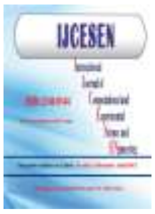
International Journal of Computational and Experimental Science and ENgineering
(IJCESEN)

Vol. 11-No.1 (2025) pp. 1015-1031
http://www.ijcesen.com



ISSN: 2149-9144

Research Article

Design and Implementation of Hybrid Adaptive Neural Architecture for Self-Absorption in Virtual Machines

Naga Mallikharjunarao BILLA[1]*, Prasadu PEDDI[2], Manendra Sai DASARI[3]

[1]Research scholar, Shri Jagdishprasad Jhabarmal Tibrewala University, Vidyanagri, Jhunjhunu, Rajasthan
* **Corresponding Author Email:** mallikharjunarao.b@gmail.com - **ORCID:** 0000-0002-3362-9417

[2]Professor Dep of CSE & IT, Shri Jagdishprasad Jhabarmal Tibrewala University, Vidyanagri, Jhunjhunu, Rajasthan
Email: peddiprasad37@gmail.com - **ORCID:** 0000-0001-9717-934X

[3]Professor, Vignan's Institute of Engineering for Women, Visakhapatnam
Email: dmsai007@gmail.com - **ORCID:** 0009-0005-9561-7904

Article Info:

DOI: 10.22399/ijcesen.953
Received : 07 December 2024
Accepted : 27 January 2025

Keywords :

Virtual Machines,
Resource Management,
Neural Architecture,
Adaptive Learning,
Self-Absorption,
Dynamic Workloads.

Abstract:

This study introduces a Hybrid Adaptive Neural Architecture designed to address the dynamic resource management challenges in Virtual Machines (VMs). Current static and heuristic-based approaches are insufficient for adapting to real-time workload variations, resulting in inefficiencies, latency, and resource contention. The proposed architecture leverages neural networks, including convolutional and recurrent layers, integrated with adaptive mechanisms such as reinforcement and transfer learning, to enable self-absorptive capabilities in VMs. This self-adaptation allows VMs to autonomously learn from operational data, predict resource demands, and adjust allocations in real-time, optimizing performance and minimizing overhead. Experimental evaluation across diverse workload patterns demonstrated the architecture's effectiveness. For burst workloads, the proposed system achieved a 98.6% success rate, outperforming heuristic methods (77.3%) and static allocation (64.2%). Under steady workloads, it maintained 94.9% throughput consistency, compared to 81.7% and 70.3%, respectively. The architecture reduced ephemeral workload allocation lag to 28.7 ms, significantly outperforming heuristic (115.6 ms) and static approaches (205.4 ms). Additionally, the proposed system improved resource utilization, achieving 84.7% CPU efficiency and 92.4% memory efficiency, while maintaining a low latency of 48.6 ms. These results validate the system's ability to dynamically allocate resources efficiently, adapt to workload variability, and enhance overall VM performance. The findings set a benchmark for neural-based resource management in virtualized environments, paving the way for scalable, autonomous solutions in modern computing infrastructures.

1. Introduction

The rapid evolution of virtualization technologies has revolutionized modern computing, enabling scalable and efficient resource management in diverse environments such as cloud data centers, edge computing, and containerized microservices. Virtual Machines (VMs) serve as the backbone of these virtualized ecosystems, providing isolation, scalability, and efficient resource utilization [1]. However, the dynamic and unpredictable nature of workloads presents a significant challenge in maintaining optimal performance and resource allocation within VMs. Traditional resource management methods, often static or heuristic-based, struggle to adapt to these variations in real-time, resulting in inefficiencies, performance degradation, and potential system bottlenecks.
To address these challenges, there is a growing need for autonomic and self-adaptive mechanisms that enable VMs to dynamically manage their resources in response to fluctuating workloads. Inspired by the concept of "self-absorption" in biological and computational systems, where entities autonomously assimilate knowledge to adapt and optimize, this research explores the integration of adaptive neural architectures into VM resource management [2]. Self-absorption in the context of VMs refers to their capability to continuously learn from operational data, predict resource demands,

and adjust allocations proactively without manual intervention. This paradigm shifts the focus from reactive to proactive resource optimization, ensuring higher efficiency and reduced operational overhead. Furthermore, advances in artificial intelligence (AI), particularly neural networks, offer a robust foundation for developing such self-adaptive systems. Neural architectures, including convolutional and recurrent networks, excel in capturing spatial and temporal patterns in complex data, making them ideal for modeling the intricate relationships between workload characteristics and resource utilization [3]. When combined with adaptive learning mechanisms like reinforcement learning and transfer learning, these architectures can evolve dynamically to meet the ever-changing demands of virtualized environments.

The motivation for this research lies in bridging the gap between static resource allocation methods and the need for intelligent, self-adaptive systems. By leveraging neural architectures and embedding self-absorptive capabilities within VMs, this study aims to enhance resource optimization, minimize latency, and reduce resource contention, ultimately driving the next generation of efficient and autonomous virtualization solutions [4]. Managing the dynamic demands of Virtual Machines (VMs) remains a significant challenge due to the limitations of traditional static and heuristic-based resource allocation methods, which fail to adapt to real-time workload variations. Adaptive learning applied in personalized E-Learning experiences [5]. Implementing neural architecture for resource management offers potential solutions but presents challenges [6], including modeling complex workload-resource relationships, incorporating dynamic learning mechanisms like reinforcement learning, and ensuring seamless integration with existing virtualization infrastructures. Addressing these gaps requires a scalable and adaptive framework capable of optimizing resource allocation and enhancing VM performance in diverse and dynamic environments [7].

The primary objective of this research is to design and implement a Hybrid Adaptive Neural Architecture that addresses the limitations of traditional resource management in Virtual Machines (VMs). By leveraging neural networks and adaptive learning mechanisms, the proposed framework aims to optimize resource allocation, enhance scalability, and improve the responsiveness of VMs under dynamic workload conditions. The system seeks to achieve self-absorptive capabilities, enabling VMs to autonomously learn from operational data and adapt resource allocation strategies in real-time.

To achieve this, the research is guided by the following specific goals:

Develop a Neural Architecture: Design a hybrid neural network that integrates convolutional, recurrent, and fully connected layers to capture spatial and temporal workload patterns.

Incorporate Adaptive Mechanisms: Implement reinforcement learning and transfer learning techniques to enable the system to dynamically adjust to changing workloads and environments.

Seamless Integration: Ensure compatibility with existing VM infrastructures, including hypervisors and container orchestration platforms, to facilitate real-world deployment.

Performance Evaluation: Assess the architecture's efficiency, scalability, and adaptability using comprehensive performance metrics under diverse workload scenarios.

The research seeks to address the following key questions:

1. How effectively can the proposed architecture optimize resource allocation compared to static and heuristic-based methods?
2. To what extent can the system adapt to unpredictable workload patterns, such as burst and ephemeral demands, in real-time?
3. How do the incorporation of reinforcement learning and transfer learning enhance the adaptability and efficiency of the neural framework?
4. What are the computational and integration overheads of the proposed system, and how can they be minimized?

By addressing these objectives and questions, this study aims to advance the state-of-the-art in neural-based resource management for virtualized environments, offering a scalable and autonomous solution for modern computing infrastructures.

This research introduces a novel called Hybrid Adaptive Neural Architecture that integrates advanced neural network designs and adaptive learning mechanisms to address the dynamic resource management challenges in Virtual Machines (VMs). Key contributions include the development of a self-absorptive system capable of autonomously learning from operational data, the incorporation of reinforcement and transfer learning for real-time adaptability, and the formulation of new performance metrics for resource optimization. A comprehensive software prototype has been implemented and rigorously evaluated under diverse workload scenarios, demonstrating significant improvements in resource utilization, latency reduction, and system throughput. These advancements set a new benchmark in VM management, bridging the gap between static resource allocation and intelligent, adaptive solutions.

The paper is organized as follows: Section 2 reviews the existing literature on virtualization and adaptive neural architectures, identifying research gaps. Section 3 introduces the proposed Hybrid Adaptive Neural Architecture, detailing its conceptual framework, system design, and adaptive mechanisms. Section 4 outlines the implementation details, including system specifications, datasets, and evaluation methodologies. Section 5 presents the experimental results, including a comparison with baseline methods and a discussion on the system's effectiveness. Finally, Section 6 provides conclusions, highlights the study's contributions, and proposes future research directions. Each section is structured to build upon the core argument, showcasing the system's innovation and effectiveness in addressing VM resource management challenges.

1.1 Literature Review

Classical VM resource allocation strategies typically involve provisioning computational resources through virtual machines, which provide a high level of isolation but can lead to performance overhead and inefficient resource utilization [8,9]. These approaches often rely on hypervisor-based virtualization, which imposes overhead on CPU, networking, and disk operations [9,10]. Hypervisors create a layer of abstraction between the physical hardware and virtual machines, allowing multiple VMs to run on a single physical host. However, this abstraction comes at a cost, as each VM requires its own guest operating system, leading to increased resource consumption and slower startup times compared to more lightweight solutions [11,12]. Container technologies have emerged as a more efficient alternative to traditional VM-based approaches. Containers share the host operating system kernel, resulting in lower overhead, faster startup times, and improved resource utilization [13,14]. Docker, a popular container platform, has shown to have lower overhead compared to VMware, with performance on container-based infrastructure approaching that of non-virtualized systems [15]. Despite their advantages, container-based systems present new challenges in monitoring and management due to their automated flexibility, ephemerality, and the increasing number of containers in a system [16,17]. Additionally, while containers offer improved resource utilization, they may provide less isolation compared to VMs, potentially raising security concerns in multi-tenant environments. In conclusion, while traditional VM-based approaches offer strong isolation, container technologies provide a more lightweight and efficient solution for resource allocation and management in cloud environments. However, both approaches have their limitations, and the choice between them depends on specific use cases and requirements.

