Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.063190

ARTICLE

Energy Efficient and Resource Allocation in Cloud Computing Using QT-DNN and Binary Bird Swarm Optimization

Puneet Sharma[1], **Dhirendra Prasad Yadav**[1], **Bhisham Sharma**[2,*], **Surbhi B. Khan**[3,4,*] **and Ahlam Almusharraf**[5]

[1]Department of Computer Engineering & Applications, G.L.A. University, Mathura, 281406, India
[2]Centre for Research Impact and Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, 140401, India
[3]School of Science, Engineering and Environment, University of Salford, Manchester, M5 4WT, UK
[4]Division of Research and Development, Lovely Professional University, Phagwara, 144411, India
[5]Department of Management, College of Business Administration, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh, 11671, Saudi Arabia

*Corresponding Authors: Bhisham Sharma. Email: bhisham.pec@gmail.com; Surbhi B. Khan. Email: surbhibhatia1988@yahoo.com

Received: 08 January 2025; Accepted: 13 May 2025; Published: 29 August 2025

ABSTRACT: The swift expansion of cloud computing has heightened the demand for energy-efficient and high-performance resource allocation solutions across extensive systems. This research presents an innovative hybrid framework that combines a Quantum Tensor-based Deep Neural Network (QT-DNN) with Binary Bird Swarm Optimization (BBSO) to enhance resource allocation while preserving Quality of Service (QoS). In contrast to conventional approaches, the QT-DNN accurately predicts task-resource mappings using tensor-based task representation, significantly minimizing computing overhead. The BBSO allocates resources dynamically, optimizing energy efficiency and task distribution. Experimental results from extensive simulations indicate the efficacy of the suggested strategy; the proposed approach demonstrates the highest level of accuracy, reaching 98.1%. This surpasses the GA-SVM model, which achieves an accuracy of 96.3%, and the ART model, which achieves an accuracy of 95.4%. The proposed method performs better in terms of response time with 1.598 as compared to existing methods Energy-Focused Dynamic Task Scheduling (EFDTS) and Federated Energy-efficient Scheduler for Task Allocation in Large-scale environments (FESTAL) with 2.31 and 2.04, moreover, the proposed method performs better in terms of makespan with 12 as compared to Round Robin (RR) and Recurrent Attention-based Summarization Algorithm (RASA) with 20 and 14. The hybrid method establishes a new standard for sustainable and efficient administration of cloud computing resources by explicitly addressing scalability and real-time performance.

KEYWORDS: Cloud computing; quality of service; virtual machine; allocation; deep neural network

1 Introduction

Cloud computing is a rapidly developing technology that provides services on a pay-as-per-use basis. Optimal RA in cloud computing is crucial for maximizing resource utilization and lowering expenses. Traditional algorithms are more straightforward in implementation and understanding but do not provide optimized results, so metaheuristic algorithms are used to provide optimized results [1]. These algorithms are designed to find a sufficiently satisfactory solution within a certain amount of computation time. The

QoS encompasses multiple variables, including environmental issues, performance, and energy consumption. Metaheuristic algorithms effectively provide optimized results, whereas traditional algorithms cannot provide better results. Integrating a metaheuristic method with machine learning enhances accuracy and yields better results [2]. Three service prototypes are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Infrastructure as a Service (IaaS) provides immediate and flexible access to computational resources. Platform as a Service (PaaS) is a robust framework that enables developers to quickly and easily create applications [3]. Conventional methods for task scheduling are not appropriate for addressing the dynamic nature of cloud environments [4]. The security is improved by the system's ability to identify malicious nodes using deep learning (DL). The availability of resources and potential threats is ensured by implementing DL, which optimizes the utilization of resources and distribution of tasks. Resource allocation optimization considers parameters such as the number of nodes in the layer and connections between nodes [5]. The approach discussed optimized system performance through the dynamic and adaptable allocation of image-based resources. The contribution of the paper is as follows.

(1) The Queuing Theory-based Deep Neural Network (QT-DNN) utilizes the neural network's capacity to comprehend detailed patterns and produce precise forecasts, allowing it to effectively manage difficult resource allocation scenarios.
(2) The integration of BBSO into the resource allocation process improves the quality of service (QoS) by intelligently distributing and allocating resources.
(3) We created a synthetic dataset and performance is compared with several other state-of-the-art methods.

The rest of the manuscript is organized as follows.

In Section 2, we have added a details summary of the previous methods, Section 3 discusses the proposed architecture. Whereas, Section 4 discusses the experimental setup. Further, in Section 5, results and discussion of the performance measures have been elaborated. Finally, in Section 6, the conclusion and future scope of the proposed method have been added.

