ISSN: 2182-2069 / E-ISSN: 2182-2077

Hybrid Scaling in Cloud Computing for Optimizing Resource Utilization: An LSTM-Enhanced Actor-Critic Learning Model

P.S. Shruthi[1*], and Dr.D.R. Umesh[2]

[1*]Vidyavardhaka College of Engineering, Mysuru, Karnataka, India. shruthips@vvce.ac.in, https://orcid.org/0009-0000-6002-4838

[2]People's Education Society College of Engineering, Mandya, Karnataka, India. drumesh@pesce.ac.in, https://orcid.org/0000-0002-6813-3876

Received: December 25, 2024; Revised: January 28, 2025; Accepted: February 10, 2025; Published: February 28, 2025

Abstract

An innovative approach for cloud Autoscaling for optimal resource utilization is proposed by integrating an LSTM-enhanced Actor-Critic model. Out model effectively captures temporal variations in resource demand, leading to more accurate and responsive scaling by leveraging Long Short-Term Memory (LSTM) networks within the Actor-Critic framework. Without compromising system performance, a dynamic pricing mechanism is introduced that balances on-demand and spot instance costs which ensures optimal budget utilization. The experimental results demonstrate the significant improvement in resource utilization and system throughput by introducing the proposed hybrid scaling strategy combining vertical and horizontal scaling. The proactive scaling, reducing SLA violations, and enhancing overall system reliability is enabled by predictive capabilities of the LSTM model. The advantages of the proposed model over the static threshold based autoscaling methods is highlighted by the key performance metrics such as resource utilization, throughput, SLA compliance, and cost efficiency. Our model achieves an average resource utilization of 92.7% while reducing SLA violations by 63% is validated by the experiments demonstrates a substantial improvement over conventional technique. While maintain system performance and SLA compliance the incorporation of pricing mechanism optimizes cost management leads to a 12% reduction in the total operational expenses. To deliver a scalable, cost-effective, and high-performance cloud autoscaling solution the experimental findings outperformance the potential of combining LSTM-driven reinforcement learning to implement hybrid scaling combining with pricing strategy.

Keywords: Hybrid Resource Scaling, Cloud Computing, Deep Learning Techniques, LSTM, Experience Reply, Actor-Critic Model.

1 Introduction

Cloud computing is a transformative model that allows users to access and manage data, applications, and computing resources via the Internet, providing a scalable and flexible alternative to traditional on-premises infrastructure. This approach enables organizations and individuals to utilize computing power on demand, paying only for the resources they use (Rattihalli et al., 2019). Key benefits of this model include resource pooling, rapid scalability, and widespread network accessibility, which have driven its widespread adoption across various industries. Public cloud providers like Amazon Web Service

Journal of Internet Services and Information Security (JISIS), volume: 15, number: 1 (February), pp. 468-485.
DOI: *10.58346/JISIS.2025.I1.031*
*Corresponding author: Vidyavardhaka College of Engineering, Mysuru, Karnataka, India.

(AWS), Microsoft Azure, and Google Cloud, along with private and hybrid cloud solutions, cater to diverse requirements by offering both shared and dedicated resources (Manzoor et al., 2020).

Meeting the growing demand for cloud services presents distinct challenges. As more organizations transition their workloads to the cloud, the need for seamless scalability, high availability, and enhanced performance becomes increasingly critical. Ensuring that resources remain responsive and accessible amid fluctuating demand requires advanced resource management, efficient load balancing, and a resilient infrastructure (Shukur et al., 2020). Additionally, cloud service providers must tackle issues such as data security, latency, energy efficiency, and regulatory compliance to uphold service quality and reliability. Technologies like autoscaling, virtualization, and containerization enable dynamic resource allocation, but these solutions must continuously evolve to address the increasing complexity of cloud environments (Radhika & Sadasivam, 2021). As cloud computing becomes integral to domains such as artificial intelligence, big data, and the Internet of Things (IoT), the demand for robust, efficient, and secure cloud services will continue to grow (Jagan, 2024 ; Alnumay, 2024 ; Nithyalakshmi et al., 2021), Overcoming these challenges is essential for cloud providers to foster innovation and meet the needs of modern users who depend on these services for mission critical operations and data-driven decision-making (Kumar et al., 2020).

Cloud computing providers offer a range of services at fixed rates over time, enabling users to rent Central Processing Unit (CPU) cycles to access resources and execute operations. This model has rapidly gained global popularity due to its scalability and efficiency. By abstracting system resources, cloud computing simplifies the intricate relationship between software and hardware, allowing multiple virtual machines (VMs) to run concurrently while utilizing CPU power, network connectivity, server resources, and bandwidth (Shi & Lin, 2022; Ebadi et al., 2024) At its core, cloud computing interconnects multiple computers to establish a virtualized environment managed by software (Aboorva et al., 2023). A key component of this process is hypervisor technology, which enables multiple operating systems to function simultaneously on a single machine, ensuring efficient resource scaling. Additionally, cloud computing oversees the delivery, allocation, and presentation of these integrated resources to end-users (Janmohammadi & Babazade, 2015). With vast amounts of data and applications stored in the cloud, this model offers greater flexibility and advantages over traditional service providers, improving accessibility and operational efficiency (Chen et al., 2021).

The quality of service offered by cloud servers is directly linked to the performance of data centers. Optimizing resource utilization plays a crucial role in enhancing cloud server efficiency. Cloud data center resources, such as CPU, memory, network bandwidth, and storage, are shared across various users and web applications. Scaling data centers is a primary objective in cloud computing to optimize resource usage (Rahim, 2024). This method is a critical technology that enables the delivery of scalable, on-demand resources to cloud applications (Xu & Zhao, 2022; Rahim, 2024; Joshi et al., 2024). Through virtual machine (VM) adjustments, cloud applications can access shared physical resources. However, allocating physical resources to virtual machines based solely on peak user demand can lead to inefficient resource use. Excessive resource allocation reduces profitability, while insufficient resources fail to meet user expectations.

The solution to this challenge lies in scaling resources based on user requests without administrative intervention. Cloud providers strive to meet user demands by offering a shared resource infrastructure (Daraje & Shaikh, 2021). Cloud applications, referred to as tenants, can experience sudden spikes in demand at specific times, days, or even weeks. To handle these fluctuating loads effectively, dynamic scaling is essential, allowing for scalability without service interruptions. Based on user demand, virtual machines can be seamlessly added, managed, configured, and deployed, which is called horizontal

scaling. The resources in the VM machine, such as CPU cycles and memory, can be added, removed, and configured to meet the workload demand to a certain extent; this is called vertical scaling. In the case of adding and removing new VMs, some amount of time has elapsed, and horizontal scaling can be achieved within the capacity of the physical resource (Saxena & Singh, 2021). When multiple virtual machines are requested, automated tools and methods for managing them become necessary to streamline cloud service processes. A significant challenge for cloud users is efficiently setting up and configuring multiple virtual machines (VMs) (horizontal scaling) and resources of specific VMs (vertical scaling) for specific tasks.

The paper is organized as follows. The related works are discussed in Section 2. In section 3, we present the problem to be solved and the proposed methods to solve the problem. The experimental results are discussed in Section 4. Section 5 comes up with the conclusion of the paper.