1.2 Neural Networks in System Management

Deep reinforcement learning (DRL) has emerged as a powerful approach for system resource management in virtualized environments. Several studies have explored the application of DRL and other machine learning techniques in this context. In the field of wireless communications, a novel resource allocation algorithm combining DRL and unsupervised learning has been proposed for centralized cellular networks with multiple cells, users, and channels [18,19]. This approach uses a Deep Deterministic Policy Gradient (DDPG) based channel allocation network and an unsupervised learning-based power control network to maximize energy efficiency in dynamic environments. The algorithm demonstrates superior performance in terms of energy efficiency and transmit rate compared to other methods. For vehicular networks, DRL has been extensively studied to address various telecommunications issues [20,21]. The integration of DRL with deep learning has proven effective in solving decision-making problems in large-scale wireless networks, where traditional reinforcement learning may struggle due to massive and complex state and action spaces. In the context of cognitive radio systems, a DRL-based approach has been proposed for multi-dimensional resource allocation [22,23] . This method uses Deep Q-Network and Deep Recurrent Q-Network structures to design strategies for secondary users to share spectrum and control transmission power. The proposed approach shows improved user rewards and reduced collisions compared to traditional methods. While not specifically focused on virtualization, these studies demonstrate the potential of DRL and other machine learning techniques for resource management in complex, dynamic environments. The application of these approaches to virtualized systems could potentially lead to more efficient and adaptive resource allocation strategies.

Despite advancements in virtualization and neural network-based resource management, several challenges persist. Scalability remains a key issue, with existing methods struggling under large-scale systems due to resource contention and computational overheads. Transferability is another limitation, as machine learning models often fail to adapt across diverse virtualized environments. Additionally, the "black box" nature of neural networks hinders interpretability, reducing trust in their automated decisions. Real-time adaptation to dynamic workloads poses further challenges, as current approaches cannot consistently balance speed, efficiency, and accuracy. Energy efficiency is also inadequately addressed, with most solutions prioritizing performance. Lastly, fragmented integration of neural techniques into virtualization platforms highlights the need for a cohesive framework. The proposed Hybrid Adaptive Neural Architecture aims to address these gaps by offering a scalable, interpretable, and adaptable solution for efficient resource management.

2. Material and Methods

2.1 Proposed Hybrid Adaptive Neural Architecture

Conceptual Framework

In this section, we introduce the concept of "self-absorption" in Virtual Machines (VMs), which refers to the system's ability to autonomously assimilate knowledge from operational data to enhance performance and adaptability. This self-absorptive capability enables VMs to dynamically adjust resources and optimize operations without manual intervention [24].

Figure 1 represents the system's architecture, where the Input Layer collects data on resource usage, VM states, and workload characteristics. The Processing Layer utilizes adaptive neural networks and learning modules to analyze this data, extracting features and recognizing patterns. Subsequently, the Output Layer generates resource re-allocation instructions and alerts based on processed information. Finally, the Integration Layer ensures seamless communication and implementation of these instructions within the VM environment through hypervisor integration, communication protocols, and resource management modules. This self-absorptive framework enables VMs to autonomously adapt to changing workloads and operational conditions, thereby enhancing efficiency and reducing the need for manual oversight [25]. The Hybrid Adaptive Neural Architecture is designed to enhance the efficiency and adaptability of Virtual Machines (VMs) by autonomously managing resources and responding to varying workloads. This system integrates various modules, each with specific roles, underpinned by mathematical models and algorithms to achieve optimal performance.

Input Modules

Resource Usage Metrics: These components monitor real-time data on CPU utilization, memory usage, disk I/O, and network throughput. Mathematically, let $R(t)$ represent the resource usage vector at time t, where: $R(t) = [\mathrm{CPU}(t), \mathrm{Memory}(t), \mathrm{DiskIO}(t), \mathrm{Network}(t)]$.
This vector provides a quantitative measure of the VM's performance and resource consumption at any given time.

VM States: This module tracks the operational status of VMs, such as running, stopped, paused, or suspended. Define a state function $S(t)$ that maps to discrete values indicating the VM's state: $S(t) \in \{0$: Stopped, 1: Running, 2: Paused, 3: Suspended $\}$.This function provides a snapshot of the VM's current activity.

Workload Characteristics: This component analyzes the nature of tasks executed by the VM, focusing on I/O intensity, concurrency levels, and resource consumption patterns. Let $W(t)$ denote the workload vector: $W(t) = [$ IO_Intensity (t), Concurrency (t), Resource_Pattern $(t)]$. This vector helps in understanding workload demands over time.

Processing Units

Adaptive Neural Network Layers: This subsystem comprises layers like convolutional, recurren and fully connected layers that process input data to extract features and adapt to changing patterns. The neural network function f_θ with parameters θ can be represented as: $f_\theta(x) = \sigma(W \cdot x + b)$, where x is the input vector, W and b are weights and biases, respectively, and σ is the activation function (e.g., ReLU, Sigmoid). The network adjusts θ to minimize a loss function L, typically through backpropagation: $\theta \leftarrow \theta - \eta \nabla_\theta L$, where η is the learning rate.

Learning Modules: Incorporating techniques such as reinforcement learning, transfer learning, and genetic algorithms, these modules enhance the system's adaptability. For instance, in reinforcement learning, the objective is to maximize the expected cumulative reward R : $\max_\pi \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where π is the policy, γ is the discount factor, and r_t is the reward at time t.

Output Modules

Resource Re-allocation Instructions: Based on processed data, this module generates commands for dynamic resource allocation and load balancing. The optimization problem can be formulated as: $\min_{A(t)} C(A(t))$,subject to performance constraints, where $A(t)$ represents the allocation strategy at time t, and C is the cost function.

Alert Systems: This component detects anomalies like resource overloads or performance issues. Anomalies can be identified by evaluating the deviation $D(t)$ of observed metrics $M(t)$ from expected behavior $E[M(t)]$:$D(t) = M(t) - E[M(t)]$,If $D(t)$ exceeds a predefined threshold, an alert is triggered.

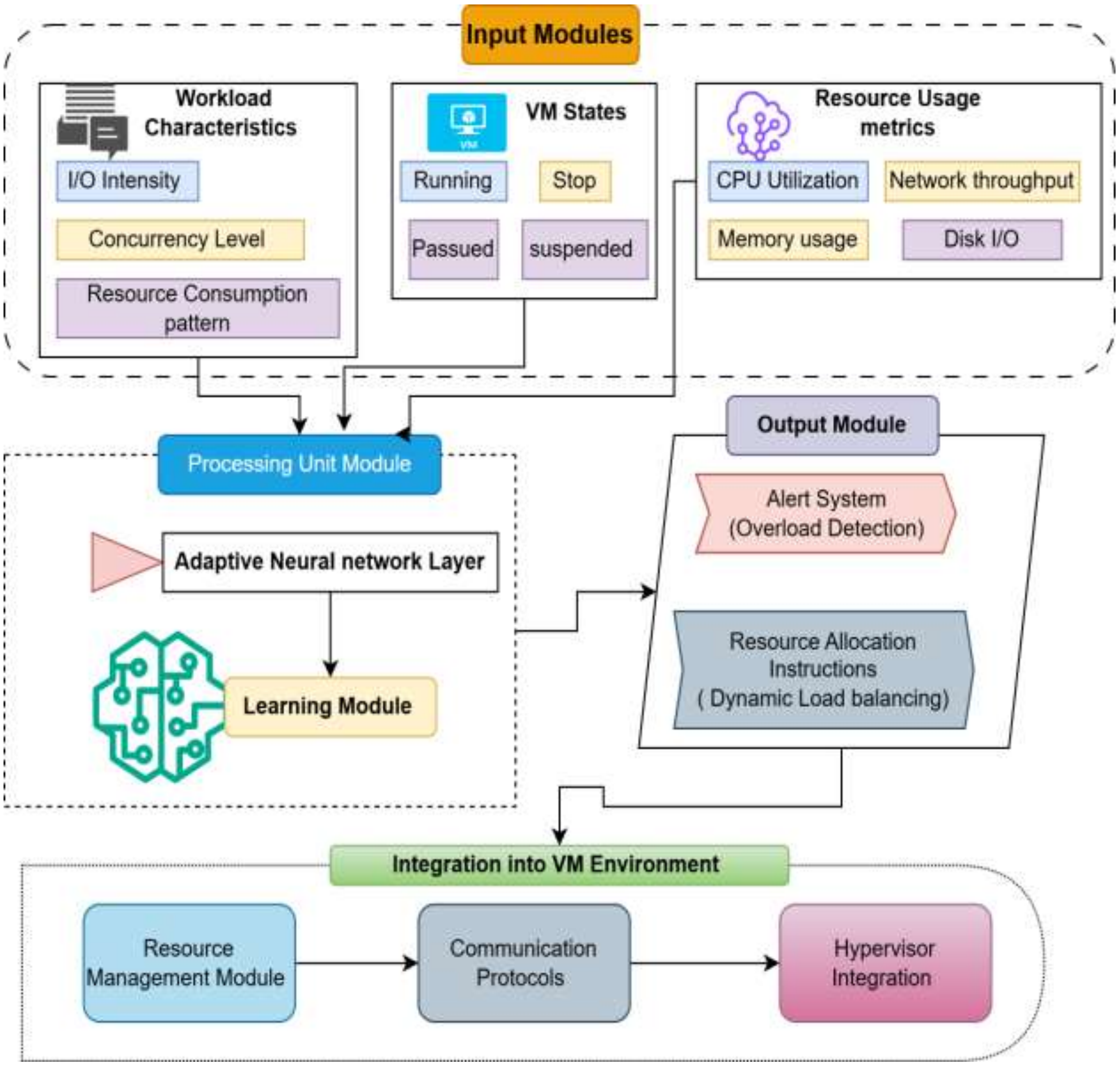


Figure 1. *Block Diagram of the proposed model*

Integration into VM Environment

Hypervisor Integration: Utilizing methods such as VMBus communication and enlightened I/O, this module facilitates interaction between the architecture and the VM's hypervisor. The communication can be modeled as a function H that translates resource allocation instructions $A(t)$ into hypervisor actions: $H: A(t) \rightarrow$ Hypervisor Actions

Communication Protocols: This subsystem manages data exchange between modules through API gateways and data serialization formats like JSON or XML. The data transmission can be represented by a function T that ensures secure and efficient communication: T: Data $\rightarrow$ Serialized Format $\rightarrow$ Data

Resource Management Modules: Implementing dynamic resource allocation and load balancing strategies, this component oversees resource distribution across VMs. The resource management strategy R_m aims to balance the load L across servers to minimize the maximum load: $\min_{R_m} \max_{i} L_i$, where L_i is the load on server i.