2 Literature Review

Lee et al. [6] presented a resource management system that integrates an energy-saving strategy. The voltage scaling approach adjusts resources by evaluating the cloud utilization of the central processing unit (CPU). Moreno et al. [7] suggested workload scheduling that significantly improves energy efficiency in cloud data centres. They reduce the negative impact on performance while improving energy efficiency. The Prediction-based resource allocation algorithm (PBRA) was proposed by Alasaad et al. [8]. Their approach seeks to alleviate the difficulties associated with deciding how resources should be distributed. The study introduced a priority scheduling technique for overseeing virtual computers, as reported by Kim et al. [9]. Tasks are scheduled based on the extent of I/O utilization and task priorities. Ming et al. [10] utilized an Ant Colony Optimization algorithm that employed polymorphism to improve the quality of service in cloud computing and enable the dynamic exchange of node information lists. In their study, Agarwal and Jain [11] employed a priority algorithm to enhance task performance by comparing round-robin with first come, first serve (FCFS) Scheduling. Mondal et al. [12] introduced a load-balancing technique. The Stochastic Hill climbing optimization method is used to assign work to servers or virtual machines. Cloud Analyst assesses the efficacy of the technique. Zheng and Wang [13] proposed a Pareto-fruit fly optimization algorithm (PFOA) for RA and task scheduling in a cloud environment. This method also solves issues of distribution of tasks among multiple resources and job scheduling.

Zheng et al. [14] suggested an AI-based methodology for effective resource allocation in dynamic cloud computing settings. It integrates XGBoost and LSTM networks to predict workload patterns utilizing historical data from an extensive cloud architecture, including 1000 servers and 52 million data points. Thilagavathy [15] proposed a model with an accuracy rate of 10%–15% superior to that of the state-of-the-art models. The proposed model reduces the erroneous percentage rate of the average request blocking probability due to traffic load by approximately 9.5%–10.2% in comparison to the forecasts of current models. Alizadeh et al. [16] proposed an autonomous system for resource allocation in Fog-cloud computing infrastructures, utilizing the Clipped Double Deep Q-Learning (CDDQL) algorithm and Particle Swarm Optimization (PSO). Zhang et al. [17] proposed that the judicious distribution of resources is essential in cloud computing. In cloud computing resource allocation, the cloud computing centre possesses finite resources, and users arrive sequentially. Rabaaoui et al. [18] present a mobile agent-based architecture for dynamic resource distribution in cloud computing to minimize the costs of virtual machines and the makespan. By integrating hybrid quantum-classical processing, we suggest compatibility with imminent quantum devices while preserving computing efficiency. In contrast to other studies that lack standardized assessments, our approach is meticulously evaluated against benchmark datasets, facilitating explicit performance comparisons. Ultimately, our methodology incorporates sophisticated error mitigation strategies, diminishing the effects of quantum noise and enhancing model dependability. These developments enable our method to connect theoretical concepts with actual applications while laying the groundwork for scalable and significant QT-DNN applications.

3 Proposed Solution

In the proposed study, we maximize resource allocation and minimize energy consumption in a cloud computing environment. Moreover, the study presents a deep neural network (QT-DNN) based on Queuing Theory. This network is designed to allocate resources in a cloud environment, considering bandwidth and resource load limits. The flowchart illustrating the suggested framework is shown in Fig. 1.

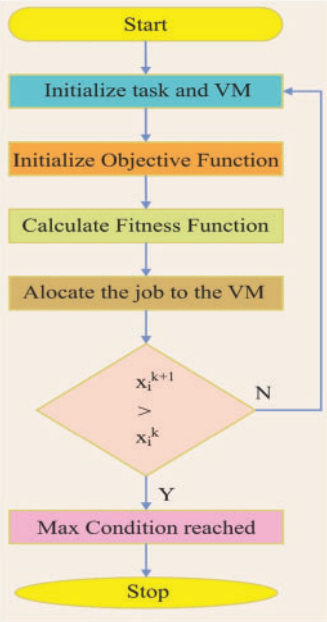

Figure 1: Flowchart of our proposed method

3.1 Resource Allocation

In today's scenario, the efficient allocation of resources is crucial to optimizing system efficiency. The demand for various resources, such as bandwidth and processing power, continues to rise as the digital environment expands. A novel approach has emerged to address this challenge that integrates the principles of Queuing Theory and deep neural networks, as shown in Fig. 2. The Queuing Theory-based deep neural network (QT-DNN) is an advanced system that optimizes resource allocation by considering various design restrictions. Organizations can attain heightened efficiency, optimize resource consumption, and eventually enhance overall performance in various applications and industries by utilizing the capabilities of QT-DNN. The architecture of the proposed method is shown in Fig. 2.

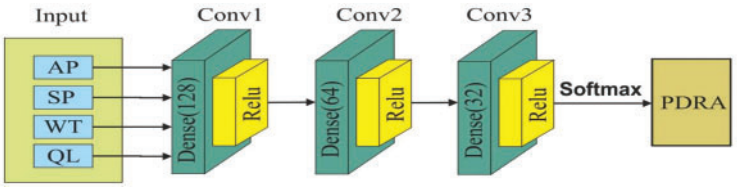

Figure 2: Architecture of the QT-DNN

3.2 Queuing Theory-Based Deep Neural Network (QT-DNN)

In the QT-DNN, we have an input layer, a hidden layer, and an output layer. The input layer consists of input nodes representing the features of each VM. We generated the vector arrival process (AP), service process (SP), queue Length (QL) and Waiting Time (WT) for the input to the hidden layer. Our model has three hidden layers: the first dense layer's size is 128, followed by relu; a second dense layer has a size of 64, followed by relu; and the first dense layer's size is 32, followed by relu. Overall, our model contains 10,976 trainable neurons. After the hidden layer, a softmax layer is applied, which produces a probability distribution representing the percentage of CPU resources to allocate to each VM.

