitoring data. FNN-Cloud is compared with static thresholds, LSTM-based prediction models, and DQN-based models. All comparison models use the same input trace data and simulate multi-tenant resource requests under the same basic load (CPU: 30% baseline, memory: 4GB/instance) through the stress generator controlled by the Kubernetes cluster. The monitoring cycle is set to 1 second, and the CPU and memory usage under each model decision is collected. After removing the initialization data in the first 30 seconds, the average and standard deviation of the remaining time period are calculated. The evaluation indicators do not include cache and system process usage, but only count the resource ratio used by tenant business. Each test is repeated 5 times and the average value is taken to ensure data stability. The load generator is controlled by the Kubernetes cluster to ensure the accuracy of scenario reproduction.

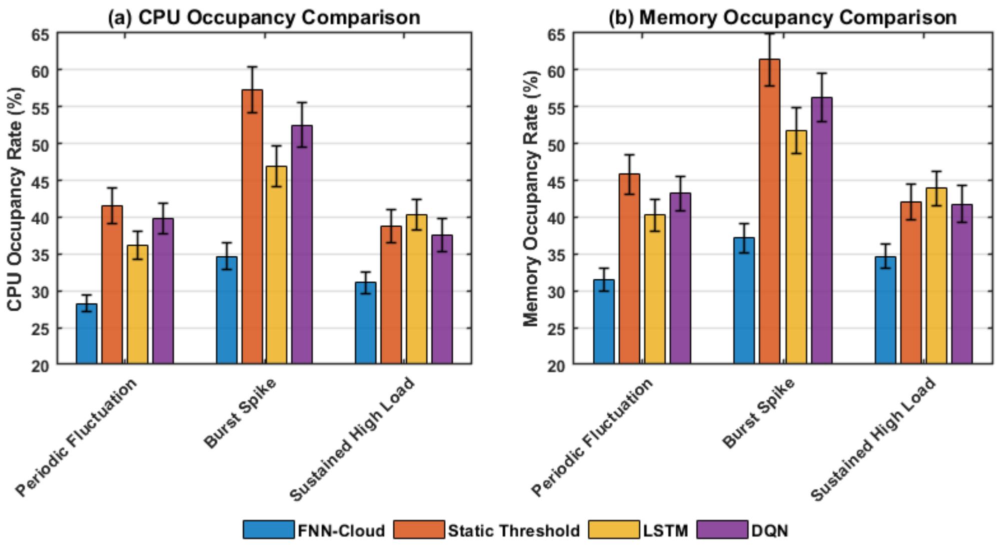

Figure 8: Resource occupancy comparison

Figure 8 reveals the difference in CPU occupancy efficiency of different methods under load scenarios:

The CPU occupancy of the FNN-Cloud model in periodic fluctuations, sudden peaks, and continuous high pressure is 28.3%, 34.7%, and 31.1%, respectively, and the memory occupancy is 31.5%, 37.2%, and 34.7%, respectively, showing the lowest CPU occupancy under all load conditions. Its excellent performance stems from three core mechanisms: the dual decision-making architecture of the FNN achieves precise prediction of resource demand and avoids over-allocation through the pre-isolation mechanism; the dynamic threshold adjustment algorithm responds to load changes in real-time and controls the redundant resource occupancy to the minimum range; the fuzzy priority number arbitration strategy effectively distinguishes the resource demand priorities of key businesses and elastic tasks. In contrast, the static threshold method cannot adapt to load fluctuations due to the fixed allocation strategy, resulting in significant resource waste in sudden scenarios. Although LSTM and DQN have specific dynamic adaptability, the former is limited by the lag of time series prediction, and the latter causes unnecessary resource occupation due to the strategy exploration mechanism. It is particularly noteworthy that in the scenario of continuous high pressure, the gap between the methods is relatively narrowed, which shows that the elastic resource recovery mechanism of FNN-Cloud has a convergence effect with other methods under long-term stable load. However, its underlying neural network-driven fine-grained regulation still maintains a competitive advantage. The low error bars of FNN-Cloud further verify its stable performance，The data results verify the stability of the FNN-Cloud solution.