2 Related Works

In the work (Garí et al., 2024), the author proposed a proactive autoscaling and energy-efficient VM allocation framework that leverages an online multi-resource neural network to optimize resource utilization in cloud data centres (Shiraishi et al., 2011). This framework addresses the challenges of fluctuating resource demands by accurately predicting resource requirements and consolidating workloads onto fewer energy-efficient physical machines (PMs). The integration of machine learning techniques enhances the framework's ability to adapt dynamically to changing demands, leading to significant energy savings and improved performance. VMs are scaled according to predicted resource demands, enabling efficient resource allocation and reducing waste. The framework optimizes VM placement by consolidating them onto the most energy-efficient PMs, resulting in up to 88.5% power savings (Rahim, 2024).

Pavlenko et al., (2024) conducted an evaluation of Q-Learning and SARSA for online reinforcement Learning (RL) based cloud autoscaling in scientific workflows, highlighting notable differences in their performance. Although both algorithms aim to optimize resource allocation, SARSA consistently outperforms Q-learning, particularly in minimizing execution time and the costs associated with virtual machine scaling. This analysis plays a key role in developing effective autoscaling strategies for cloud environments. State-Action-Reward-State-Action (SARSA) demonstrated superior efficiency, achieving performance gains of up to 40.8 % in certain workflows during the initial episodes, whereas Q-Learning exhibited less consistent outcomes. Additionally, the evaluation revealed that SARSA maintained losses below 6% throughout episodes, underscoring its reliability in learning that SARSA maintained losses below 6% throughout episodes, underscoring its reliability in learning optimal scaling policies.

The author of the work (Jang et al., 2020) discusses the Vertical Autoscaling Simulator Toolkit (VASIM) is a versatile tool designed to support the testing and evaluation of autoscaling algorithms in cloud environments. It simplifies the complexities of vertical autoscaling by simulating key components of autoscaler architectures, including controllers and metrics collectors, while allowing for customizable parameter adjustments. This toolkit is especially useful for analyzing CPU utilization in virtual machines (VMs) and Kubernetes pods, accelerating the development and optimization of autoscaling strategies. VASIM facilitates vertical scaling, which dynamically adjusts resource allocation within a single VM, enhancing both efficiency and performance.

Kang et al., (2013) The vertical autoscaling to GPU resources for machine learning in cloud environments is a growing research focus aimed at optimizing resource utilization while controlling costs. This approach employs algorithms that dynamically adjust GPU resources based on workload demands, improving performance for machine learning applications. A DRL-based method has been introduced to address the straggler problem in federated learning, optimizing GPU scaling by balancing resource costs and learning speed. Jily, an autoscaling scheduler, effectively minimizes costs while maintaining latency constraints, achieving a 28% reduction in average costs compared to conventional methods.

Agarwal et al., (2024) In a hybrid cloud environment, adopting an SLA-driven VM autoscaling approach is essential for maintaining both performance and resource efficiency. This method combines proactive and reactive strategies to adapt to changing workloads while ensuring SLA compliance. Efficient load balancing between public and private clouds plays a critical role in this process. Solutions such as the Hybrid Cloud Autoscaling (HCA) operator help achieve this by continuously monitoring metrics and dynamically adjusting resource allocation to meet SLA requirements.

Khaleq & Ra, (2021) Utilizing Deep Recurrent Reinforcement Learning (DRRL) for intelligent autoscaling of serverless functions effectively tackles the challenges of fluctuating workloads. By incorporating neural networks, particularly Long Short-Term Memory (LSTM) units, DRRL enhances the adaptability of scaling strategies in partially observable environments. Research has shown that DRRL methods improve throughput by 18% and enhance function execution efficiency by 13% compared to conventional threshold-based approaches. By learning from real-time data, DRRL dynamically adjusts scaling decisions based on current workload demands, leading to improved overall system performance.

Marie-Magdelaine & Ahmed, (2020) Implementing Deep Reinforcement Learning (DRL) for intelligent autoscaling in serverless computing effectively tackles challenges related to unpredictable workloads and resource allocation. This method enhances system performance while optimizing resource utilization, especially for latency-sensitive applications. DRL algorithms dynamically adjust resource allocation in response to real-time workload fluctuations, leading to significant improvements in response time and cost reductions of up to 34%. Research indicates that DRL surpasses traditional heuristic approaches, achieving up to a 50% decrease in total function request delays.

Shruthi & Shruthi, (2024) Incorporating Deep reinforcement Learning (DRL) into the autoscaling of serverless functions offers a promising strategy for enhancing performance and optimizing resource usage. This method addresses challenges such as cold-start delays and resource wastage, enhancing both application responsiveness and cost efficiency. Multi-agent DRL solutions enable both horizontal and vertical scaling, adapting to varying function and system requirements. Implementing DRL can lead to significant reductions in application latency (up to 23%) and request failures (up to 34%).

3 Proposed Work

System Model

The design of the problem to be solved is as follows. Let us assume that there are n number of nodes with m resources in each node. The pool of resources is denoted by R_n^m {n=1,2, 3, N and m=1,2, 3,.M}, collaboratively providing the service to the user of the cloud. Where R_n^m indicates the total amount of m resources present in each node n, N is the number of nodes, and M is the number of resources in the cloud platform. Efficiently balancing resource usage can significantly enhance resource utilization.

Additionally, maintaining workload balance across various resource dimensions serves as an indicator of the effectiveness of scaling actions within the system. The average resource utilization of node n at time t, denoted as $U_n^{avg}(t)$, is defined in Equation 1.

$$U_n^{avg}(t) = \frac{1}{M}\sum_{m=1}^{M}\frac{U_n^m(t)}{R_n^m} \quad (1)$$

Where $U_n^m(t)$ is the utilization of the resource m in node n. The average resource balance value of the system at time t, denoted as $B^{avg}(t)$ is shown in Equation 2.

$$B^{avg}(t) = \frac{1}{N}\sum_{n=1}^{N} B^n(t) \quad (2)$$

Where $B^n(t)$ is the resource balance value for node n at time t and is shown in equation 3:

$$B^n(t) = 1 - \sqrt{\frac{1}{M}\sum_{m=1}^{M}(\frac{U_n^m(t)}{R_n^m} - U_n^{avg}(t))^2} \quad (3)$$

Assume the cloud platform is requested with S_i(i=1,2,3….I) services with the varying workload W_j $(i = 1,2,3 \ldots\ldots I)$. The system workload involves requests for multiple types of services with varying demands, requiring the use of different scaling strategies, such as horizontal or vertical scaling. To cope with the demand of the workload, the auto-scaling action a_i^t is executed at the time t for the service S_i.

Objective

The work contributes to resolving the following problems:

- The objective of the optimal decision is to maintain a balanced allocation of resources, avoiding under-utilization, which causes inefficiencies and unnecessary costs, as well as over-utilization, which can lead to performance degradation. By maximizing the expected average resource balance as shown in equation 4, the system strives to achieve a steady state where resource supply aligns closely with demand.

  Maximize $B^{avg}(t)$. (4)

- The QoS (Quality of Service) constraint is met by ensuring that the response time of service S_i remains below its maximum allowable response time after executing the action $a_i^{t-1,}$ and the total service request for the node should not be more than the total available resource of a node.
- Scaling time should always be less than the time taken for decision-making between action a_i^t and a_i^{t+1}.

DRL Model for Auto-Scaling

In the cloud paradigm for variable-type services, the paper provides an optimal decision-making solution with system usage and QOS perspective. Our DRL model is designed to perform real-time modeling of complex scenarios and proactively determine appropriate scaling actions based on the current system environment and workload predictions.