Let $A_i(t)$ represent the arrival rate of tasks to the i-th VM at time t. The arrival process can be modelled as a Poisson process, where the probability of tasks (k) arrivals in time t is given as follows.

$$P\left(N\left(t\right)=k\right)=\frac{\left(A_i t\right)^k e^{-A_i t}}{k!} \tag{1}$$

where $A_i\left(t\right)$ represents the expected number of arrival at time t, $\left(A_i t\right)^k$ represents the number of ways events can happen given the arrival rate, $e^{-A_i t}$ represents the probability of having zero arrivals in time t and k! represents the factorial of k which accounts for the number of ways to arrange task arrivals.

Whereas service process (SP) is denoted by B_i provide service rate at which tasks are processed by the i-th VM. If we assume the service times are exponentially distributed, the probability that a task will be completed in time t which is expressed as follows.

$$P\left(T\le t\right)=1-e^{-B_i t} \tag{2}$$

where B_i represents t service rate of the i-th VM, I represent the index of the VM, t represents the Specific time by which task completion is measured, $P(T \le t)$ reprints the Probability that a task is completed by time t and $e^{-B_i t}$ represents the exponential decay term, representing the probability that the task is still incomplete

by time t. Furthermore, the difference between the arrival rate and the service rate is expressed as follows.

$$L_i(t) = L_i(t-1) + A_i(t) - B_i(t) \tag{3}$$

where $L_i(t)$ represents the Queue length for the i-th VM at time t, t represents Specific time at which the queue length is measured, $L_i(t-1)$ represents Queue length at the previous time step $t-1$, $A_i(t)$ represents the arrival rate and $B_i(t)$ represents the service rate at time t. Moreover, Waiting Time (WT) is expected waiting time for tasks in the queue can be derived from Little's Law which is expressed as follow.

$$w_i = \frac{L_i(t)}{A_i(t)} \tag{4}$$

where W_i represents Expected waiting time for tasks in the queue for the i-th VM, $L_i(t)$ represents Queue length (number of tasks waiting) for the i-th VM at time t and $A_i(t)$ represents Arrival rate of tasks to the i-th VM at time t. We applied Objective Function of Queuing Metrics to minimize the total waiting time W across all VMs which is given as follows.

Minimize

$$\sum_{i=1}^{n} W_i \sum_{i=1}^{n} \frac{L_i(t)}{A_i(t)} \tag{5}$$

where $\sum_{i=1}^{n} w_i$ represents the total expected waiting time across all VMs, which we aim to minimize, W_i represents Expected waiting time for the i-th VM, $\sum_{i=1}^{n} \frac{L_i(t)}{A_i(t)}$ represents expanded form of the total waiting time, showing how queue length and arrival rate for each VM contribute, I represents Index of the VM, from 1 to n and n represents total number of VMs in the system. $L_i(t)$ represents Queue length at time t for the i-th VM and $A_i(t)$ represents Arrival rate at time t for the i-th VM. We define the input vector $v_i(t)$ for each VM at time t as shown as follows.

$$v_i(t) = [Q_i(t), A_i(t), B_i(t), T_i(t)] \tag{6}$$

where $Q_i(t)$ represents queue length, $A_i(t)$ represents arrival rate, $B_i(t)$ represents service process, $T_i(t)$ represents waiting time. The input vector is passed to the hidden Layer 1; in which we apply a linear transformation followed by a non-linear ReLU activation is shown as follows.

$$H_1 = \mathrm{ReLU}(W_1 v_i(t) + b_1) \tag{7}$$

where H_1 represents the Output of the first hidden layer after applying ReLU, W_1 represents the Weight matrix for the first hidden layer, $v_i(t)$ represents the Input vector for the i-th data point at time t, b_1 represents the Bias vector added to the linear transformation, $W_1 v_i(t) + b_1$ represents the Linear transformation of the input vector and ReLU is the Activation function that introduces non-linearity. Furthermore, the feature extracted from hidden layer 1 is passed to the hidden layer 2 for substituent transformation as follows.

$$H_2 = \mathrm{ReLU}(W_2 H_1 + b_2) \tag{8}$$

The feature extracted from the second hidden layer is passed to the third hidden layer as follows.

$$H_3 = \mathrm{ReLU}(W_3 H_2 + b_3) \tag{9}$$

On the top of the model, we applied Softmax layer to produce a probability distribution over the possible resource allocation strategies as expressed as follows.

$$P_{out} = Softmax\left(W_{out}H_3 + b_{out}\right) \tag{10}$$

where weight matrix is represented by W_{out} and probability of resource allocation is represented by P_{out} and b_{out} = Bias of the output layer.