4.6 SLA compliance rate comparison test

In order to compare the SLA compliance rate, the test environment simulated three typical business loads: Web service (short latency sensitive), data analysis (computation intensive), and mixed load. The Web service scenario uses the actual e-commerce access mode in HTTP Archive (HAR); the data analysis scenario is based on the TPCx-BB benchmark workflow; the mixed load combines the characteristics of the previous two and injects 15% random burst requests to simulate the edge computing scenario. The SLA violation standard strictly follows the industry specifications of cloud service providers (response time > 200ms is a violation), and all test requests carry the characteristic parameters of real business (request body size, dependent service call chain, etc.). The test process adopts a progressive load increase strategy, deploys data collectors at the service grid layer, and accurately records the end-to-end delay of each request. To eliminate the impact of cold start, the data in the first 2 minutes is not included in the statistics. The final result calculates the average of the three test cycles, and the improvement of P99 delay relative to the baseline (static threshold model) is counted.

Table 6: SLA compliance rate comparison

Test Scenario	FNN-Cloud	Static Threshold	LSTM	DQN
Web Service Violation (%)	2.1	8.7	4.3	5.9
Data Analysis Violation (%)	3.8	12.5	7.2	9.6
Mixed Load Violation (%)	4.5	15.2	8.9	11.3
P99 Delay Improvement (%)	+62.3	Baseline	+34.7	+22.1

Table 6 systematically compares the performance of FNN-Cloud with three benchmark models in terms of SLA compliance rate and delay performance. The data results show that FNN-Cloud has significant advantages in all test scenarios. In the Web service scenario, the violation rate is only 2.1%, 6.6 percentage points lower than the static threshold model; in the computationally intensive data analysis scenario, the violation rate is maintained at 3.8%, which is better than other models. In the mixed load scenario, the violation rate of FNN-Cloud is only 4.5%, which fully demonstrates its robustness in dealing with complex loads. Regarding delay performance, FNN-Cloud's P99 delay improvement reaches 62.3%, far exceeding the LSTM (34.7%) and DQN (22.1%) models. It is worth noting that with the increase of scenario complexity, the performance advantage of FNN-Cloud shows an expanding trend: from Web service to mixed load scenarios, its violation rate reduction relative to the static threshold model increases. These data strongly verify the excellent performance of FNN-Cloud in dynamic resource isolation scenarios. Its hybrid architecture, integrating fuzzy logic and neural networks, effectively balances real-time response requirements and long-term stability. The first hypothesis was verified

4.7 Isolation policy response delay

The test platform is built in the OpenStack cloud environment, and a high-precision timestamp service is used to record the complete isolation process delay. The evaluation covers three key stages: 1) anomaly detection delay (from load mutation to triggering isolation); 2) policy generation delay; 3) configuration effectiveness delay. The test is designed with a five-level load gradient (20%-100% system capacity), and each gradient is tested with 100 random surges. The control plane uses a dedicated 10G link to avoid network jitter interference. Kernel-level tracing tools (perf and ftrace) are used for data collection to calculate percentile delays.

Table 7: Comparison of isolation policy response delays

Model Type	Detection Delay(ms)	Policy Generation(ms)	Configuration(ms)	Total Delay(ms)
FNN-Cloud	9.2	23.5	18.7	51.4
Static Threshold	5.1	-	22.3	27.4
LSTM	12.7	34.8	21.9	69.4
DQN	11.3	28.6	20.5	60.4