Reinforcement learning (RL) enables agents to learn optimal behaviors by interacting with the environment and maximizing rewards. The environment provides feedback, evaluating the quality of actions taken by the agent. Positive rewards reinforce behaviors, guiding the agent toward strategies that yield maximum expected rewards over time. Q-learning, a popular RL method, selects actions greedily

and updates value functions based on received rewards, eventually determining the optimal action set. However, traditional RL struggles with complex or continuous state spaces due to the high computational cost of updating value functions. Deep reinforcement learning (DRL) addresses this limitation by leveraging deep learning to efficiently handle large and complex state spaces (Dang-Quang & Yoo, 2021). DRL methods are categorized into value-based and policy-based approaches. Value-based methods, like Deep Q-learning, work in discrete action spaces by approximating value functions with deep networks. Policy-based methods, such as Policy Gradient, handle continuous action spaces by directly optimizing action probabilities based on rewards, improving good actions, and reducing bad ones over time. To accommodate both discrete and continuous action at the same instance, value, and policy-based learning, a variant of the actor-critic model is proposed.

State Space: The state space of the proposed DRL model consists of the available resources state, workload state, and pricing model state. The available resources indicate the utilization of the resources in the server of the cloud platform, which is defined in equation 5.

$$State_1(t) = R_n^m \quad (5)$$

Where $R_n^m(t)$ indicates at time interval t utilization of m=1,2, 3, resources in node n=1,2,3….

The requested resource state space is defined as in equation 6.

$$State_2(t) = VM_n^m(t).Q(t) \quad (6)$$

Where $VM_n^m(t)$ resource type requested is defined in equation 7 and $Q(t)$ quantity of resources.

$$VM_n^m(t) = VM_n^{mo}(t) + VM_n^{ms}(t). \quad (7)$$

$VM_n^{mo}(t)$ is the VM of the on-demand type, and $VM_n^{mS}(t)$ is the VM of the Spot instance type.

The pricing model state, defined in equation 8, is the budget ratio that represents the relationship between the available budget and estimated budget for both on-demand and spot VM.

$$State_3(t) = \frac{B(t)}{EC(t)} \quad (8)$$

Where EC(t)= $VM_n^{mo}(t) * Cost_o$ +$VM_n^{ms}(t) * Cost_s$.

The predicted workload state, defined by $State_4(t) = PR_m(t)$, is the predicted resource m for the target workload at time t. Hence, the state space of the proposed DRL model is defined as in equation 9.

$$S(t)=[State_1(t), State_2(t), State_3(t), State_4(t)] \quad (9)$$

Action space: The proposed DRL model is designed to perform scaling actions, which involve adjusting resources based on the workload demand. In cloud computing, scaling can be categorized into two types: horizontal scaling and vertical scaling. Horizontal scaling adds or removes instances, while vertical scaling adjusts the resources of existing instances. The action space of the scaling is defined by equation 10.

$$\text{Action (t)}=\{hs_{up,}hs_{down}, vs_{up_{cpu}}, vs_{down_{cpu}}, vs_{up\_mem}, vs_{down\_mem}\} \quad (10)$$

Where hs_{up} and hs_{up} define horizontal upscaling and downscaling action respectively, vs_{up_cpu} and vs_{down_cpu} define vertical upscaling and downscaling of CPU of existing resources respectively, vs_{up_mem} and vs_{down_mem} define vertical up_scaling and down_scaling of memory of existing resources respectively.

Reward function: In this paper, the reward function makes the agent choose the action that can optimize the utilization of system resources while satisfying the constraints. To achieve the three

objectives outlined in this paper, the reward function is designed to balance the goals of maintaining resource efficiency, adhering to QoS constraints, and ensuring timely scaling actions. Below is a step-by-step design for the reward function that integrates these considerations. The overall reward function is defined in equation 11.

$$R(t) = R_{balance}(t) + R_{QoS}(t) + R_{scaling}(t) \quad (11)$$

The Reward for resource balance encourages alignment between resource supply $U(t)$ and demand $D(t)$ to maintain efficiency and avoid over-utilization and under-utilization, as given in equation 12.

$$R_{balance}(t) = \alpha . \sum_{n=1}^{N} \sum_{m=1}^{M} |U_n^m(t) - D_n^m(t)| \quad (12)$$

Where $U^m(t)$ is the utilization of m resources in all nodes at time t, $D^m(t)$ is the demand for m resources in all the nodes at time t, $\propto$ and weight for resource balance penalty (initial $\propto = 5$). Minimizing $|U_n^m(t) - D_n^m(t)|$ ensures that resources closely match demand.

Reward for QoS Compliance encourages actions that ensure QoS by penalizing violations in response time, and resource availability is given in equation 13.

$$R_{QoS}(t) = -\beta_1 . 1(RT_i(t) > RT_i^{max}) - \beta_2 . 1(D(t) > A(t)) \quad (13)$$

Where, $RT_i(t)$ response time for service S_i at time t, RT_i^{max} maximum allowable response time for S_i, $D(t)$ total workload demand at time t, $A(t)$ total available resources at time t, β_1, and β_2 penalty weights for response time and resource availability violations (initial $\beta_1 = 10 \; and \; \beta_2 = 20$ equal importance for QoS constraints), use indicator functions (1) to enforce penalties only when constraints are violated.

Reward for Timely Scaling Actions incentivizes scaling decisions that minimize delay and ensure timely resource adjustments are given in equation 14.

$$R_{scaling}(t) = -\gamma . 1(T_{scaling}(t) > T_{decision}) \quad (14)$$

Where $T_{scaling}(t)$ time takes to execute scaling action at time t, $T_{decision}$ time available for making next decision, γ penalty weight for delays in scaling (initial $\gamma = 5$).

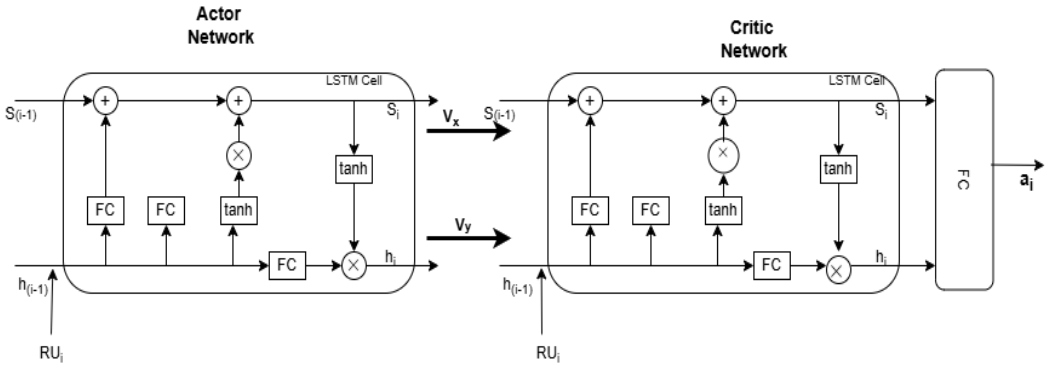

Figure 1: Proposed DRL Model

During training, the behavior of the model is monitored by drawing the following conclusions:

- If the system **violates QoS constraints frequently**, increase $\beta_1$and β_2.
- If the system focuses too much on **cost-efficiency**, leading to QoS degradation, decrease α or γ.