3.3 Binary Bird Swarm Optimization (BBSO) Model

The proposed approach is influenced by the social dynamics observed in bird flocks, particularly their behaviours during the search for food patches. It incorporates three main imitative behaviours: foraging, vigilance, and flying. The five criteria in BBSO encapsulate and replicate the behaviours demonstrated by birds, rendering it a very efficient and successful optimization technique. The first rule of the suggested technique is that each bird can engage in two actions: foraging or retaining vigilance. This is depicted as a stochastic process. In the given situation, when virtual machines (VMs) are overloaded, workloads are taken away from them and redistributed to underloaded VMs, depending on the availability of resources. Conversely, remaining vigilant might be compared to particles actively searching for their optimal places. If a randomly generated number, uniformly distributed between 0 and 1, is less than a given constant value P, where P is between 0 and 1, then the bird will participate in foraging for food. Otherwise, it will persist in maintaining vigilance. The jobs continuously adjust their placements based on their accumulated knowledge of the food locations (destination VMs). Similarly, particles adjust their places within the issue environment, using their optimal previous positions (p_best) and aligning with the corresponding virtual machines (VMs). The processes are derived using Eq. (11), while the initial positions are computed using Eq. (12).

$$X_i^{k+1} = X_i^k + c \times rand_1 \times \left(P_{best_i} - x_i^k\right) + s \times rand_2 \times \left(gbest - X_i^k\right) \tag{11}$$

$$X_0^k = X_{min} + \left(X_{max} - X_{min}\right) \times rand \tag{12}$$

here, x_o^k represents the initial position, X_{min} represents the minimum value (0.4), X_{max} represents the maximum value (4.0), x_i^k represents current position at iteration k, x_i^{k+1} represents the position vector at iteration k, p_{best_i} represents the best position, gbest represents the global best position. As per rule three, during the vigilance phase, each bird in the flock competes with others to approach the centre rather than directly heading there. In the problem context, this behaviour of keeping vigilance representing tasks with the least positions competing to move closer to the forager with the best position. The correlation is mathematically represented by following (13)–(15) equations, respectively.

$$X_i^{k+1} = X_i^k A_1(p_{best_j} - X_i^k) \times rand_a + A_2(p_{best_i} - X_i^{k)} \times rand_b \tag{13}$$

$$A_1 = a_1 \times \exp\left(-\frac{p_{best_i}}{p_{best_j}} \times N\right) \tag{14}$$

$$A_2 = a_2 \times \exp\left(\left(-\frac{p_{best_i}}{p_{best_j}} \times N\right)\frac{N \times P_{best_i}}{P_{best_j} + e}\right) \tag{15}$$

The variables in the equation are defined as follows: p_{best_j} represents the best position of the i-th particle (chosen from particle 1 to N), rand a is a random number between 0 and 1, rand is a random number between 1 and 1, A1 and A2 represent the indirect and direct effects of the surroundings, and a1 is a constant with a positive value. In this context, the bird that possesses the highest fitness value is referred to as the producer, whereas the bird with the lowest fitness value is known as the scrounger. The pseudocode for the Binary Bird

Swarm Optimization algorithm. Table 1 depicts the pseudo-code for the binary bird swarm optimization algorithm, which presents the particle size, particle positions, p_{best} and g_{best} values, and then find out the best-known solution. This solution represents the optimal allocation of resources that maximizes QoS.

Table 1: Pseudo code for binary bird swarm optimization algorithm

Pseudo Code for BBSOA

Input: Initialize population (X), binary variables, pbest, and gbest
Maximum number of iterations (i = 350)
for each iteration, i = 1 to maximum iterations
Update the velocity (d) and position of the particle
for each dimension
Generate a random number r1 and r2
Update velocity
$d = w \times [d] + c_1 \times r_1 \times \left(P_{best}[d] - X_i^k[d]\right) + c_2 \times r_2 \times \left(gbest - X_i^k[d]\right)$
Update position:
$X_i^k[d]$ = threshold (1/(1 + exp(−velocity[d])))
Binary threshold function
Calculate the fitness of the current particle's position (X_i^k)
if X_i^k > pbest, X_i^k > gbest
Update the particle's pbest, *gbest* to the X_i^k
else
Repeat the Process until the stopping criterion met
end if
End

4 Experimental Setup

A synthetic dataset for cloud resource allocation often comprises several structured files, encompassing task descriptions, virtual machine (VM) requirements, task to VM mappings, and performance indicators. The dataset size fluctuates according to the simulation scale, spanning from 1000 to 100,000 activities and 50 to 2000 virtual machines. Each task file encompasses attributes such job length (in Million Instructions), CPU and RAM specifications, task priority, and arrival time, whereas the VM file comprises processing power (in MIPS), available memory, storage capacity, and network bandwidth. Task allocation files record scheduling decisions, assigning each task to a suitable VM, ensuring a minimum of one mapping per task, with bigger datasets surpassing 50,000 mappings for distributed workloads. Execution logs monitor essential performance indicators such as execution duration, resource usage, SLA breaches, and energy expenditure. Virtual machines are classified into small (1000–4000 MIPS), medium (4000–10,000 MIPS), and large (10,000–50,000 MIPS) categories, with RAM capacities spanning from 2 to 128 GB and storage options ranging from 20 GB to 2 TB. The dataset size may vary from 10 MB to more than 5 GB, contingent upon the detail level in execution logs. The experimental settings are shown in Table 2.

Table 2: Simulation system configuration

Components	Version
MATLAB	R2021a
OS	Windows 10
Memory	8 GB DDR3
CPU	i5 processor @ 3.5 GHz

5 Results and Discussion

This section presents the results based on RA approaches in cloud computing environments. Through extensive experimentation, the performance metrics, including task execution time, makespan, resource utilization, and energy consumption, are analyzed and compared with existing methods. Fig. 3 shows the resource utilization of six algorithms in different task counts from 1 to 5, respectively.