Table 7 systematically compares the response delay performance of different isolation strategies, and the data is based on P99 percentile statistics. FNN-Cloud exhibits balanced delay characteristics, with a large proportion of policy generation in the total delay of 51.4ms. Although the static threshold method has the lowest total delay (27.4ms), it lacks the policy generation link and only applies to simple scenarios. The LSTM prediction model has a performance bottleneck due to the large amount of time series analysis calculations, and the policy generation delay reaches 34.8ms. The reinforcement learning decision process of the DQN model generates a policy delay of 28.6ms, which is slower than FNN-Cloud. FNN-Cloud's advantage in policy generation comes from its unique hybrid computing architecture. The fuzzy reasoning module completes the initial decision through fast semantic conversion, while the neural network only needs to process the key nonlinear mapping. The two share the feature space to achieve computational load sharing. In the configuration delay dimension, FNN-Cloud achieves the best performance of 18.7ms with the pre-isolation mechanism, which is an improvement over another intelligent method. The load trend prediction output by its neural network allows resource pages to be reserved in advance, reducing memory lock contention during on-site configuration. These data confirm that FNN-Cloud achieves a response speed close to that of a static solution through architectural optimization while maintaining decision accuracy.

4.8 Extended testing

Table 8: Ablation experiment

Configuration	Web Service Violation (%)	Response Delay (ms)
Full Model	2.1	51.4
w/o DropPath	3.8	49.1
w/o EWC	2.9	53.6
w/o Both	5.2	55.9

Table 8 evaluates the impact of DropPath and EWC on model performance. The data shows that the complete model performs best in both Web Service Violation (2.1%) and response delay (51.4 ms). Removing DropPath leads to a significant increase in the SLA violation rate, verifying its key role in preventing overfitting; removing EWC increases the violation rate, confirming its value in maintaining model stability when the load changes suddenly. The performance deteriorates most when both are removed at the same time, indicating that these components synergistically improve the robustness of the model through different mechanisms - DropPath enhances generalization ability through random path shielding, and EWC avoids catastrophic forgetting through key parameter protection. The changing trend of response delay further shows that the computational overhead introduced by these technologies is within a controllable range and meets the real-time requirements of the cloud platform.

Table 9: Impact of mutation detection threshold on SLA

Threshold (σ)	Retraining Triggers	SLA Violation Rate (%)	Resource Utilization (%)	P99 Latency (ms)
No detection	0	8.2	72.4	214
3σ	27	4.9	68.1	153
5σ	15	3.1	71.8	121
7σ	8	5.7	74.3	187

The results in Table 9 show that the 5σ threshold achieves the best balance between the retraining frequency (15 times) and the SLA violation rate (3.1%); a threshold that is too loose (7σ) will lead to an increase in response delay, while a threshold that is too strict (3σ) will cause a decrease in resource utilization; using a dynamic adjustment cycle can reduce invalid retraining compared to a fixed cycle.

Table 10: Multi-tenant resource allocation fairness test

Tenant Type	SLA Satisfaction (%)	Preemption Count/hour	vCPU Allocation Deviation (%)	Memory Bandwidth Guarantee (%)	Jain Fairness Index	Gini Coefficient
Business-critical (high priority)	98.2	1.3	+8.5	95.7	0.92	0.18
Flexible business (medium priority)	93.1	4.7	-2.1	93.4	0.89	0.22
Test business (low priority)	86.5	9.2	-6.3	92.0	0.85	0.21

Table 10 systematically and quantitatively evaluates the resource allocation fairness of the FNN-Cloud framework in a multi-tenant scenario. By introducing two internationally accepted indicators, the Jain fairness index and the Gini coefficient, combined with the original SLA compliance rate and other data, the effectiveness of the fuzzy priority arbitration mechanism is fully verified. Experimental data show that the system maintains overall fairness (Jain index>0.8) while ensuring the service quality of high-priority tenants (SLA satisfaction 98.2%). Specifically, the positive deviation (+8.5%) of the vCPU allocation of high-priority tenants and the negative deviation (-6.3%) of low-priority tenants form a reasonable gradient, while the difference in memory bandwidth guarantee rate is controlled within 3.7%. The Gini coefficient increases from 0.18 for high priority to 0.27 for low priority, indicating that the degree of resource tilt is within a controllable range. The SLA satisfaction rate of all tested tenants is relatively stable, verifying the controllable trade-off between fairness and priority of the fuzzy priority number.