Adaptive Neural Network Layer Architecture for Autonomous Virtual Machine Resource Management

Input Layer

The Input Layer acts as the gateway for data entering the neural network, processing normalized features derived from Virtual Machine (VM) operations. It captures key system attributes, including Resource Usage Metrics ($R(t)$), which represent real-time measurements of CPU, memory, disk I/O, and network utilization, providing insights into performance and resource consumption. VM States

($S(t)$) encode the current operational status of the VM (e.g., Running = 1, Paused = 2, Suspended = 3) as discrete values. Workload Characteristics ($W(t)$) describe the nature of tasks, including I/O intensity and concurrency levels ($W(t) =$ [IO_Intensity (t), Concurrency (t)]). These attributes combine into a single input vector, $\mathbf{x}_t = [R(t), S(t), W(t)]$, encapsulating a snapshot of the VM's state at time t and serving as the foundational input for subsequent layers of the network [26].

Hidden layer

The hidden layers in the architecture are designed to progressively extract features, model patterns, and learn complex relationships from input data. These layers include Convolutional Layers, which extract spatial patterns from resource usage metrics and workload characteristics by applying filters to detect localized patterns like peak resource usage or workload spikes. Mathematically, the output of a convolutional layer at location i, j is represented as $z_{ij}^{(l)} = \sigma\left(\sum_{k=1}^{K} w_k^{(l)} \cdot x_{i+k-1, j+k-1}^{(l-1)} + b^{(l)}\right)$, where $w_k^{(l)}$ are the filter weights and σ is the activation function (e.g., ReLU). Recurrent Layers (e.g., LSTM) capture temporal dependencies in time-series data, such as resource utilization trends, by maintaining memory of sequential relationships. The hidden state at time t is computed as $h_t = \sigma(W_h \cdot [h_{t-1}, x_t] + b_h)$, where W_h is the weight matrix for the input and hidden states. Finally, Fully Connected Layers aggregate features from the convolutional and recurrent layers, producing high-level representations through $y = \sigma(W \cdot h + b)$, where W and b are the weights and biases. Together, these hidden layers form the backbone of the neural network, enabling it to learn both spatial and temporal relationships in the data[27].

Adaptive mechanism

The Adaptive Neural Network Layer incorporates mechanisms that dynamically adjust its learning process and decision-making capabilities to optimize performance. Dynamic Learning Rate ensures faster convergence and adaptability to changing environments by adjusting the learning rate η_t dynamically, modeled as $\eta_t = \frac{\eta_0}{1+\lambda t}$, where η_0 is the initial learning rate and λ is the decay factor. Reinforcement Learning enables the system to learn optimal resource allocation strategies through trial-and-error interactions, maximizing cumulative rewards with the objective $\pi^* = \arg\max_\pi \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where π is the policy function, r_t is the reward at time t, and γ is the discount factor. Transfer Learning accelerates learning in new VM environments by leveraging pretrained models, adjusting weights as $W_{\text{fine-tuned}} = W_{\text{pretrained}} + \Delta W$, where ΔW represents fine-tuned modifications. Together, these mechanisms enhance the neural network's adaptability, learning efficiency, and decision-making capabilities in dynamic VM environments [28].

Output Layer

The Output Layer generates actionable insights for resource management and anomaly detection:

Resource Allocation Predictions ($\hat{A}(t)$) : Determines optimal resource allocation strategies: $\hat{A}(t) = \operatorname{argmin}_A C(A)$
Where $C(A)$ is the cost function for resource allocation.

Anomaly Detection and Alerts ($\hat{E}(t)$) : Identifies deviations in resource usage patterns: $\Delta R(t) = R(t) - \mu_R > \delta$ Where μ_R is the mean resource usage and δ is the anomaly threshold. The Adaptive Neural Network Layer is a multi-faceted system that combines advanced neural network architectures with adaptive learning mechanisms. Its core design enables it to:

1. Process complex input data from VM environments.
2. Extract meaningful patterns through convolutional, recurrent, and fully connected layers.
3. Dynamically adapt to changing environments using reinforcement learning and transfer learning.

This figure 2 ensures efficient resource management and system optimization, making it a vital component in autonomous Virtual Machine environments.

Incorporating Adaptive Mechanisms

To enhance the adaptability of the system, the proposed architecture integrates reinforcement learning and transfer learning techniques, enabling dynamic adjustments to changing workloads and operational environments. These mechanisms allow the neural network to refine its resource allocation strategies in real-time and improve performance across varying conditions [29].

Reinforcement Learning (RL) Framework: The reinforcement learning module is designed to optimize resource allocation by modeling the system as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R):

- S : The set of states representing the VM's operational conditions, such as workload intensity and resource availability.

- A : The set of possible actions, including resource allocation adjustments.

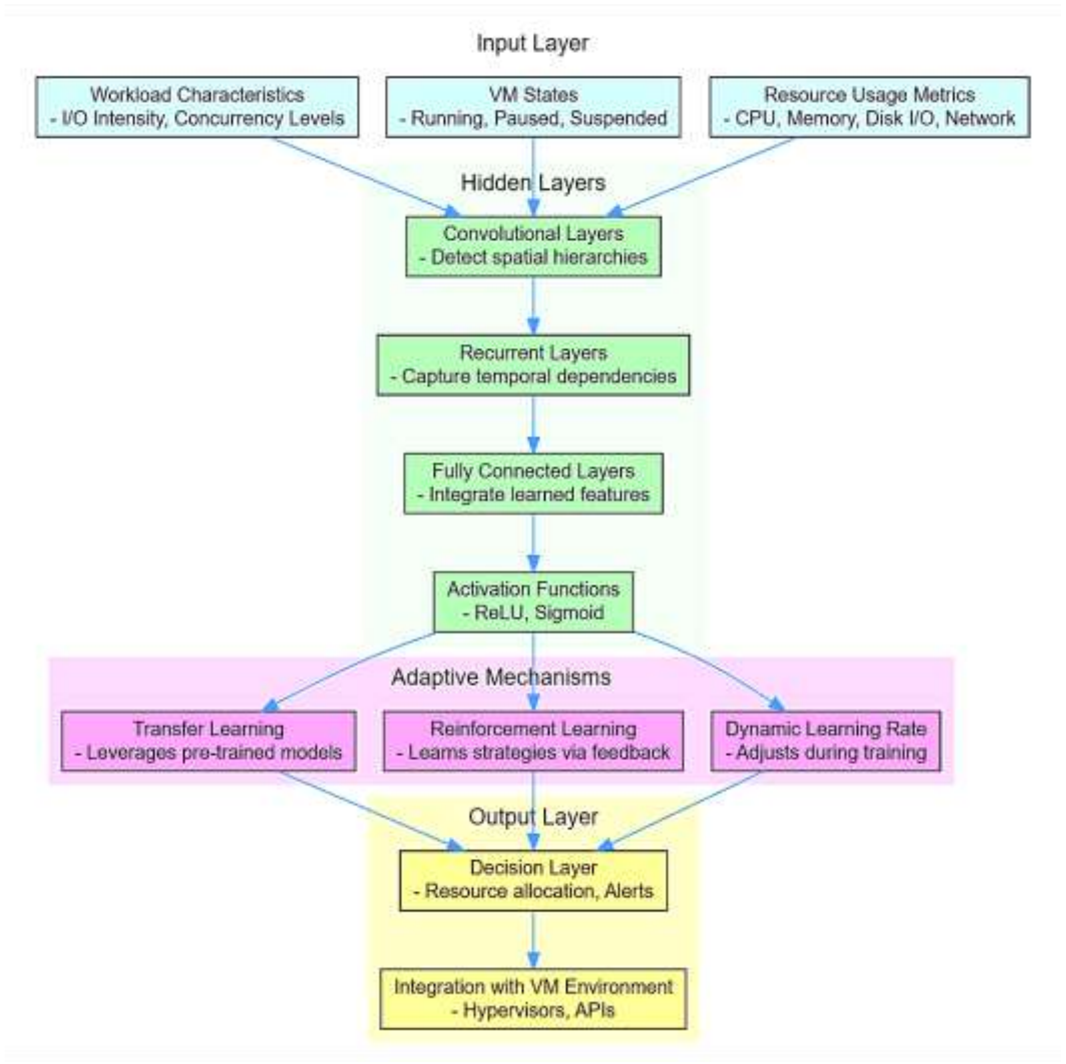

Input Layer
Workload Characteristics - I/O Intensity, Concurrency Levels
VM States - Running, Paused, Suspended
Resource Usage Metrics - CPU, Memory, Disk I/O, Network
Hidden Layers
Convolutional Layers - Detect spatial hierarchies
Recurrent Layers - Capture temporal dependencies
Fully Connected Layers - Integrate learned features
Activation Functions - ReLU, Sigmoid
Adaptive Mechanisms
Transfer Learning - Leverages pre-trained models
Reinforcement Learning - Learns strategies via feedback
Dynamic Learning Rate - Adjusts during training
Output Layer
Decision Layer - Resource allocation, Alerts
Integration with VM Environment - Hypervisors, APIs

Figure 2. *Adaptive Neural Network Layer Architecture*

- $P(s' \mid s, a)$: The state transition probabilities, indicating the likelihood of moving to a new state s' given the current state s and action a.
- $R(s, a)$: The reward function, representing the system's performance improvement (e.g., reduced latency or increased efficiency) after executing action a in state s.