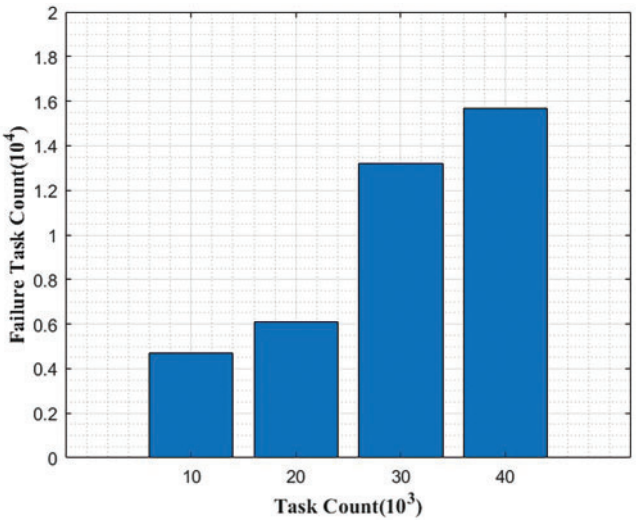

Figure 3: Actual and predicted failure with task count

Fig. 3 shows that as the task count increases (from 1 to 5), the resource utilization percentage also shows an increasing trend. For instance, for Task Count 1, the resource utilization is 79%, while for Task Count 5, it has risen to 88.1%. This upward trend in resource utilization suggests that as more tasks are performed, a higher proportion of available resources is being utilized, indicating better resource efficiency and utilization in handling multiple tasks.

5.1 Comparative Study

This section shows the comparative analysis in terms of Energy Consumption Analysis in various task counts of various algorithms such as energy efficient dynamic task scheduling (EFDTS), quick task completion strategy (QTCS) and power efficient fault-tolerant scheduling (PEFS). Moreover, various performance metrics, such as accuracy, precision, recall, and f-measure, are also considered. In addition, the response time for EFDTS and FESTAL is considered, and the makespan time of RR and RASA is also considered for various methods. Performance measures of Genetic algorithm-support vector machine (GA_SVM) and Adaptive

resonance theory (ART) are also measured. Fig. 4 shows energy consumption analysis in various task counts. The values in the table are in energy consumption units, and each cell corresponds to the energy consumed by a particular task under a specific type. Upon analyzing the data, it can be observed that energy consumption varies slightly across different tasks and types. The energy consumption values remain relatively consistent, with only minor fluctuations. For instance, for Task Count 5, the energy consumption ranges from 0.1032 to 0.1042 units across the six types. As the task count increases (from 5 to 25), there are subtle changes in the energy consumption values for each type, but the differences are not substantial. This suggests that the energy consumption pattern remains relatively stable regardless of the task count. Table 3 displays the performance metrics for three distinct models, including GA-SVM, ART, and the proposed method.

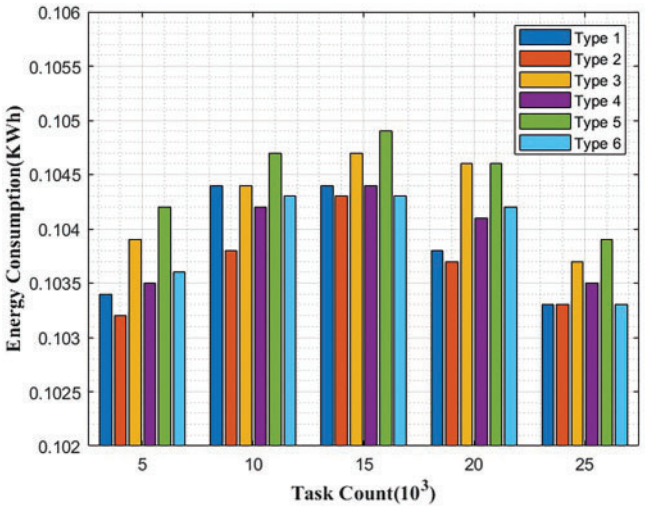

Figure 4: Energy consumption analysis in various task counts

Table 3: Performance comparison with other methods

Performance metric	GA-SVM	ART	Proposed
Accuracy	96.31	95.45	98.12
Precision	92.32	94.26	97.35
Recall	94.24	95.61	97.27
F-Measure	96.23	95.35	97.49

The proposed approach demonstrates the highest level of accuracy, reaching 98.1%. This surpasses both the GA-SVM model, which achieves an accuracy of 96.3%, and the ART model, which achieves an accuracy of 95.4%. The superior accuracy of the suggested model indicates its greater efficacy in accurately categorizing situations in comparison to the other two models. Moreover, GA-SVM provides 92.3 precision value, ART gives 94.2 and our proposed method gives 97.3 precision value. Table 4 presents an examination comparing energy consumption.

Table 4: Energy composition comparison with other methods

Energy consumption				
Task	**EFDTS**	**QTCS**	**PEFS**	**Proposed**
1	0.62	2.00	1.95	0.58
2	0.615	1.64	1.70	0.54
3	0.613	1.98	1.62	0.52
4	0.612	2.40	1.81	0.519
5	0.605	2.91	2.01	0.506

The values indicate the amount of energy consumed by each strategy when performing the corresponding task. Smaller values indicate superior energy efficiency. After examining the data, it is clear that the proposed method consistently has the lowest energy use for all tasks when compared to the other ways. It surpasses EFDTS, QTCS, and PEFS, attaining notably lower energy consumption numbers. In job 1, the suggested technique utilizes 0.58 units of energy, whereas EFDTS utilizes 0.62 units, QTCS utilizes 2 units, and PEFS utilizes 1.95 units. The tendency persists in all jobs, with the proposed technique consistently exhibiting greater energy efficiency in each instance. Task 5 demonstrates a notable disparity, as the proposed technique utilizes only 0.506 units of energy, while QTCS utilizes 2.91 units, PEFS utilizes 2.01 units, and EFDTS utilizes 0.605 units. Table 5 shows task response time comparison with existing methods.