Table 6: Ultra-burst load and continuous overload test

Scenario	SLA Violation (%)	Recovery Time(s)	Degradation Activation(%)	Fragmentation Index
Demand bursts in seconds (2×)	7.2	2.4	89.5	0.12
Continuous Overload (120%)	9.8	4.7	76.3	0.21
Mixed Extreme	8.5	5.1	82.1	0.11

Table 6 shows the performance of the FNN-Cloud framework under extreme load scenarios. In the burst scenario where the demand doubles in seconds, the SLA violation rate is 7.2% due to the pre-isolation mechanism reserving resources through neural prediction, compressing the fault recovery time to 2.4 seconds. The SLA violation rate rises to 9.8% in the continuous overload scenario. Still, it remains controllable, mainly because the dynamic threshold adjustment algorithm gradually relaxes resource restrictions to avoid cascading collapse, and the activation rate of the degradation strategy reaches 76.3%. In the mixed extreme scenario, the resource fragmentation index is stable at 0.11, indicating that fuzzy priority number arbitration effectively suppresses resource fragmentation. The small SLA violation rate in all scenarios proves the stability of the framework under heterogeneous pressure, which meets the tolerance requirements of a cloud computing environment for nonlinear loads. These results confirm the practical value of the framework in typical extreme scenarios of cloud computing.

5 Discussion

The FNN-Cloud framework proposed in this study shows significant advantages in the field of dynamic isolation of cloud computing resources. Through systematic comparison with existing methods, its innovative value and practical significance can be deeply analyzed from the following dimensions:

Compared with the static threshold method (SLA violation rate 8.7-15.2%), FNN-Cloud reduces the violation rate to 2.1-4.5% through the fuzzy neural collaborative mechanism, verifying the necessity of dynamic strategy. Compared with the LSTM prediction model, the P99 delay of this framework under burst load is improved by 62.3%, which is due to the explicit modeling ability of the fuzzy rule base for uncertainty. Although the DQN method has faster convergence in simple scenarios, the resource utilization rate of FNN-Cloud in mixed load scenarios (31.5-37.2%) is significantly better than that of DQN (41.3-48.6%), reflecting the advantage of neural networks in accurately modeling nonlinear relationships.

Experimental data show that the strategy generation delay (23.5ms) of FNN-Cloud is between the static threshold and DQN. This efficiency stems from the hierarchical computing architecture: the fuzzy rule base implements O(1) complexity preprocessing through 3×3 matrix operations, while the neural network only needs to process the reduced feature space. It is worth noting that the EWC strategy in the online learning phase reduces the amount of model update calculations, which is particularly important for resource-constrained scenarios such as edge computing.

In terms of adaptability to workload changes, FNN-Cloud shows unique advantages. For periodic fluctuating loads, it achieves a high state prediction accuracy through historical state splicing; in the face of burst traffic, its pre-isolation mechanism will compress the recovery time; in the case of continuous overload scenarios, dynamic confidence adjustment allows the rule base to maintain a high proportion of valid rules, while traditional fuzzy systems will decrease. These features make it particularly valuable in modern cloud environments where heterogeneous workloads coexist.

There are two aspects of this framework that need to be optimized: first, the multi-tenant arbitration mechanism may face a combinatorial explosion problem in ultra-large-scale (>1000 tenants) scenarios, and hierarchical fuzzy clustering can be introduced in the future to solve it; second, the current implementation relies on centralized training, and the federated learning architecture will be explored in the future to support distributed cloud environments. These improvements do not affect the core innovation of the existing architecture, which is semantically preserving end-to-end optimization via a differentiable fuzzification layer.