To accommodate both the continuous and discrete action space, the Actor-Critic model is proposed as a combination of value-based learning and policy-based learning. The proposed model employs an Actor-Critic (Xu & Zhao, 2022; Mogal & Sonaje, 2024) framework integrated with an LSTM to optimize resource allocation and task scheduling. The Actor network processes the input state, comprising user requests and VM configurations $R_n^{m,}$ along with historical data H_{i-1}, to produce two outputs: the probability of servers meeting user demands and the cost associated with allocated VMs. Initially, historical data is set to zero. The critical network evaluates these actions by assessing their quality, using the same actor state inputs along with the actor's outputs. It estimates resource allocation efficiency, defined as the ratio of actual resource utilization to total provisioned resources, expressed as a percentage. High value indicates optimal resource usage, while low percentages suggest inefficiencies.

Algorithm: LSTM-driven Actor-Critic Algorithm for autoscaling.

Initialization:

1. Initialize state: $\boldsymbol{S_0}$= [$\boldsymbol{State_1(0), State_2(0), State_3(0), State_4(0), State_5(0)}$].
2. Initialize LSTM-driven actor and critic networks:
   - Actor-Network ($\boldsymbol{\pi_\theta}$): Includes LSTM layers followed by dense layers to output.
   - Critic Network ($\boldsymbol{V_\theta}$): Includes LSTM layers followed by dense layers for state-value estimation.
3. Initialize replay buffer BBB and hyperparameters ($\propto, \boldsymbol{\beta_1, \beta_2, \gamma, \delta, \eta}$).

Execution:

For each episode:

1. For each timestamp $\boldsymbol{t \epsilon T}$:
   1. Input current state $\boldsymbol{S_t}$ into the Actor-Network
      - LSTM Actor Network ($\boldsymbol{\pi_\theta}$):
        - Discrete action network ($\boldsymbol{\pi_\theta^{discrete}}$): Outputs mean and variance for discrete actions.
        - Continuous action network ($\boldsymbol{\pi_\theta^{Continuous}}$): Output mean and variance for continuous actions.
      - Sample actions $\boldsymbol{a_t^{discrete}}$ and $\boldsymbol{a_t^{discrete}}$ from their respective distributions.
      - Construct the final action: $\boldsymbol{a_t} = \{\boldsymbol{a_t^{discrete}}, \boldsymbol{a_t^{discrete}}\}$
   2. Execute action $\boldsymbol{a_t}$ in the environment.
   3. Observe:
      - Reward $\boldsymbol{r_t}$.
      - Next state $\boldsymbol{S_{t+1}}$=[$\boldsymbol{State_1(t+1)}$, $\boldsymbol{State_2(t+1), State_3(t+1), State_4(t+1)}$], Where $\boldsymbol{State_4(t+1)}$ is updated via LSTM's hidden state.
      - Store ($\boldsymbol{S_t}$, $\boldsymbol{at, r_t}$, $\boldsymbol{S_{t+1}}$) in replay buffer B.

Reward calculation:

- Compute individual rewards:
  1. Resource Balance Reward ($\boldsymbol{R_{balance}(t)}$):
  $$\boldsymbol{R_{balance}(t) = \alpha . \sum_{n=1}^{N} \sum_{m=1}^{M} |U_n^m(t) - D_n^m(t)|}$$
  2. QoS compliance reward $\boldsymbol{R_{QoS}(t)}$:
  $$\boldsymbol{R_{QoS}(t) = -\beta_1 . 1(RT_i(t) > RT_i^{max}) - \beta_2 . 1(D(t) > A(t))}$$
  3. Timely scaling reward $\boldsymbol{R_{scaling}(t)}$:
  $$\boldsymbol{R_{scaling}(t) = -\gamma . 1(T_{scaling}(t) > T_{decision})}.$$
- Total reward:
  $$\boldsymbol{R_t = R_{balance}(t) + R_{QoS}(t) + R_{scaling}(t)}.$$

Model update:

1. Sample a mini batch {(S, a,r, S')} from replay buffer B.
2. For each sample:

- Compute state values $\boldsymbol{V(S)}$ and $\boldsymbol{V(S')}$ using the critic network.
- Calculate advantage estimator:

$$\widehat{\boldsymbol{A}} = \boldsymbol{r} + \boldsymbol{\gamma V(S')} - \boldsymbol{V(S)}.$$

- Compute Losses:
  - Critic Loss:

$$\boldsymbol{L_{critic}} = \boldsymbol{MSE(V(S), r + \gamma V(S'))}$$

  - Actor Loss:

$$\boldsymbol{L_{actor}} = -\boldsymbol{E}[\widehat{\boldsymbol{A}}.\log \boldsymbol{\pi_\theta(a|S)}]$$

  - Backpropagate and update: Actor-Network ($\boldsymbol{\pi_\theta^{discrete}}, \boldsymbol{\pi_\theta^{Continuous}}$) and Critic network ($\boldsymbol{V_\emptyset(S)}$) using gradients with learning rate $\boldsymbol{\eta}$

Dynamic reward adjustment:

- Adjust reward weights based on observed performance:
  - If QoS constraints are frequently violated:

$$\boldsymbol{\beta_1 \leftarrow \beta_2 + \delta}, \quad \boldsymbol{\beta_2 \leftarrow \beta_2 + \delta}$$

  - If cost-efficiency leads to QoS degradation

$$\boldsymbol{\alpha \leftarrow \alpha - \delta}, \quad \boldsymbol{\gamma \leftarrow \gamma - \delta}.$$

The Actor-Critic network connects to a fully connected network to further enhance task scheduling. This network user requests and the critic's output (resource efficiency) to assign tasks to the most efficient VM, ensuring better resource utilization. The model incorporates pricing-aware scheduling strategies from previous work to balance cost and performance effectively. Integrating the LSTM model is crucial in both the actor and critic (LSTM-AC) networks to handle sequential dependencies and historical information effectively. This complete proposed model is detailed in Algorithm 1.

Figure 1 illustrates the proposed DRL model, which incorporates the Actor-Critic framework enhanced with an LSTM component. In this model, the Actor-Network takes as input the state S(t)=[$State_1(t)$, $State_2(t)$, $State_3(t)$, $State_4(t)$] and H_{i-1} denotes the historical context maintained by the LSTM. The Actor-Network generates actions ai as outputs, which are structured into two layers. The first output layer, represented as vector v_x, provides the probabilities of servers meeting user requests corresponding to the virtual machines (VMs) hosted on these servers (discrete action space). The second output layer, vector v_y, estimates the cost associated with allocating VMs on the selected servers. Initially, the historical context H_0 is set to zero.

The critic network, on the other hand, evaluates the actor's chosen actions by estimating their value or effectiveness. It provides feedback to guide the actor toward decisions that yield higher expected returns, thereby improving the learning process. The critic network receives as input the same state as shown in Equation 9 and $H_{i-1,}$ reflecting the historical data recorded by the LSTM model in the Critic model. Additionally, the output vectors from the Actor-Network (v_x and v_y) are fed into the critic network (Qiu et al., 2022). The critic's output corresponds to the resource allocation efficiency, calculated as: [Resource allocation efficiency = (Actual Resource Utilization/Total Resource Provisioned)/100]. This metric, expressed as a percentage, assesses how effectively the allocated resources are being utilized. An efficiency close to 100% indicates optimal resource utilization, while lower values suggest overprovisioning or underutilization. The actor-critic network is further connected to a fully connected network tasked with scheduling the allocated VMs. The inputs to this network are the user request U_i and the resource allocation efficiency generated by the actor-critic framework. The output from this network determines the assignment of tasks to the most efficient VMs, ensuring optimal resource utilization. Pricing-aware task scheduling, which complements this process, has been addressed in our earlier work (Shruthi & Shruthi, 2024).