Table 5: Response time comparison with other methods

Metric	**EFDTS**	**FESTAL**	**Proposed**
Response time	2.31	2.04	1.598

The response time represents the time taken to complete a task, and the task count represents the number of tasks being processed. Response time for the proposed technique is consistently lower compared to EFDTS and FESTAL across different task counts. This indicates that the proposed technique is more efficient in completing tasks in a shorter amount of time, leading to better performance. On the other hand, the proposed technique maintains relatively stable and low response times even with an increasing number of tasks, showcasing its ability to efficiently manage high workloads and maintain performance. Table 6 shows makespan comparison results with existing methods.

Table 6: Makespan comparison with other methods

Techniques	**RR**	**RASA**	**Proposed**
Makespan	20	14	12

It depends on the scheduling problem or project at hand, in a simple scenario, where tasks are scheduled sequentially (no overlap), sum the durations of all tasks to find the makespan. However, in more complex scheduling problems, such as task scheduling or project networks, the calculations can be more intricate.

Fig. 5 illustrates the energy consumption comparison analysis. The values represent the energy consumed by each technique for executing the respective task. Lower values indicate better energy efficiency.

Upon analyzing the data, it is evident that the proposed method consistently exhibits the lowest energy consumption for all tasks compared to the other techniques. It outperforms EFDTS, QTCS, and PEFS, achieving significantly lower energy consumption values. For instance, in task 1, the proposed method consumes only 0.58 units of energy, while EFDTS consumes 0.62, QTCS consumes 2, and PEFS consumes 1.95 units. The trend continues across all tasks, with the proposed method demonstrating superior energy efficiency in each case. Task 5 showcases the most significant difference, where the proposed technique consumes only 0.506 units of energy, whereas QTCS consumes 2.91, PEFS consumes 2.01, and EFDTS consumes 0.605. These results indicate that the proposed method is highly effective in optimizing energy consumption and can be a valuable approach for enhancing energy efficiency in various tasks within the context of this study.

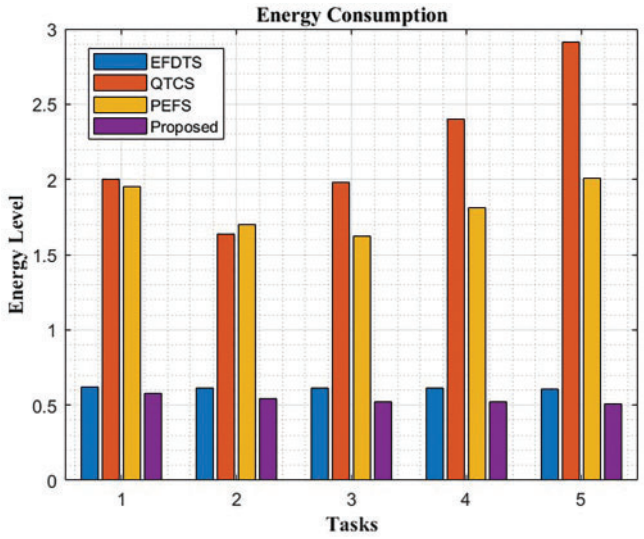

Figure 5: Energy consumption comparison analysis

5.2 Ablation Study

This study presents the components affecting model performance, as shown in Fig. 6. Fig. 6a shows a response time comparison analysis with existing methods. This indicates that the proposed technique is more efficient in completing tasks in a shorter amount of time, leading to better performance. EFDTS and FESTAL show higher response times as the task count increases, suggesting that these techniques may struggle to handle larger workloads effectively. On the other hand, the proposed technique maintains relatively stable and low response times even with increasing tasks, showcasing its ability to manage high workloads and maintain performance efficiently. Fig. 6b shows makespan comparison results with existing methods such as RR and RASA algorithms. It depends on the scheduling problem or project at hand; in a simple scenario, where tasks are scheduled sequentially (no overlap), sum the durations of all tasks to find the makespan. However, the calculations can be more intricate in more complex scheduling problems, such as task scheduling or project networks. According to the above figure, the proposed approach gives better results for load balancing among nodes.