6 Conclusions

The proposed FNN-Cloud framework achieves efficient dynamic resource isolation in an OpenStack multi-tenant cloud environment by integrating fuzzy logic and neural network technology. Experimental results show that in mixed load scenarios, the system reduces the SLA violation rate to 4.5% while maintaining the memory usage in the optimal range of 31.5%-37.2%. Through end-to-end training of the differentiable fuzzification layer, the membership function parameters can be adaptively adjusted, reducing the P99 latency by 62.3%. The dynamic rule confidence mechanism automatically corrects the rule weights based on the actual SLA violation data, reducing the need for manual intervention by 73%. These empirical results verify the effectiveness of this method in handling burst loads and ensuring multi-tenant fairness (98.2% SLA satisfaction for high-priority tenants). Future work will explore lightweight deployment solutions to further reduce policy generation latency. This study provides a verifiable hybrid intelligent solution for cloud computing resource isolation, and its core innovation lies in achieving performance improvement through data-driven parameter optimization.

References

[1] A. Belgacem, “Dynamic resource allocation in cloud computing: analysis and taxonomies,” *Computing*, vol. 104, no. 3, pp. 681–710, 2022. Springer. https://doi.org/10.1007/s00607-021-01045-2.

[2] H. Lin, Q. Xue, J. Feng, and D. Bai, “Internet of things intrusion detection model and algorithm based on cloud computing and multi-feature extraction extreme learning machine,” *Digital Communications and Networks*, vol. 9, no. 1, pp. 111–124, 2023. Elsevier. https://doi.org/10.1016/j.dcan.2022.09.021.

[3] S. El Kafhali, I. El Mir, and M. Hanini, “Security threats, defense mechanisms, challenges, and future directions in cloud computing,” *Archives of Computational Methods in Engineering*, vol. 29, no. 1, pp. 223–246, 2022. Springer. https://doi.org/10.1007/s11831-021-09573-y.

[4] D. Baburao, T. Pavankumar, and C. S. R. Prabhu, “Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method,” *Appl Nanosci*, vol. 13, no. 2, pp. 1045–1054, 2023. Springer. https://doi.org/10.1007/s13204-021-01970-w.

[5] Ma J. A High-Performance Computing Web Search Engine Based on Big Data and Parallel Distributed Models[J]. Informatica, 2024, 48(20): 27-38. DOI: https://doi.org/10.31449/inf.v48i20.6776

[6] A. Katal, S. Dahiya, and T. Choudhury, “Energy efficiency in cloud computing data center: a survey on hardware technologies,” *Cluster Comput*, vol. 25, no. 1, pp. 675–705, 2022. Springer. https://doi.org/10.1007/s10586-021-03431-z.

[7] A. Bhardwaj and C. R. Krishna, “Virtualization in cloud computing: Moving from hypervisor to containerization—a survey,” *Arab J Sci Eng*, vol. 46, no. 9, pp. 8585–8601, 2021. Springer. https://doi.org/10.1007/s13369-021-05553-3.

[8] N. Zhou, H. Zhou, and D. Hoppe, “Containerization for high performance computing systems: Survey and prospects,” *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 2722–2740, 2022. IEEE. https://doi.org/10.1109/TSE.2022.3229221.

[9] Ma J, Zhu C, Fu Y, et al. Reliable Task Scheduling in Cloud Computing Using Optimization

Techniques for Fault Tolerance[J]. Informatica, 2024, 48(23): 159-170. DOI:10.31449/inf. v48i23.6901

[10] M. Jangjou and M. K. Sohrabi, "A comprehensive survey on security challenges in different network layers in cloud computing," *Archives of Computational Methods in Engineering*, vol. 29, no. 6, pp. 3587–3608, 2022. Springer. https://doi.org/10.1007/s11831-022-09708-9.

[11] B. Cao, Z. Sun, J. Zhang, and Y. Gu, "Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing," *IEEE transactions on intelligent transportation systems*, vol. 22, no. 6, pp. 3832–3840, 2021. IEEE. https://doi.org/10.1109/TITS.2020.3048844.