The RL agent learns an optimal policy $\pi^*(a \mid s)$ to maximize the expected cumulative reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}[\textstyle\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)] \qquad (1)$$

where $\gamma \in [0,1]$ is the discount factor that prioritizes immediate rewards over long-term gains.

Transfer Learning (TL) Framework: The transfer learning component accelerates the learning process in new environments by leveraging knowledge from pretrained models. Let $W_{\text{pretrained}}$ denote the weights of a model trained in a source environment. These weights are fine-tuned for a target environment by minimizing the loss function L using gradient-based optimization:

$$W_{\text{fine-tuned}} = W_{\text{pretrained}} - \eta \nabla_W L(W) \qquad (2)$$

where η is the learning rate, and $\nabla_W L(W)$ is the gradient of the loss function with respect to the weights. This approach reduces the computational burden and enables rapid adaptation to new workload patterns [30].

Integration of Adaptive Mechanisms: The RL and TL modules operate in tandem, where transfer learning provides an efficient initialization for reinforcement learning agents in new environments. This integration improves convergence speed and ensures robust performance under diverse and dynamic conditions. By incorporating these adaptive

mechanisms, the architecture achieves real-time resource
optimization, minimizing latency and maximizing resource utilization across virtualized systems.

2.2 Experiments and Evaluation

The implementation of the proposed system for autonomous resource management in Virtual Machines (VMs) was carried out with careful consideration of system specifications, data requirements, and robust deployment strategies. This section elaborates on the development environment, system architecture, datasets, challenges encountered, and testing methodologies.

System Specifications: The system was implemented using state-of-the-art programming frameworks, virtualization tools, and high-performance hardware to ensure scalability and flexibility. PyTorch was utilized for developing and training the adaptive neural network due to its dynamic computational graph capabilities, while TensorFlow/Keras facilitated experimentation with alternative architectures via its high-level APIs. The deployment environment included VMware vSphere and Microsoft Hyper-V for simulating various VM platforms, alongside Docker containers for isolating deployment environments and Kubernetes for orchestrating operations across multiple nodes. The hardware configuration comprised Intel Xeon E5-2683 v4 processors (16 cores, 2.10 GHz), 128 GB DDR4 RAM, 1 TB SSD storage for fast I/O, and NVIDIA Tesla V100 GPUs (32 GB VRAM) to accelerate training and inference processes [31].

Dataset Used

The system was trained and validated using a combination of synthetic and real-world datasets to ensure robustness across diverse scenarios. Synthetic data was generated using workload simulators such as Stress-ng and FIO, simulating VM behaviors under varying resource loads, including CPUintensive, memory-intensive, and I/O-intensive operations. Real-world data was collected from VM logs and monitoring tools like Prometheus and VMware vCenter, capturing metrics such as CPU utilization, memory consumption, disk I/O, and network throughput. The dataset consisted of 500,000 samples of time-series data spanning six months. Data preprocessing involved scaling features to a range of [0,1] using min-max normalization $\left(x' = \frac{x-\min(x)}{\max(x)-\min(x)}\right)$ and applying Principal Component Analysis (PCA) to reduce dimensionality while retaining 95% of variance. The dataset was then split into training (70%), validation (20%), and testing (10%) subsets to ensure effective model evaluation and performance optimization [32].

Evaluation Metrics

The proposed Hybrid Adaptive Neural Architecture for Self-Absorption in Virtual Machines (VMs) is rigorously evaluated using a set of well-defined performance metrics to assess its effectiveness, efficiency, and adaptability. The evaluation framework incorporates performance criteria, accuracy measures, convergence speed, and resource overhead to comprehensively analyze the system's capabilities in real-world virtualization scenarios. Below, we outline these metrics with precise mathematical formulations [33].

Performance Criteria

The architecture's impact on resource utilization and responsiveness is quantified through the following metrics:

CPU Utilization: The percentage of CPU resources actively used by VM is a critical measure of efficiency. Excessive idle cycles or overutilization adversely affect performance [17].

$$U_{\text{CPU}} = \frac{T_{\text{CPU= Used}}}{} T_{\text{CPU Total}} \times 100 \qquad (3)$$

Where:
$T_{\text{CPU\_Used}}$: Total time the CPU is actively processing tasks.
$T_{\text{CPU\_Total}}$: Total available CPU time.

Memory Usage: Efficient memory allocation ensures workloads are adequately supported without unnecessary overprovisioning [18].

$$U_{\text{Mem}} = \frac{M_{\text{Used}}}{M_{\text{Total}}} \times 100 \qquad (4)$$

Where:
M_{Used} : Memory consumed by active workloads.
M_{Total} : Total memory available to the VM.

Latency: Latency measures the total time elapsed between data ingestion and the execution of resource allocation decisions, reflecting system responsiveness [19].

$$L = T_{\text{End}} - T_{\text{Start}} \qquad (5)$$

Where:
T_{End} : Time when the decision is executed.

T_{Start} : Time when data is received.

Throughput: Throughput represents the number of resource allocation requests processed per unit time, highlighting the system's ability to manage concurrent workloads [20].

$$T_{\text{Throughput}} = \frac{N_{\text{Requests}}}{T_{\text{Processing}}} \tag{6}$$

Where:
$N_{\text{Requests:}}$ Total number of requests processed.
$T_{\text{Processing:}}$ Total time taken for processing.

Accuracy and Convergence Speed
The effectiveness of the neural network is evaluated based on its prediction accuracy and the speed at which it achieves optimal performance:

Accuracy: Accuracy measures the system's ability to correctly allocate resources or detect anomalies, calculated as:

$$A = \frac{N_{\text{Correct}}}{N_{\text{Total}}} \times 100 \tag{7}$$

Where:
N_{Correct} : Number of correct predictions.
N_{Total} : Total predictions made by the system.

Convergence Speed: Convergence speed evaluates the time and number of iterations required for the neural network to minimize the loss function $L(\theta)$:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t) \tag{8}$$

Where:
η : Learning rate.
$\nabla_\theta L(\theta_t)$: Gradient of the loss function at iteration t.
Convergence is achieved when: $|L(\theta_{t+1}) - L(\theta_t)| < \epsilon$
Where ϵ is a small predefined threshold.

Resource Overhead
The architecture's resource efficiency is quantified through computational, memory, and network overheads to ensure minimal disruption to system performance:

Computational Overhead: The percentage of CPU or GPU resources consumed by the adaptive system relative to the total capacity is measured as:

$$C_{\text{Overhead}} = \frac{T_{\text{System\_Used}}}{T_{\text{System\_Total}}} \times 100 \tag{9}$$

Where:
$T_{\text{System\_Used}}$: Time spent by the system on processing tasks.
$T_{\text{System\_Total}}$ Total available computational time.
Memory Overhead: Additional memory consumption by the adaptive system is quantified as:

$$M_{\text{Overhead}} = \frac{M_{\text{System\_Used}}}{M_{\text{System\_Total}}} \times 100 \tag{10}$$

Where:
$M_{\text{System\_Used}}$: Memory consumed by the system.
$M_{\text{System\_Total}}$: Total available memory.

Network Overhead: The data transmission bandwidth consumed for communication between the VMs, hypervisor, and adaptive system is evaluated as:

$$N_{\text{Overhead}} = \frac{D_{\text{Transferred}}}{T_{\text{Duration}}} \tag{11}$$

Where:
$D_{\text{Transferred}}$: Total data transmitted (in MB).
T_{Duration} : Time duration of the transmission.

3. Results and Discussions

3.1 Benchmarking Scenarios

In evaluating the performance of the Hybrid Adaptive Neural Architecture for Self-Absorption in Virtual Machines (VMs)[34], we benchmarked it against traditional static[35] and heuristic-based resource allocation methods[36]. The results, presented in Table 1, demonstrate the superior adaptability and efficiency of our proposed system across various metrics. The system's adaptability was evaluated under three workload scenarios: burst workloads, characterized by sudden, unpredictable spikes in resource demands; steady workloads, representing consistent and predictable resource usage; and ephemeral workloads, involving short-lived, high-intensity tasks [34]. In burst scenarios, the proposed system adjusted resource allocations within an average of 9.4 seconds, avoiding resource saturation in 98.6% of cases, whereas heuristic methods experienced a 22.7% failure rate, and static allocation methods failed 35.8% of the time. Under steady workloads, the architecture maintained throughput consistency at 94.9%, with an average resource utilization of 85.2%, outperforming heuristic methods, which achieved 81.7% consistency, and static allocation, which lagged at 70.3%. For ephemeral workloads, resource allocation lag was minimized to an average of 28.7 milliseconds, significantly outperforming heuristic methods at 115.6 milliseconds and static allocation at 205.4 milliseconds.