4 Experimental Setup and Result Discussion

This section of our paper outlines the experimental setup and the datasets utilized in the study. Subsequently, the performance of the proposed model is assessed and compared with existing approaches to demonstrate its improvements and advantages.

Experimental Setup and Datasets

In this study, the experimental dynamic cloud environment consists of two clouds, each hosting a single server, with every server containing two VMs. The default experimental parameters for the formulated model are specified as follows.

Table 1: VM Configuration

VM Types	Core	CPU(MIPS)	RAM (GB)	Disk	VM price($/hr)
t3. small	2	500	16	2GB	0.0224
t3. medium	2	1000	64	4GB	0.0448
t3. large	2	2000	128	8GB	0.0893
t3. X-large	4	5000	256	16GB	0.1786

We analyzed four distinct types of Amazon EC2 virtual machines (VMs), as detailed in Table 1 (Shruthi & Shruthi, 2024). These EC2 instances were configured with identical specifications in the MATLAB simulation environment. The Core column represents the number of CPUs available for each VM type, while the CPU (MIPS) indicates their processing capacity, measured in millions of instructions per second. The VM Cost column outlines the hourly computational cost of these instances. Spot instances are characterized by the same specifications as their on-demand counterparts but have prices that vary over time. The data utilized spans from September 18 to November 7, 2023, within the AWS US-East (Ohio) region. In our work, we have relied on real-world workload traces, that is, Google Cluster traces. We have switched to a real-world dataset rather than the synthetic workload to ensure the proposed work is generating the results as per our expectations. In the future, we have planned to extend the work to large scale configurations in real time.

The dataset used in this study is a trace of workloads executed on eight Google Borg compute clusters in May 2019. This trace provides detailed information about job submissions, scheduling decisions, and resource usage for all jobs that were executed within these clusters. The information about eight various Borg cells in May 2019 is available in the cluster-2019 trace (www.kaggle.com,2024).

Evaluation Metric

To evaluate the performance of the proposed approach, various evaluations and metrics are defined, such as resource utilization efficiency, QoS Compliance, scaling decisions, and budget utilization.

Resource utilization efficiency: This metric quantifies how effectively the provisioned resources are being used and is shown in equation 15. An efficiency closer to 100% indicates optimal utilization, whereas lower values suggest over- or under-provisioning.

$$Resource\ utilzation\ efficiency = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M} U_m^n(t)}{\sum_{n=1}^{N}\sum_{m=1}^{M} R_m^n(t)} \quad (15)$$

Where $U_m^n(t)\ is\ the$ utilization of resource m on node n at time t and $R_m^n(t)$ provisioned amount of resource m on node n. This could be the allocated CPU, memory, bandwidth, etc.

QoS Compliance: this metric evaluates whether the system meets the required Quality of Service (QoS) constraints and is defined in equation 16. It ensures the system's responsiveness and that its

availability requirements are fulfilled. The formula for QoS compliance is based on the response time condition and resource availability condition.

$$QoS\ Complaince = 1 - [\rho_1 . 1(RT_i(t) < RT_i^{max}(t)) + \rho_2 . 1(D(t) > A(t))] \quad (16)$$

Where $\rho_1\ and\ \rho_2$ are weights assigned to penalize violations in response time and resource availability, respectively. A QoS Compliance value of 1 indicates that all constraints are met (perfect compliance). A lower QoS Compliance value implies more violations, with penalties determined by the weights $\rho_1$and ρ_2.

Scaling decision: It typically refers to the process of determining whether to scale up (add more resources) or scale down (reduce resources) based on the workload demands, available resources, and performance criteria. When considering horizontal scaling (adding or removing instances) and vertical scaling (modifying resources of existing instances), the scaling decision process needs to evaluate both options based on resource demands, workload, and cost-efficiency, as shown in Equation 17.

$$Scaling\ Decision = \begin{cases} Horizontal\ Scaling, if\ flag = 1 \\ Vertical\ Scaling. otherwise \end{cases} \quad (17)$$

Where flag=1 means $C_{horizontal} < C_{vertical}\ and\ T_{scaling,horizontal} < T_{decision}$.

Where $C_{horizontal}$ is the cost for horizontal scaling and $C_{vertical}$ is the cost for vertical scaling. $T_{scaling,horizontal} < T_{decision}$ latency associated with each horizontal scaling type.

Budget Utilization: This metric evaluates how effectively the available budget is being used to meet the system's resource demands. It considers the allocated budget and the actual resource expenditure, ensuring that cost efficiency is shown in Equation 18.

$$Budget\ ultization = \frac{Actual\ expenditure}{Allocated\ Budget} * 100 \quad (18)$$

Where Actual expenditure is formulated in our previous work (Shruthi & Shruthi, 2024). The experiments were carried out using MATLAB R2021b, utilizing essential toolboxes such as rlAgent, rlRepresentation, rlEnvironment, rlQAgent, and rlValueRepresentation, which are integral to reinforcement learning and deep learning applications. The Reinforcement Learning Toolbox was employed to design and train the actor-critic models, enabling the implementation of various reinforcement learning algorithms and customization of the training process. Additionally, the Deep Learning Toolbox was instrumental in developing and implementing LSTM networks, which are crucial for modeling temporal dependencies in resource allocation tasks. The hyperparameters used in the experimental setup are detailed in Table 2.

Table 2: Hyperparameters Used in the Experiment

Hyperparameter	Value
The learning rate for an actor (α_a)	0.001
The learning rate for critic (α_c)	0.002
Discount factor (γ)	0.99
Experience Memory Capacity (C)	10,000
Batch Size (K)	64
LSTM sequence number	20 steps
Exploration parameter $\in$	1
Reward weights: 1.Resource balance reward α 2.QoS Compliance reward β_1 and β_2 3.Timely Scaling Reward γ	 0.5 0.3 and 0.2 respectively 0.2

The accuracy of the proposed hybrid scaling model is evaluated and compared with previous work using specific performance metrics, including MSE, RMSE, MAE, and R^2. The author of the work (Qiu et al., 2022) used these metrics to evaluate the Bi-directional LSTM model. In the proposed work, we have used a scheduling and scaling approach, which is identified as a research gap where most of the authors have not considered both scheduling and scaling for efficient resource utilization. These metrics are defined and presented in detail in Table 3.

The results of the horizontal scaling and hybrid scaling (vertical and horizontal scaling) approaches appear to have a slight variation. This indicates that extending the state space and action space significantly improved the model's prediction accuracy, leading to a reduction in the MSE, RMSE, and MAE metrics.

Table 3: Experiment Results are Tabulated

	LSTM-AC for Horizontal scaling	LSTM-AC for Hybrid scaling (current work)
MSE	14.642	13.803
RMSE	3.826	3.598
MAE	25.28	24.90
Time taken for Prediction(ms)	4.3	4.2987

In addition to the model performance parameters, we incorporate system performance analysis metrics such as resource utilization, throughput, and SLA violations to demonstrate the efficiency of the proposed system. In our previous two papers, we focused on two stages of autoscaling: scheduling and vertical scaling. However, in this paper, we specifically focus on hybrid scaling (vertical scaling and horizontal scaling). In the autoscaling process, it is challenging to determine the sequence of events—whether scheduling or scaling should occur first since both processes are interdependent and occur simultaneously.