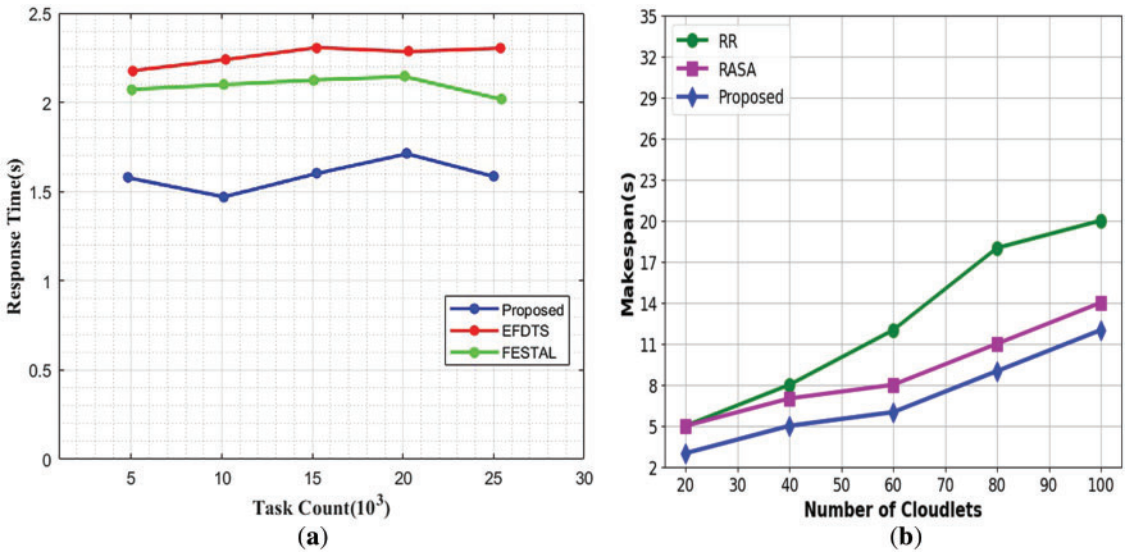

Figure 6: Effect of different components (**a**) Response Time and (**b**) Makespan

In the proposed method, we designed a three-layer deep learning model to extract high-dimension features of the different types of files. Our model segregates different file types more efficiently than other SOTA methods. Furthermore, we optimize the resource allocations on different VMs using a nature-inspired Binary Bird Swarm Optimization algorithm. The efficacy of the current model may be contingent upon the dataset's properties, and its applicability to real-world, noisy, or significantly imbalanced datasets requires further evaluation. Moreover, the computational complexity arising from deeper layers or a higher number of neurons may provide scalability issues in resource-limited settings. Future research may investigate techniques such as model pruning, adaptive learning rate scheduling, or hybrid architectures to enhance efficiency while maintaining accuracy. Presenting these insights will enhance the discourse by delivering a balanced perspective on the advantages and opportunities for improvement in the suggested methodology.

We summarize the performance evaluation on large cloud environment where we take QoS parameters execution time, SLA, energy consumption and utilization of VM. There is an improvement in energy consumption up to 20%, moreover, the better utilization of VM is also possible, however, the execution time is much faster. Table 7 shows the performance evaluation on large cloud environment.

Table 7: Performance evaluation on large cloud environment

QoS	(Without QT-DNN + BBSO)	(With QT-DNN + BBSO)	Improvement
Execution time	1.2 s per task	0.84 s per task	**Faster**
SLA	Delayed by 20%	Delayed by 5%	**Reduction by 60%**
Energy consumption	500 kW	350 kW	**Saving up to 20%**
Utilization of VM	Either idle or overloaded	Balanced Load	**Better resource utilization**

We summarize the expected energy saving in the real word clod infrastructure in Table 8. There is an energy saving of 15%–20% in inter cluster overhead, moreover, in dynamic scaling, there is a reduction of 25%/30% in idle power consumption, however, in hardware aware optimization, there is a reduction of 10%–15% through the effective and efficient use of hardware.

Table 8: Expected energy savings by the model

Optimization level	Expected energy savings
Cluster-based segmentation	15%–20% reduction in inter-cluster overhead.
Dynamic scaling	25%–30% reduction in idle power consumption.
Hardware-aware optimization	10%–15% reduction through efficient hardware usage.
Federated optimization	20%–25% reduction in global energy wastage.

6 Conclusion

This research has explored innovative techniques and methodologies to enhance the efficiency and RA. In this study, we introduced a Queuing Theory-based deep neural network (QT-DNN) for efficient resource allocation in the cloud, considering various design constraints such as bandwidth and resource load. By optimizing the overall resource management process through Binary bird swarm optimization (BBSO), this model aims to enhance the QoS for cloud users. The result analysis of the proposed model was simulated using MATLAB software, demonstrating impressive outcomes. The resource allocation accuracy gain achieved in this study reached 98.1%. Furthermore, the research achieved a remarkable reduction in energy consumption, with a value of 0.506 attained. This reduction in energy consumption is paramount in optimizing resource allocation, contributing to the sustainability and cost-effectiveness of cloud computing solutions. This research opens up new avenues for future studies. The introduced methodologies and techniques can be a foundation for further advancements and innovations. In future studies, a transformer-based model will be utilized to enhance the feature map. In addition, another nature-inspired algorithm can be utilized to optimize resource allocation.