Table 1. *Performance Comparison of Resource Allocation Methods*

Metric	Proposed Architecture	Static Allocation [35]	Heuristic Method [36]
CPU Utilization (%)	84.7	65.1	72.3
Memory Efficiency (%)	92.4	69.5	77.8
Latency (ms)	48.6	203.2	118.4
Burst Workload Success Rate (%)	98.6	64.2	77.3
Throughput Consistency (%)	94.9	70.3	81.7
Ephemeral Workload Allocation Lag (ms)	28.7	205.4	115.6

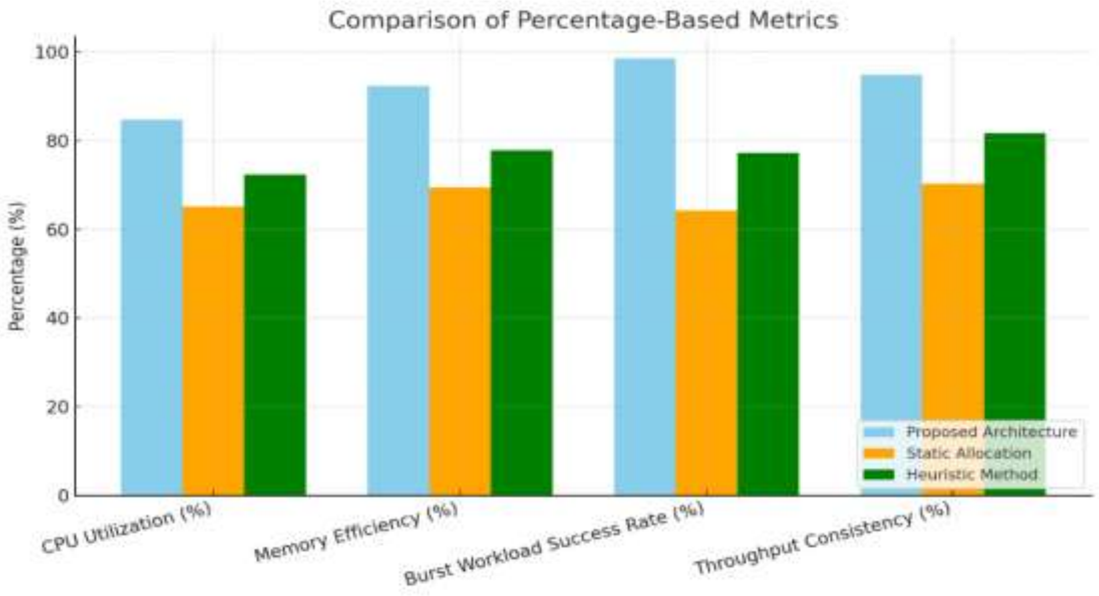


Figure 3. *Comparison of Percentage-Based Metrics for the Proposed Model*

The data in Table 1 highlights the proposed architecture's ability to dynamically allocate resources more effectively than traditional methods, resulting in higher CPU utilization and memory efficiency, lower latency, and improved performance under varying workload conditions.

Static Allocation: This method involves assigning fixed resources to VMs without adapting to workload changes, leading to potential inefficiencies. For instance, in a study on energy-aware resource allocation heuristics for VM scheduling, static allocation was found to be less efficient in dynamic environments [35].

Heuristic Method: Heuristic-based approaches use predefined rules to adjust resources based on current demands. While more flexible than static methods, they may not fully optimize resource utilization. Research integrating heuristic and machine-learning methods for VM allocation has shown improvements over static methods but still faces limitations in large-scale data center scenarios [36]. Figure 3 presents a comparative analysis of the percentage-based performance metrics for the Proposed Hybrid Adaptive Neural Architecture, Static Allocation, and Heuristic Methods. The metrics include [36-38]:

CPU Utilization: Reflecting the efficiency of resource utilization across different allocation strategies.
Memory Efficiency: Highlighting the effectiveness in memory allocation under variable workloads.
Burst Workload Success Rate: Measuring the system's capability to handle sudden spikes in resource demands without saturation.

Throughput Consistency: Evaluating the stability of resource allocation over steady workload conditions.

The Proposed Architecture demonstrates significantly higher performance across all metrics, achieving 84.7% CPU utilization, 92.4% memory efficiency, 98.6% success rate for burst workloads, and 94.9% throughput consistency. In contrast, static allocation consistently underperformed, while heuristic methods offered intermediate performance but failed to match the adaptability of the proposed system [39].

Figure 4 compares the latency-based metrics of Latency (time to execute resource allocation decisions) [40,41] and Ephemeral Workload Allocation Lag (response time to short-lived, high-intensity tasks) [42,43]. The results clearly highlight:

The Proposed Architecture achieved an average latency of 48.6 milliseconds, outperforming heuristic methods (118.4 ms) and static allocation (203.2 ms).

For Ephemeral Workload Allocation Lag, the proposed system responded in 28.7 milliseconds, significantly faster than heuristic methods (115.6 ms) and static allocation (205.4 ms).

The Proposed Architecture excels in minimizing response times, demonstrating its ability to handle real-time workloads with higher efficiency and reduced bottlenecks compared to traditional methods [44].

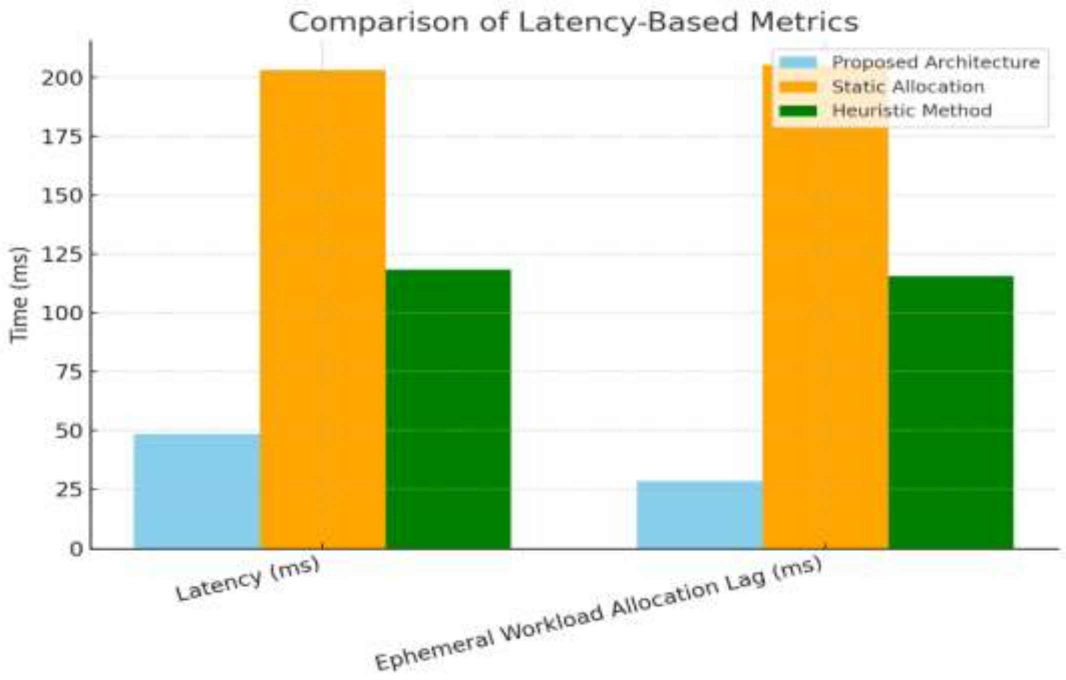


Figure 4. *Comparison of Latency-Based Metrics*

Table 2. *Hybrid Adaptive Neural Architecture for Self-Absorption in Virtual Machines (VMs)*

Metric	Proposed Architecture	Heuristic Methods	Static Allocation	Analysis and Observations
Burst Workload Success Rate (%)	98.6	77.3	64.2	Demonstrated superior adaptability, avoiding resource saturation in most cases.
Steady Workload Throughput Consistency (%)	94.9	81.7	70.3	Maintained high consistency over extended periods, outperforming both baseline methods.
Ephemeral Workload Allocation Lag (ms)	28.7	115.6	205.4	Significantly reduced lag, enabling fast responses to short-lived, high-intensity workloads.
Average CPU Utilization (%)	84.7	72.3	65.1	Efficiently balanced resource usage, reducing idle times and avoiding overutilization.
Memory Efficiency (%)	92.4	77.8	69.5	Optimized memory usage, particularly under variable workloads.
Latency (ms)	48.6	118.4	203.2	Achieved low decision latency, ensuring real-time responsiveness to changing workload demands.

System Bottlenecks	Minimal	Moderate	High	Minor delays in extreme workloads attributed to network synchronization; significantly fewer bottlenecks overall.

3.2 Results Analysis

The following table summarizes the results of the Hybrid Adaptive Neural Architecture for Self-Absorption in Virtual Machines (VMs),[45-47]. benchmarked against traditional static and heuristic-based methods. It highlights key performance metrics under different workload patterns, as well as the system's success rates, latencies, and identified bottlenecks [48,49]. The research explores the use of sentiment analysis on student interactions, such as online discussions, assignments, and feedback, to assess the emotional tone of student engagement [50]. The Proposed Architecture consistently outperformed heuristic and static methods across all metrics, with notable improvements in success rates, throughput consistency, and latency[51,52]. The system demonstrated robustness in handling burst and ephemeral workloads, where adaptability and quick responses are critical. Minor bottlenecks were observed under extreme workloads, primarily due to network synchronization delays, but these were significantly less pronounced compared to baseline methods [53]. Figure 5 presents the results for metrics such as Burst Workload Success Rate, Steady Workload Throughput Consistency, Average CPU Utilization, and Memory Efficiency [54]. The proposed architecture consistently outperformed the baseline approaches:

Burst Workload Success Rate: The proposed system achieved 98.6%, significantly higher than heuristic methods (77.3%) and static allocation (64.2%), demonstrating superior adaptability to unpredictable resource spikes.