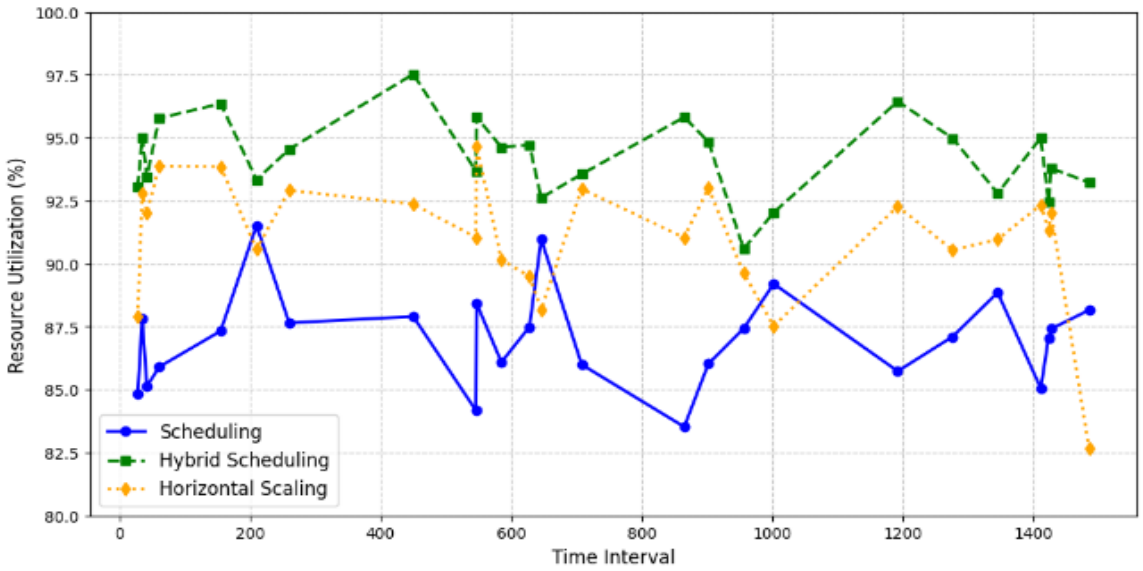

Figure 2: Resource Utilization at Various Stages of Autoscaling

Figure 2 illustrates the resource utilization at various stages of autoscaling after integrating the proposed model. Scheduling alone contributes minimally to the overall resource utilization, whereas hybrid scaling consistently enhances the system's efficiency.

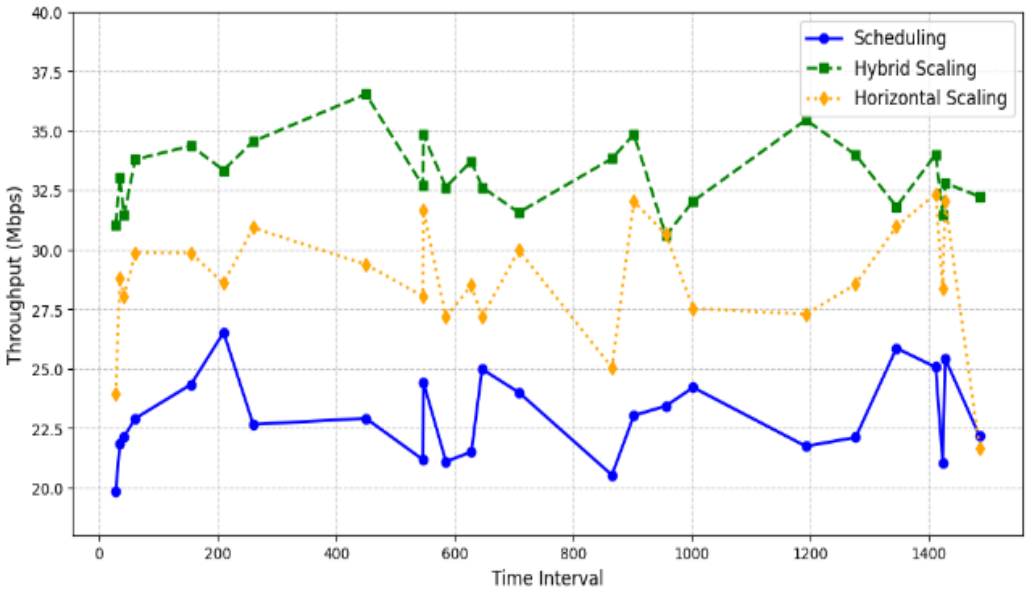

Figure 3: Throughput at Various Stages of Autoscaling

In addition to resource utilization, throughput is another key metric used to highlight the effectiveness of the proposed model. Figure 3 demonstrates that the proposed model achieves the highest throughput, underscoring its superior performance.

Service Level Agreements (SLAs) are formal contracts between service providers and customers that define the level of service expected from the provider to provide Quality of Service (QoS). SLAs specify measurable parameters and expectations for the quality, availability, and reliability of cloud services. If the cloud service provider fails to meet the agreed-upon levels in the SLA, the violation is usually penalized. The penalty may take the form of reduced charges, compensation, or other remedies, depending on the severity of the breach. The SLA violations in our proposed model are depicted in Figure 4. This parameter adds another advantage, demonstrating that our proposed model is more efficient. While scheduling alone results in more SLA violations, hybrid scaling significantly reduces the number of violations compared to just horizontal scaling.

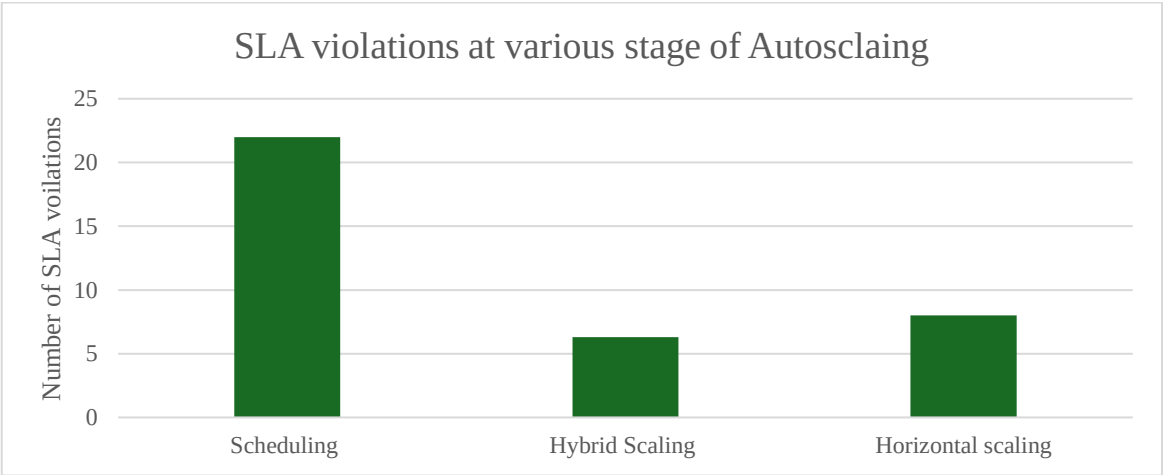

Figure 4: SLA Violation at the Stages of Autoscaling

Comparison with State-of-the-art Approaches

We compared the performance of the proposed hybrid scaling model with some of the state-of-art methods like Q-learning, A traditional reinforcement learning approach for decision-making systems like resource allocation, and SARSA: A state-action-reward-state-action method optimized for online scaling. DRL: A deep reinforcement learning model integrating recurrent neural networks (RNN) proposed by authors (Ebadi et al., 2024; Xu & Zhao, 2022; Mogal & Sonaje, 2024; Joshi et al., 2024; Daraje & Shaikh, 2021; Agarwal et al., 2024; Garí et al., 2022). The results, summarized in Table 4, demonstrate the superiority of the proposed model across key performance parameters.