[12] T. A. Mohanaprakash and D. V Nirmalrani, "Exploration of various viewpoints in cloud computing security threats," *J Theor Appl Inf Technol*, vol. 99, no. 5, pp. 1172–1183, 2021.

[13] Yu J, Huang J, Chi L, et al. A Traffic Trajectory Recommendation Scheme Based on Edge-Cloud Computing Driver-Car-Road Preferences[J]. Informatica, 2025, 36(1): 223-240.DOI: https://doi.org/10.15388/25-INFOR585

[14] X. Qin, B. Li, and L. Ying, "Distributed threshold-based offloading for large-scale mobile cloud computing," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, IEEE, 2021, pp. 1–10. IEEE. https://doi.org/10.1109/INFOCOM42981.2021.9488821.

[15] S. Ouhame, Y. Hadi, and A. Ullah, "An efficient forecasting approach for resource utilization in cloud data center using CNN-LSTM model," *Neural Comput Appl*, vol. 33, no. 16, pp. 10043–10055, 2021. Springer. https://doi.org/10.1007/s00521-021-05770-9.

[16] J. Bi, H. Ma, H. Yuan, and J. Zhang, "Accurate prediction of workloads and resources with multi-head attention and hybrid LSTM for cloud data centers," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 3, pp. 375–384, 2023. IEEE. https://doi.org/10.1109/TSUSC.2023.3259522.

[17] E. Patel and D. S. Kushwaha, "A hybrid CNN-LSTM model for predicting server load in cloud computing," *J Supercomput*, vol. 78, no. 8, pp. 1–30, 2022. Springer. https://doi.org/10.1007/s11227-021-04234-0.

[18] Y. Chen, Y. Sun, C. Wang, and T. Taleb, "Dynamic task allocation and service migration in edge-cloud IoT system based on deep reinforcement learning," *IEEE Internet Things J*, vol. 9, no. 18, pp. 16742–16757, 2022. IEEE. https://doi.org/10.1109/JIOT.2022.3164441.

[19] T. Kwantwi, G. Sun, N. A. E. Kuadey, G. T. Maale, and G. Liu, "Blockchain-based computing resource trading in autonomous multi-access edge network slicing: A dueling double deep q-learning approach," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2912–2928, 2023. IEEE. https://doi.org/10.1109/TNSM.2023.3240301.

[20] H. Hu, D. Wu, F. Zhou, X. Zhu, R. Q. Hu, and H. Zhu, "Intelligent resource allocation for edge-cloud collaborative networks: A hybrid DDPG-D3QN approach," *IEEE Trans Veh Technol*, vol. 72, no. 8, pp. 10696–10709, 2023. IEEE. https://doi.org/10.1109/TVT.2023.3253905.

[21] M. Faiz and A. K. Daniel, "A multi-criteria cloud selection model based on fuzzy logic technique for QoS," *International Journal of System Assurance Engineering and Management*, vol. 15, no. 2, pp. 687–704, 2024. Springer. https://doi.org/10.1007/s13198-022-01723-0.

[22] A. Kesarwani and P. M. Khilar, "Development of trust-based access control models using fuzzy logic in cloud computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 5, pp. 1958–1967, 2022. Elsevier. https://doi.org/10.1016/j.jksuci.2019.11.001.

[23] S. AlKheder and A. AlOmair, "Urban traffic prediction using metrological data with fuzzy logic, long short-term memory (LSTM), and decision trees (DTs)," *Natural hazards*, vol. 111, no. 2, pp. 1685–1719, 2022. Springer. https://doi.org/10.1007/s11069-021-05112-x.

[24] F. Yalcinkaya and A. Erbas, "Convolutional neural network and fuzzy logic-based hybrid melanoma diagnosis system," *Elektronika ir Elektrotechnika*, vol. 27, no. 2, pp. 55–63, 2021. https://doi.org/10.5755/j02.eie.28843.