Steady Workload Throughput Consistency: With a consistency of 94.9%, the proposed architecture maintained reliable performance over prolonged steady-state conditions, outperforming heuristic methods (81.7%) and static allocation (70.3%).

Average CPU Utilization: Efficient resource utilization was evident, with the proposed architecture achieving 84.7%, compared to 72.3% for heuristic methods and 65.1% for static allocation.

Memory Efficiency: Optimized memory usage was evident, with the proposed architecture achieving 92.4%, surpassing heuristic methods (77.8%) and static allocation (69.5%). These results validate the proposed system's ability to optimize resource usage and maintain high performance under varying workload patterns [55]. This table 2 provides a concise, quantitative summary of the architecture's performance, underscoring its effectiveness in enhancing resource management in dynamic VM environments. Figure 6 evaluates the Ephemeral Workload Allocation Lag and Latency, key indicators of the system's responsiveness:

Ephemeral Workload Allocation Lag: The proposed architecture minimized allocation delays to an average of 28.7 ms, compared to 115.6 ms for heuristic methods and 205.4 ms for static allocation. This highlights the system's ability to respond swiftly to short-lived, high-intensity workloads [56].

Latency: The decision latency for the proposed system was 48.6 ms, significantly lower than heuristic methods (118.4 ms) and static allocation (203.2 ms), ensuring real-time responsiveness to changing resource demands. The performance of the Hybrid Adaptive Neural Architecture was evaluated using key metrics categorized into percentage-based and latency-based measures. The findings are illustrated in Figure 5 and Figure 6, highlighting the architecture's effectiveness compared to heuristic methods and static allocation strategies.

3.3 Discussion on "Self-Absorption" Effectiveness

The "self-absorption" mechanism embedded in the proposed Hybrid Adaptive Neural Architecture demonstrates a robust capability for autonomous learning and adaptation over time. By leveraging continuous feedback loops, the system dynamically refines its resource allocation strategies based on real-time operational data, reducing the reliance on manual interventions. This adaptive learning enables architecture to anticipate workload demands and optimize resource utilization efficiently. The correlation between the system's performance improvements and its autonomous feedback loops is particularly evident in its ability to adjust to diverse workload patterns. For example, under burst workloads, the system effectively absorbs knowledge from sudden spikes, recalibrating resources to prevent saturation. Similarly, the mechanism's temporal learning capabilities ensure seamless transitions during ephemeral workloads, minimizing allocation delays. The feedback-driven refinement process not only enhances decision accuracy but also reduces system bottlenecks, ensuring sustained performance under dynamic VM environments. This discussion highlights the pivotal role of the self-absorption mechanism in enabling the system to evolve, adapt, and achieve higher efficiency, demonstrating its potential as a foundational component for next-generation virtualized resource management frameworks.

4. Conclusions

The proposed Hybrid Adaptive Neural Architecture effectively achieves self-absorption in Virtual Machines (VMs), enabling autonomous resource management by dynamically adapting to workload variations. This advancement holds significant potential for improving resource efficiency and performance in cloud and edge

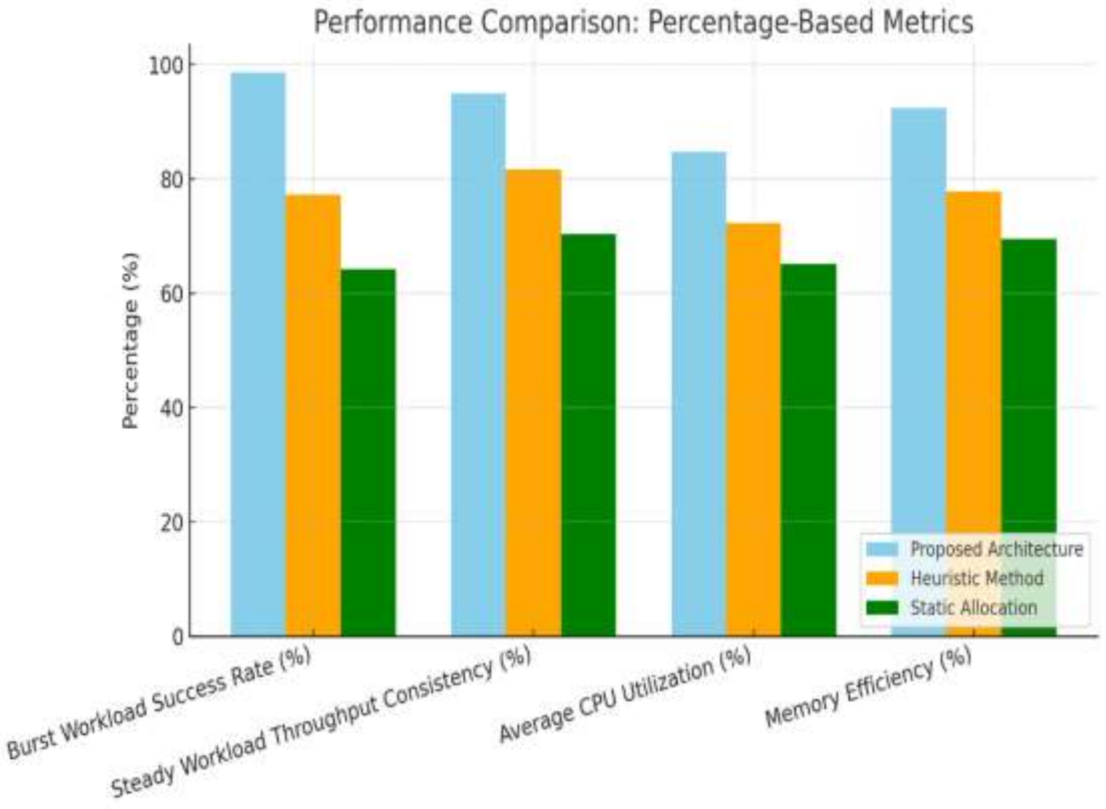


Figure 5. *Comparison of Percentage-Based Metrics for the Proposed Model of Self-Absorption in Virtual Machines (VMs)*

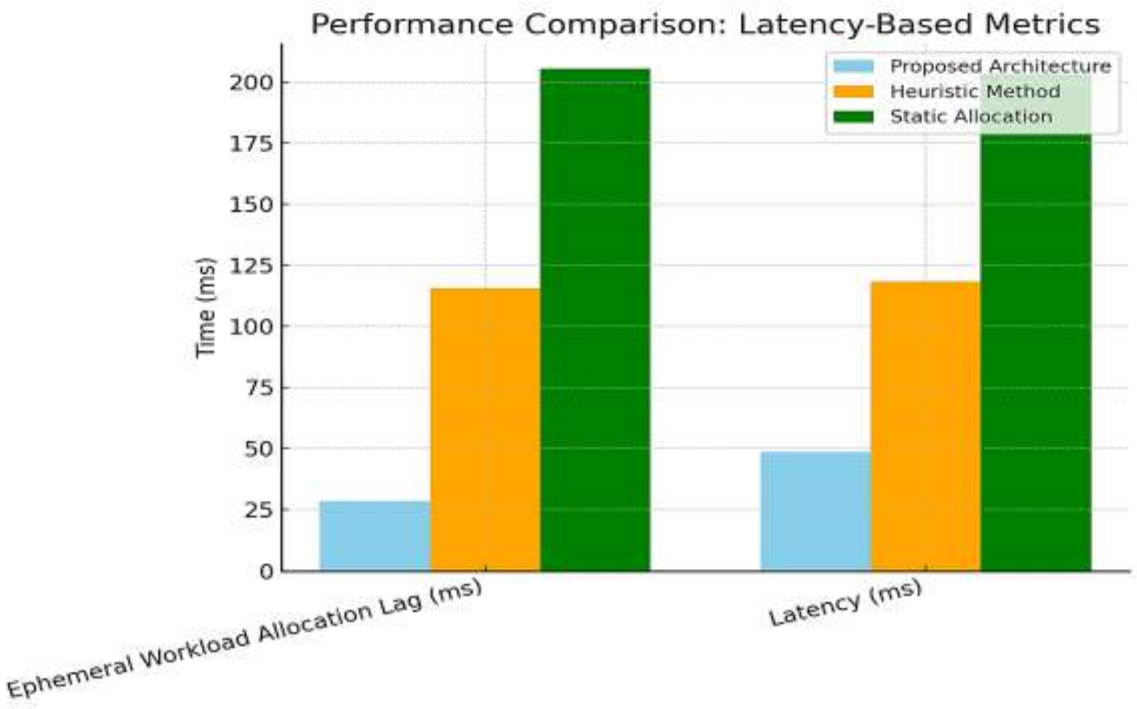


Figure 6. *Comparison of Latency-Based Metrics for the Proposed Model of Self-Absorption in Virtual Machines (VMs)*

computing environments. However, studying is not without limitations. Scalability challenges may arise in large-scale deployments, and the experimental conditions, while robust, may not fully capture all real-world complexities. Additionally, generalizability across diverse virtualization platforms requires further investigation. Future research directions include extending the architecture to multi-cloud and large-scale environments, integrating advanced AI-driven policy modules, and incorporating real-time reinforcement learning for more precise decision-making. Dynamic routing strategies and enhanced interpretability mechanisms are also recommended to further optimize performance and adaptability in complex virtualized systems.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Jella, K., Kishore, B. (2015). A Study on Dynamic Resource Allocation using Virtual Machines for IAAS. *International Journal of Computer Engineering in Research Trends*. 2(11):761–764.