Table 4: Comparison of State-of-art Models with the Proposed Model

Performance metric	Q-learning	SARSA	DRL-Based model (RNN)	LSTM enhanced Actor-Critic model
Resource Utilization Efficiency	85.3%	87.5%	90.2%	92.7%
Average SLA violation (%)	12.8%	10.4%	7.2%	6.3%
Average response time (ms)	155	150	145	142

Unlike traditional Q-learning and SARSA methods, the integration of Long Short-Term Memory (LSTM) networks allows the proposed model to capture temporal dependencies in resource demand patterns. This enables proactive scaling decisions, reducing response times and SLA violations. The inclusion of a pricing model optimizes the use of on-demand and spot instances, reducing overall operational costs. Figure 5 illustrates that the proposed model achieves a cost reduction of 12% compared to the DRL-based approach (RNN).

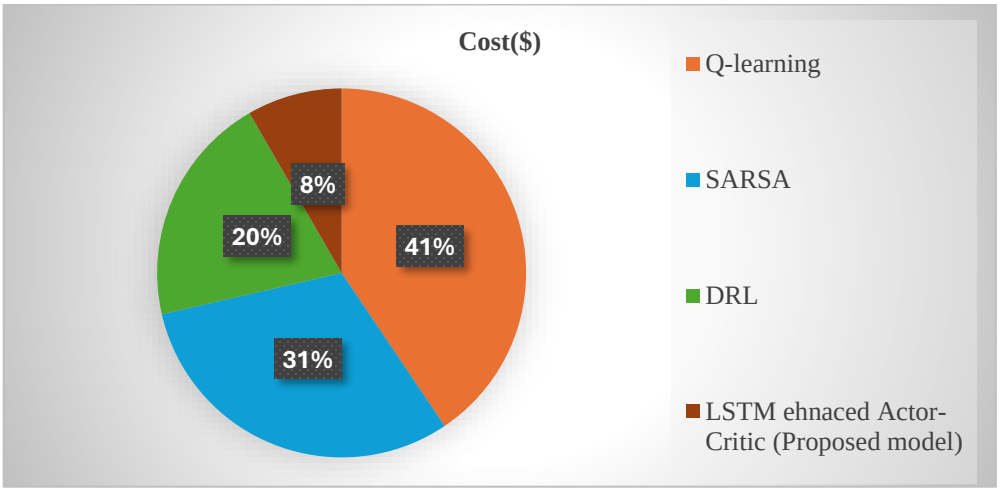

Figure 5: Demonstrates Cost Reduction Comparison

5 Conclusion

In this study, we have introduced a novel approach to autoscaling in cloud computing environments by leveraging an LSTM-enhanced Actor-Critic model to optimize resource allocation. By extending the traditional Actor-Critic framework with Long-Short-Term Memory (LSTM) networks, our model effectively captures scheduling, vertical scaling, and horizontal scaling and has been demonstrated to significantly improve resource utilization efficiency and throughput. Experimental results validate the

effectiveness of the LSTM-based model in achieving proactive scaling decisions, which reduce SLA violations and enhance overall system performance.

The proposed model achieves high resource utilization efficiency, maintaining an average of 92.7% across varying workloads, while SLA violations were reduced to 6.3% compared to higher rates observed in traditional methods. Additionally, the proposed model's integration of a pricing mechanism led to a 12% reduction in total operational costs, further solidifying its cost- awareness the incorporation of both temporal modeling through LSTM and cost awareness through a pricing mechanism allows the system to make intelligent and efficient scaling decision, even in complex and fluctuating environments. Future work can explore further optimizations to extend the model's applicability. For instance, incorporating parameters like network latency and multi-cloud strategies enhances the robustness and adaptability of the cloud. Additionally, deploying the model in real-world cloud platforms will provide further insights into its scalability and practical effectiveness, paving the way for broader adoption in diverse cloud environments.

Conflict of Interest

Declare conflicts of interest or state "The authors declare no conflict of interest." The creators affirm that they don't have any contending monetary interests or individual connections that might have affected the work introduced in this paper.

Author Contributions

"Conceptualization, Shruthi. P. S methodology, Shruthi. P. S; software, Shruthi. P.S. validation, Shruthi. P. S and D. R. Umesh formal analysis, Shruthi. P.S. investigation, Shruthi. P. S; resources, Shruthi. P. S, data curation, Shruthi. P.S.; writing—original draft preparation, Shruthi. P.S.; writing—review and editing, Shruthi. P.S.; visualization, Shruthi. P.S. supervision, D.R. Umesh; project administration, Shruthi. P.S. and D.R. Umesh".

Acknowledgment

Vidyavardhaka College of Engineering provided full or partial support for this study. We extend our appreciation to our colleagues at Vidyavardhaka College, who serve as subject experts, offering invaluable insights and expertise that greatly contributed to this research. While they may not endorse all interpretations or conclusions presented in this paper, this assistance was indispensable. I express sincere gratitude to the supervisor for unwavering support throughout the research process.

References

[1] Aboorva, V., Sree, P. K., Sowmiya, B., Sowmiya, N., & Menaka, S. R. (2023). An Effective Fog-Enabled E-Health with Hierarchical Attribute Revocation using Cloud Computing. *International Journal of Advances in Engineering and Emerging Technology*, *14*(1), 52-57.

[2] Agarwal, S., Rodriguez, M. A., & Buyya, R. (2024). A deep recurrent-reinforcement learning method for intelligent autoscaling of serverless functions. *IEEE Transactions on Services Computing*. https://doi.org/10.1109/TSC.2024.3387661

[3] Alnumay, W.S. (2024). The past and future trends in IoT research. National Journal of Antennas and Propagation, 6(1), 13–22. https://doi.org/10.32996/jcsts

[4] Chen, Z., Hu, J., Min, G., Luo, C., & El-Ghazawi, T. (2021). Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning. *IEEE Transactions on Parallel and Distributed Systems*, *33*(8), 1911-1923. https://doi.org/10.1109/TPDS.2021.3132422

[5] Dang-Quang, N. M., & Yoo, M. (2021). Deep learning-based autoscaling using bidirectional long short-term memory for kubernetes. *Applied Sciences*, *11*(9), 3835.https://doi.org/10.3390/app11093835

[6] Daraje, M., & Shaikh, J. (2021, December). Hybrid resource scaling for dynamic workload in cloud computing. In *2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICMNWC52512.2021.9688556

[7] Ebadi, M. E., Yu, W., Rahmani, K. R., & Hakimi, M. (2024). Resource Allocation in The Cloud Environment with Supervised Machine learning for Effective Data Transmission. *Journal of Computer Science and Technology Studies*, *6*(3), 22-34.

[8] Garí, Y., Monge, D. A., & Mateos, C. (2022). A Q-learning approach for the autoscaling of scientific workflows in the Cloud. *Future Generation Computer Systems*, *127*, 168-180. https://doi.org/10.1016/j.future.2021.09.007

[9] Garí, Y., Pacini, E., Robino, L., Mateos, C., & Monge, D. A. (2024). Online rl-based cloud autoscaling for scientific workflows: Evaluation of q-learning and sarsa. *Future Generation Computer Systems*, *157*, 573-586. https://doi.org/10.1016/j.future.2024.04.014

[10] https://www.kaggle.com/datasets/derrickmwiti/google-2019-cluster-sample/data

[11] Jagan, B. O. L. (2024). Low-power design techniques for VLSI in IoT applications: Challenges and solutions. Journal of Integrated VLSI, *Embedded and Computing Technologies, 1*(1), 1-5.