Acknowledgement: The authors extend their heartfelt gratitude to the Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2025R432), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Funding Statement: The authors extend their heartfelt gratitude to the Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2025R432), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Author Contributions: Conceptualization: Puneet Sharma, Dhirendra Prasad Yadav, Bhisham Sharma; Data curation: Dhirendra Prasad Yadav, Bhisham Sharma; Formal analysis: Surbhi B. Khan, Ahlam Almusharraf; Investigation: Puneet Sharma, Bhisham Sharma; Methodology: Dhirendra Prasad Yadav, Puneet Sharma, Bhisham Sharma; Project administration: Surbhi B. Khan, Ahlam Almusharraf; Resources: Surbhi B. Khan, Ahlam Almusharraf; Software: Puneet Sharma, Bhisham Sharma; Visualization: Puneet Sharma, Bhisham Sharma; Writing—original draft: Puneet Sharma, Dhirendra Prasad Yadav; Writing—review & editing: Bhisham Sharma, Surbhi B. Khan, Ahlam Almusharraf. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Provided by the first author on request basis.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

Abbreviations

RA	Resource Allocation
QoS	Quality of Service

QT-DNN	Queuing Theory-Based Deep Neural Network
BBSO	Binary Bird Swarm Optimization
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service

References

1. Ning J, Zhang C, Sun P, Feng Y. Comparative study of ant colony algorithms for multi-objective optimization. Information. 2018;10(1):11. doi:10.3390/info10010011.
2. Razaque A, Vennapusa NR, Soni N, Janapati GS. Task scheduling in cloud computing. In: 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT); 2016 Apr 29; Farmingdale, NY, USA. p. 1–5.
3. Lynn T, Xiong H, Dong D, Momani B, Gravvanis G, Filelis-Papadopoulos C, et al. Cloudlightning: a framework for a self-organising and self-managing heterogeneous cloud. In: Proceedings of the 6th International Conference on Cloud Computing and Services Science; 2016 Apr 23–25; Rome, Italy.
4. Sun G, Liao D, Zhao D, Xu Z, Yu H. Live migration for multiple correlated virtual machines in cloud-based data centers. IEEE Trans Serv Comput. 2015;11(2):279–91. doi:10.1109/tsc.2015.2477825.
5. Xia JY, Li S, Huang JJ, Yang Z, Jaimoukha IM, Gündüz D. Metalearning-based alternating minimization algorithm for nonconvex optimization. IEEE Trans Neural Netw Learn Syst. 2022;34(9):5366–80. doi:10.1109/tnnls.2022.3165627.
6. Lee LT, Liu KY, Huang HY, Tseng CY. A dynamic resource management with energy saving mechanism for supporting cloud computing. Int J Grid Distrib Comput. 2013;6(1):67–76.
7. Moreno IS, Yang R, Xu J, Wo T. Improved energy-efficiency in cloud datacenters with interference-aware virtual machine placement. In: 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS); 2013 Mar 6–8; Mexico City, Mexico. p. 1–8.
8. Alasaad A, Shafiee K, Behairy HM, Leung VC. Innovative schemes for resource allocation in the cloud for media streaming applications. IEEE Trans Parallel Distrib Syst. 2014;26(4):1021–33. doi:10.1109/tpds.2014.2316827.
9. Kim D, Kim H, Jeon M, Seo E, Lee J. Guest-aware priority-based virtual machine scheduling for highly consolidated server. In: Euro-Par 2008—Parallel Processing: 14th International Euro-Par Conference; 2008 Aug 26–29; Las Palmas de Gran Canaria, Spain. Berlin/Heidelberg, Germany: Springer; 2008. p. 285–94.
10. Wei M, Zhang CY, Qiu F, Cui Y, Sui QQ, Ding WB. Resources allocation method on cloud computing. In: 2014 International Conference on Service Sciences; 2014 May 22–23; Wuxi, China. p. 199–201.
11. Agarwal DA, Jain S. Efficient optimal algorithm of task scheduling in cloud computing environment. arXiv:1404.2076. 2014.
12. Mondal B, Dasgupta K, Dutta P. Load balancing in cloud computing using stochastic hill climbing—a soft computing approach. Procedia Technol. 2012;4:783–9. doi:10.1016/j.protcy.2012.05.128.
13. Zheng XL, Wang L. A pareto based fruit fly optimization algorithm for task scheduling and resource allocation in cloud computing environment. In: 2016 IEEE Congress on Evolutionary Computation (CEC); 2016 Jul 24–29; Vancouver, BC, Canada. p. 3393–400. doi:10.1109/cec.2016.7744219.
14. Zheng H, Xu K, Zhang M, Tan H, Li H. Efficient resource allocation in cloud computing environments using AI-driven predictive analytics. Appl Comput Eng. 2024;82:17–23.
15. Thilagavathy C. Leveraging machine and deep learning models for load balancing strategies in cloud computing. Indian J Sci Technol. 2024;17(45):4722–31. doi:10.17485/ijst/v17i45.2728.
16. Alizadeh Javaheri SD, Ghaemi R, Monshizadeh Naeen H. An autonomous architecture based on reinforcement deep neural network for resource allocation in cloud computing. Computing. 2024;106(2):371–403. doi:10.1007/s00607-023-01220-7.

17. Zhang Y, Liu B, Gong Y, Huang J, Xu J, Wan W. Application of machine learning optimization in cloud computing resource scheduling and management. In: Proceedings of the 5th International Conference on Computer Information and Big Data Applications; 2024 Apr 26–28; Wuhan, China. p. 171–5.
18. Rabaaoui S, Hachicha H, Zagrouba E. Mobile agents-based framework for dynamic resource allocation in cloud computing. In: ICAART 2024—16th International Conference on Agents and Artificial Intelligence; 2024 Feb 24–26; Rome, Italy. p. 766–73.