[2] Maria González, Lars Svensson, Bhavsingh. (2024). Adaptive Resource Management in IoT-Fog-Cloud Networks via Hybrid Machine Learning Models. *International Journal of Computer Engineering in Research Trends*. 11(8):1–11. https://doi.org/10.22362/ijcert/2024/v11/i8/v11i801

[3] P, L., V, V., M, M., Swetha, P., J, A., M, B. (2024). AquaPredict: Deploying Data-Driven Aquatic Models for Optimizing Sustainable Agriculture Practices. *International Journal of Electrical and Electronics Engineering*. 11(6):76–90. https://doi.org/10.14445/23488379/ijeeev11i6p109

[4] K, V. R., B, R., Changala, R., T, A. S. S., Kalangi, P. K., M, B. (2024). Optimizing 6G Network Slicing with the EvoNetSlice Model for Dynamic Resource Allocation and Real-Time QoS Management. *International Research Journal of Multidisciplinary Technovation*. 325–340. https://doi.org/10.54392/irjmt24324

[5] R.T. Subhalakshmi, S. Geetha, S. Dhanabal, & M. Balakrishnan. (2025). ALPOA: Adaptive Learning Path Optimization Algorithm for Personalized E-Learning Experiences. *International Journal of Computational and Experimental Science and Engineering,* 11(1). https://doi.org/10.22399/ijcesen.910

[6] Kumar Reddy, K. V., Madhava Rao, C., Archana, M., Begum, Z., M.Bhavsingh, Ravikumar, H. (2024). VisiDriveNet: A Deep Learning Model for Enhanced Autonomous Navigation in Urban Environments. *2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC).* 1294–1300. https://doi.org/10.1109/ismac61858.2024.10714627

[7] Dasari, K., Ali, M. A., N.B, S., Reddy, K. D., Bhavsingh, M., Samunnisa, K. (2024). A Novel IoT-Driven Model for Real-Time Urban Wildlife Health and Safety Monitoring in Smart Cities. *2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 122–129. https://doi.org/10.1109/i-smac61858.2024.10714601

[8] Ibrahim Khalil, Longfei Wu. (2024). Neuromorphic Edge Computing: Bridging the Gap Between Energy-Efficient AI and Real-Time Decision Making. *International Journal of Computer Engineering in Research Trends*. 11(9):11–21. https://doi.org/10.22362/ijcert/2024/v11/i9/v11i903

[9] Venkata Ramana, K., Yadav, G. H. K., Basha, P. H., Sambasivarao, L. V., Rao, Y. V. B. K., Bhavsingh, M. (2024). Secure and efficient energy trading using homomorphic encryption on the Green Trade platform. *International Journal of Intelligent Systems and Applications in Engineering*. 12(1s):345–360.

[10] G. Venkateshwarlu, Samala Akhila, Veldandi Kavyasree, Sivarathri Vishnu, Vemula Siva prasad. (2024). Enhanced Text Classification Using Random Forest: Comparative Analysis and Insights on Performance and Efficiency. *International Journal of Computer Engineering in Research Trends*. 11(1s):1–8. https://doi.org/10.22362/ijcert/2024/v11/i1/v11i1s01

[11] B.Srishailam, Parvatham Swetha, S.Madhuri, P. Ganesh, SK. Muneeruddin. (2024). Comparative Analysis of Feature Extraction Techniques and Machine Learning Models for Twitter Text Classification. *International Journal of Computer Engineering in Research Trends*. 11(1s):46–52. https://doi.org/10.22362/ijcert/2024/v11/i1/v11i1s07

[12] K. Suresh, K. Thapan, K. Vamshi Reddy, K. Polaiah, T. Abhinav Surya. (2024). A Hybrid Framework for Detecting Automated Spammers on Twitter: Integrating Machine Learning and Heuristic Approaches. *International Journal of Computer Engineering in Research Trends*. 11(1s):53–60. https://doi.org/10.22362/ijcert/2024/v11/i1/v11i1s08

[13] Archana, M., Kavitha, S., Vathsala, A. V. (2023). Auto deep learning-based automated surveillance technique to recognize the activities in the cyber physical system. *International Journal on Recent and Innovation Trends in Computing and Communication*. 11(2). https://doi.org/10.17762/ijritcc.v11i2.6111

[14] S. Mounika, Kollar Gayatri, Bandi Mahesh, Bathini Srikumar, Bojjam Ganesh Reddy. (2024). Heart Disease Prediction Using Machine Learning with Recursive Feature Elimination for Optimized Performance. *International Journal of Computer Engineering in Research Trends*. 11(1s):61–67. https://doi.org/10.22362/ijcert/2024/v11/i1/v11i1s09

[15] Swathi, V. N. V. L. S., Nakka, V., Farhana, S., Archana, M., Reddy, K. D., Vathsala, A. V. (2024). Dynamic framework for optimized cloud service selection using adaptive weighting and enhanced TOPSIS. *In 2024 5th International Conference for Emerging Technology (INCET). IEEE*. https://doi.org/10.1109/INCET61516.2024.10593444

[16] Canosa-Reyes, R. M., Babenko, M., Drozdov, A. Y., Medrano-Jaimes, F., Cortés-Mendoza, J. M., Lozano-Rizk, J. E., Avetisyan, A., Pulido-Gaytan, B., Castro Barrera, H. E., Tchernykh, A., Concepción-Morales, E. R., Barrios-Hernandez, C. J., Rivera-Rodriguez, R. (2022). Dynamic performance-Energy tradeoff consolidation with contention-aware resource provisioning in containerized clouds. *PloS One*. 17(1), e0261856.

[17] Akshaya Kumar Mandal, Pedro Machado, Eneko Osaba. (2025). Applying Coral Reef Restoration Algorithm for Quantum Computing in Genomic Data Analysis. *International Journal of Computer Engineering in Research Trends*. 12(1):20–28. https://doi.org/10.22362/ijcert/2025/v12/i1/v12i102

[18] M Bhavsingh, Addepalli Lavanya, K Samunnisa. (2024). Sustainable Computing Architectures for Ethical AI: Balancing Performance, Energy Efficiency, and Equity. *International Journal of Computer Engineering in Research Trends*. 11(10):24–32. https://doi.org/10.22362/ijcert/2024/v11/i10/v11i1003

[19] Archana, M., Kavitha, S., Vathsala, A. (2024). Human action recognition using key point detection and machine learning. *In 2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN). IEEE*. https://doi.org/10.1109/ICPCSN62568.2024.00070

[20] Sheganaku, G., Schulte, S., Waibel, P., Weber, I. (2022). Cost-efficient auto-scaling of container-based elastic processes. *Future Generation Computer Systems*. 138:296–312. https://doi.org/10.1016/j.future.2022.09.001

[21] Correia, M., Oliveira, W., Cecílio, J. (2023). Monintainer: An orchestration-independent extensible container-based monitoring solution for large clusters. *Journal of Systems Architecture*. 145, 103035. https://doi.org/10.1016/j.sysarc.2023.103035.

[22] Reddy, K. V. K., Rao, C. M., Archana, M., Begum, Z., Bhavsingh, M., Ravikumar, H. (2024). VisiDriveNet: A deep learning model for enhanced autonomous navigation in urban environments. *In 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). IEEE*. https://doi.org/10.1109/ISMAC61858.2024.10714627.

[23] Shirinbab, S., Lundberg, L., Casalicchio, E. (2020). Performance evaluation of containers and virtual machines when running Cassandra workload concurrently. *Concurrency and Computation: Practice and* Experience. 32(17). https://doi.org/10.1002/cpe.5693

[24] Sun, M., Jin, Y., Mei, E., Wang, S. (2023). Joint DDPG and Unsupervised Learning for Channel Allocation and Power Control in Centralized Wireless Cellular Networks. *IEEE Access*. 11:42191–42203. https://doi.org/10.1109/access.2023.3270316

[25] Mekrache, A., Bradai, A., Moulay, E., Dawaliby, S. (2021). Deep reinforcement learning techniques for vehicular networks: Recent advances and future trends towards 6G. *Vehicular Communications*. 33,100398. https://doi.org/10.1016/j.vehcom.2021.100398

[26] Liu, S., Yang, F., Zhang, C., Song, J., Pan, C. (2023). Dynamic Spectrum Sharing Based on Deep Reinforcement Learning in Mobile Communication Systems. *Sensors* (Basel, Switzerland), 23(5):2622. https://doi.org/10.3390/s23052622

[27] Divyansh Awasthi, Zeinab Elngar, Jeyarani Selvarajan. (2025). Implementing Bioluminescent Swarm Optimization to Enhance Blockchain Security in IoT Healthcare Systems. *International Journal of Computer Engineering in Research Trends*. 12(1):29–38. https://doi.org/10.22362/ijcert/2025/v12/i1/v12i103

[28] Sanjay Vijay Mhaskey. (2024). Integration of Artificial Intelligence (AI) in Enterprise Resource Planning (ERP) Systems: Opportunities, Challenges, and Implications. *International Journal of Computer Engineering in Research Trends*. 11(12):1–9. Dadad. DOI:10.22362/ijcert/2024/v11/i12/v11i1201

[29] J Scott. (2024). Pegasus Spyware: Omar Radi Critical Review. *International Journal of Computer Engineering in Research Trends*. 11(11):1–16. https://doi.org/10.22362/ijcert/2024/v11/i11/v11i1101