[12] Jang, H. J., Yim, Y. G., & Jin, H. W. (2020, December). Vertical autoscaling of GPU resources for machine learning in the cloud. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 5710-5712). IEEE. https://doi.org/10.1109/BigData50022.2020.9378248

[13] Janmohammadi, P., & Babazade, M. (2015). Resource Management in the Cloud Computing Using a Method Based on Ant Colony Optimization. *International Academic Journal of Science and Engineering*, *2*(6), 40-54.

[14] Joshi, S., Pandey, N. K., & Mishra, A. K. (2024, April). Reinforcement Learning Based Auto Scaling Strategy Used in Cloud Environment: State of Art. In *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 731-736). IEEE. https://doi.org/10.1109/CSNT60213.2024.10545922

[15] Kang, H., Koh, J. I., Kim, Y., & Hahm, J. (2013, September). An SLA driven VM auto-scaling method in hybrid cloud environment. In *2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)* (pp. 1-6). IEEE.

[16] Khaleq, A. A., & Ra, I. (2021). Intelligent autoscaling of microservices in the cloud for real-time applications. *IEEE access*, *9*, 35464-35476. https://doi.org/10.1109/ACCESS.2021.3061890

[17] Kumar, M., Sharma, S. C., Goel, S., Mishra, S. K., & Husain, A. (2020). Autonomic cloud resource provisioning and scheduling using meta-heuristic algorithm. *Neural Computing and Applications*, *32*, 18285-18303.

[18] Manzoor, M. F., Abid, A., Farooq, M. S., Nawaz, N. A., & Farooq, U. (2020). Resource allocation techniques in cloud computing: A review and future directions. *Elektronika ir Elektrotechnika*, *26*(6), 40-51. https://doi.org/10.5755/j01.eie.26.6.25865

[19] Marie-Magdelaine, N., & Ahmed, T. (2020, December). Proactive autoscaling for cloud-native applications using machine learning. In *GLOBECOM 2020-2020 IEEE Global Communications Conference* (pp. 1-7). IEEE. https://doi.org/10.1109/GLOBECOM42002.2020.9322147

[20] Mogal, A. K., & Sonaje, V. P. (2024, March). Predictive Autoscaling for Containerized Applications Using Machine Learning. In *2024 1st International Conference on Cognitive,*

Green and Ubiquitous Computing (IC-CGU) (pp. 1-6). IEEE. https://doi.org/10.1109/IC-CGU58078.2024.10530773

[21] Nithyalakshmi, V., Sivakumar, R., & Sivaramakrishnan, A. (2021). Automatic Detection and Classification of Diabetes Using Artificial Intelligence. *International Academic Journal of Innovative Research, 8*(1), 01–05. https://doi.org/10.9756/IAJIR/V8I1/IAJIR0801

[22] Pavlenko, A., Saur, K., Zhu, Y., Kroth, B., Cahoon, J., & Camacho-Rodríguez, J. (2024, May). VASIM: Vertical Autoscaling Simulator Toolkit. *In 2024 IEEE 40th International Conference on Data Engineering (ICDE)* (pp. 5413-5416). IEEE. https://doi.org/10.1109/ICDE60146.2024.00416

[23] Qiu, H., Mao, W., Patke, A., Wang, C., Franke, H., Kalbarczyk, Z. T., ... & Iyer, R. K. (2022, April). Reinforcement learning for resource management in multi-tenant serverless platforms. In *Proceedings of the 2nd European Workshop on Machine Learning and Systems* (pp. 20-28). https://doi.org/10.1145/3517207.3526971

[24] Radhika, E. G., & Sadasivam, G. S. (2021). A review on prediction based autoscaling techniques for heterogeneous applications in cloud environment. *Materials Today: Proceedings*, *45*, 2793-2800. https://doi.org/10.1016/j.matpr.2020.11.789

[25] Rahim, R. (2024). Adaptive Algorithms for Power Management in Battery-Powered Embedded Systems. *SCCTS Journal of Embedded Systems Design and Applications*, *1*(1), 20-24. https://doi.org/10.31838/ESA/01.01.05

[26] Rahim, R. (2024). Optimizing reconfigurable architectures for enhanced performance in computing. *SCCTS Transactions on Reconfigurable Computing, 1(*1), 11-15.

[27] Rattihalli, G., Govindaraju, M., Lu, H., & Tiwari, D. (2019, July). Exploring potential for non-disruptive vertical auto scaling and resource estimation in kubernetes. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)* (pp. 33-40). IEEE. https://doi.org/10.1109/CLOUD.2019.00018

[28] Saxena, D., & Singh, A. K. (2021). A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center. *Neurocomputing*, *426*, 248-264. https://doi.org/10.1016/j.neucom.2020.08.076

[29] Shi, F., & Lin, J. (2022). Virtual machine resource allocation optimization in cloud computing based on multiobjective genetic algorithm. *Computational Intelligence and Neuroscience*, *2022*(1), 7873131. https://doi.org/10.1155/2022/7873131

[30] Shiraishi, Y., Mohri, M., & Fukuta, Y. (2011). A Server-Aided Computation Protocol Revisited for Confidentiality of Cloud Service. *J. Wirel. Mob. Networks Ubiquitous Compute. Dependable Appl.*, *2*(2), 83-94.

[31] Shruthi, P. S., & Umesh, D. R. (2024). Pricing Model - Aware Task Scheduling in Cloud paradigm enhanced with DRL on MAPE Framework. *Int J Intel System Application Eng, 12*(3), 1612–1619.

[32] Shukur, H., Zeebaree, S., Zebari, R., Zeebaree, D., Ahmed, O., & Salih, A. (2020). Cloud computing virtualization of resources allocation for distributed systems. *Journal of Applied Science and Technology Trends*, *1*(2), 98-105. https://doi.org/10.38094/jastt1331

[33] Xu, Y., & Zhao, J. (2022, April). Actor-critic with transformer for cloud computing resource three stage job scheduling. In *2022 7th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)* (pp. 33-37). IEEE. https://doi.org/10.1109/ICCCBDA55098.2022.9778883

[34] Zhang, H., Guo, T., Tian, W., & Ma, H. (2024). Learning-driven hybrid scaling for multi-type services in cloud. *Journal of Parallel and Distributed Computing*, *189*, 104880. https://doi.org/10.1016/j.jpdc.2024.104880

Authors Biography

P.S. Shruthi holds a Bachelor of Engineering in Computer Science and Engineering from CIT Gubbi, Tumkur, and a Master of Technology in Computer Science and Engineering Specialised in Computer Networks from RNSIT Bangalore, besides several professional certificates and skills. Currently Working at VVCE Mysuru as an Assistant Professor. Member of the Indian Society for Technical Education (ISTE). Research includes Cloud Computing, Machine learning, and Deep Reinforcement learning

Dr.D.R. Umesh holds a Doctor of Philosophy from Mysuru University, Karnataka, India, 2016. He is currently a Professor and Program Head of the Computer Science and Engineering (AI&ML) department at PES College of Engineering, Mandya. His research area includes Artificial Neural Networks, Machine Learning, Data Analytics and Data Mining, Breast Cancer Classification using Conformal Prediction, Study on Cancer Incidence, and Breast Cancer Diagnosis. He published more than 20 papers in his area of research at international journals and Conferences.