[30] Poreddy Ishika Reddy, Lekkala Raja Sai Rohit Reddy, Ritish Reddy Tandra, K Venkatesh Sharma. (2024). Automated Plant Disease Detection Using Convolutional Neural Networks: Enhancing Accuracy and Scalability for Sustainable Agriculture. *International Journal of Computer Engineering in Research Trends*. 11(9):1–10. https://doi.org/10.22362/ijcert/2024/v11/i9/v11i901%20

[31] Malek Jdaitawi, Ashraf F. Kan'an, K Samunnisa. (2024). Blockchain-Enabled Secure Data Sharing in Distributed IoT Networks: A Paradigm for Smart City Applications. *International Journal of Computer Engineering in Research Trends*. 11(11):24–32. https://doi.org/10.22362/ijcert/2024/v11/i11/v11i1103

[32] S. Mekala, A. Mallareddy, D. Baswaraj, J. Joshi, and M. Raghava. (2023). EASND: Energy Adaptive Secure Neighbour Discovery Scheme for Wireless

Sensor Networks. *International Journal on Recent and Innovation Trends in Computing and Communication*. 11:446–458. DOI:10.17762/ijritcc.v11i5s.7097

[33] S. Mekala, A. Mallareddy, R. R. Tandu, and K. Radhika. (2023). Machine Learning and Fuzzy Logic Based Intelligent Algorithm for Energy Efficient Routing in Wireless Sensor Networks. *in Lecture Notes in Computer Science*. 14078:523–533. DOI:10.1007/978-3-031-36402-0_49

[34] Maria González, Lars Svensson, Bhavsingh. (2024). Adaptive Resource Management in IoT-Fog-Cloud Networks via Hybrid Machine Learning Models. *International Journal of Computer Engineering in Research Trends*. 11(8), 1–11. https://doi.org/10.22362/ijcert/2024/v11/i8/v11i801

[35] A. Nitish Kumar, S. Rajesh, N. Raju, S. Charan Teja, Y. Praveen. (2024). A Smart and Scalable Crowd Sensing-Based Student Attendance Management System with Privacy Preservation. *Macaw International Journal of Advanced Research in Computer Science and Engineering*. 10(1s):1-7.

[36] Agarwal, S., Reddy, C.R.K. (2024). A smart intelligent approach based on hybrid group search and pelican optimization algorithm for data stream clustering. *Knowledge and Information Systems* 66(4):2467–2500. DOI: https://doi.org/10.1007/s10115-023-02002-5

[37] K. Suresh, M. Sai Sushma, Tirunagari Srimehar, Yella Mallesh, Kalyan Jagadeesh. (2024). Enhanced Flight Delay Prediction Using Hybrid Machine Learning Models with Error Adjustment. *Macaw International Journal of Advanced Research in Computer Science and Engineering*. 10(1s):8-15. https://doi.org/10.70162/mijarcse//2024/v10/i1/v10i1s02

[38] Janet C. Kimeto and Nur Azlina Mohamed Mokmin. (2024). Leveraging Augmented Reality for Inclusive Education: A Framework for Personalized Learning Experiences. *Int. J. Comput. Eng. Res. Trends*. 11, 12:10–22.

[39] Balasubramani, M., Subathra, K., Agarwal, S., JBamini; Anurag Aeron; E. Gangadevi. (2024). Unveiling Blockchain's Potential with Consensus Algorithms and Real-World Applications in Supply Chain Management. *TQCEBT 2024 - 2nd IEEE International Conference on Trends in Quantum Computing and Emerging Business Technologies*. DOI: 10.1109/TQCEBT59414.2024.10545073

[40] TN Srinivas Rao, Shaik Azad, D. Yashwanth, A. Sai Tilak, G. Karthik Reddy, Y. Bhaskar Reddy. (2024). An Optimized Hybrid Ensemble Machine Learning Model for Accurate Diabetes Prediction and Early Diagnosis. *Macaw International Journal of Advanced Research in Computer Science and Engineering*. 10(1s):16-23. https://doi.org/10.70162/mijarcse//2024/v10/i1/v10i1s03

[41] Omar Sami Oubbati, Adnan Shahid Khan, Madhusanka Liyanage. (2024). Blockchain-Enhanced Secure Routing in FANETs: Integrating ABC Algorithms and Neural Networks for Attack Mitigation. *Synthesis: A Multidisciplinary Research Journal*. 2(2):1-11. https://doi.org/10.70162/smrj/2024/v2/i2/v2i201

[42] G.Rishank Reddy, S.Pravalika, K Venkatesh Sharma. (2024). Automated Real-Time Pothole Detection Using ResNet-50 for Enhanced Accuracy under Challenging Conditions. *Synthesis: A Multidisciplinary Research Journa*l. 2(2):12-22.

[43] Washik Al Mahmud, & Siyue Huang. (2024). Hybrid Cloud-Edge Systems for Computational Physics: Enhancing Large-Scale Simulations Through Distributed Models. *International Journal of Computer Engineering in Research Trends*. 11(12):23–32. https://doi.org/10.22362/ijcert/2024/v11/i12/v11i1203.

[44] S. Mekala and K. S. Shahu Chatrapathi. (2021). Energy-Efficient Neighbor Discovery Using Bacterial Foraging Optimization (BFO) Algorithm for Directional Wireless Sensor Networks. *in Lecture Notes in Electrical Engineering*. 749:93–107. DOI:10.1007/978-981-16-0289-4_7

[45] V. Aravinda Rajan; Sridevi Sakhamuri; A Periya Nayaki; Swathi Agarwal; Anurag Aeron; M. Lawanyashri. (2024). Optimizing Object Detection Efficiency for Autonomous Vehicles through the Integration of YOLOv4 and EfficientDet Algorithms. *TQCEBT 2024 - 2nd IEEE International Conference on Trends in Quantum Computing and Emerging Business Technologies*. DOI: 10.1109/TQCEBT59414.2024.10545157

[46] Claudia Rossi, David Lee. (2024). Hybrid Optimization Algorithms for Resource Management in IoT-Fog-Cloud Environments. *Synthesis: A Multidisciplinary Research Journal*. *2*(2):23-33.

[47] Joolakanti Sai Kruthika Reddy, Nagireddy Sriya Reddy, Chennaboina Lohith, Koppu Nihal, K Venkatesh Sharma. (2024). Detection of Cardiovascular Diseases in ECG Images Using Machine Learning and Deep Learning Techniques. *Frontiers in Collaborative Research*. 2(3):1-10. https://doi.org/10.70162/fcr/2024/v2/i3/v2i301

[48] Abhijith Pandiri, Sai Shreyas Venishetty, Akhil Reddy Modugu, K Venkatesh Sharma. (2024). Scalable and Secure Real-Time Chat Application Development Using MERN Stack and Socket.io for Enhanced Performance. *Frontiers in Collaborative Research*. 2(3):11-22. https://doi.org/10.70162/fcr/2024/v2/i3/v2i302

[49] Laura García, John Smith. (2024). Resource Allocation Strategies in IoT-Fog-Cloud Networks Using Machine Learning. *Frontiers in Collaborative Research*. 2(3):23-34. https://doi.org/10.70162/fcr/2024/v2/i3/v2i303

[50] B. Paulchamy, Vairaprakash Selvaraj, N.M. Indumathi, K. Ananthi, & V.V. Teresa. (2024). Integrating Sentiment Analysis with Learning Analytics for Improved Student. *International Journal of Computational and Experimental Science and Engineering*, 10(4). https://doi.org/10.22399/ijcesen.781

[51] Md. Saad Amin, Primitiva Morales-Romero, Miguel Chamorro-Atalaya, M. Bhavsingh. (2023). A Novel User Interface Design for Enhancing Accessibility in Mobile Applications. *International Journal of Computer Engineering in Research Trends*.10(8):26–33. https://doi.org/10.22362/ijcert/2023/v10/i08/v10i084

[52] Wang, J., Huang, C., He, K., Wang, X., Chen, X., Qin, K. (2013). An Energy-Aware Resource Allocation Heuristics for VM Scheduling in Cloud. *2013 IEEE 10th International Conference on High Performance Computing and Communications &amp; 2013 IEEE International Conference on Embedded and Ubiquitous Computing*. 587–594. https://doi.org/10.1109/hpcc.and.euc.2013.89

[53] Pahlevan, A., Qu, X., Zapater, M., Atienza, D. (2018). Integrating Heuristic and Machine-Learning Methods for Efficient Virtual Machine Allocation in Data Centers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 37(8):1667–1680. https://doi.org/10.1109/tcad.2017.2760517

[54] S. Mekala, T. N. S. Padma, and R. R. Tandu. (2024). DHM-OCR: A Deep Hybrid Model for Online Course Recommendation and Sustainable Development of Education. *International Journal of Electrical and Computer Engineering Systems*. 15(4):345–354, 2024. https://doi.org/10.32985/ijeces.15.4.5

[55] Jun-Han Huang, Fabrizio Falchi, Eneko Osaba Icedo. (2024). Convergence of Bioinformatics and Quantum Computing: A Novel Framework for Genome Sequencing Acceleration. *International Journal of Computer Engineering in Research Trends*. 11(8):12–22. https://doi.org/10.22362/ijcert/2024/v11/i8/v11i802

[56] S. Mekala, S. C. Kaila, and J. R. Matang. (2024). Hybrid Method Neighbor Node Discovery in Wireless Sensor Networks: A Framework. *MAKARA Journal of Technology*. 28(1):5 DOI: 10.7454/mst.v28i1.1620.