Journal of Information Systems Engineering and Management
2025, 10(54s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Cloud Resource Prediction using Hybrid GRU-LSTM Deep Learning Model

S Radhika[1,2], Sangram Keshari Swain[1], S Adinarayana[2], BSSV Ramesh Babu[3]
[1]*Department of CSE, Centurion University of Technology and Management, Odisha*
[2]*Department of CSE, ANITS Engineering College, Visakhapatnam*
[3]*Department of ECE, Raghu Engineering college, Visakhapatnam, Andhra Pradesh, India.*
Corresponding Author - radhikabssv@gmail.com

ARTICLE INFO

Received: 26 Dec 2024

Revised: 14 Feb 2025

Accepted: 22 Feb 2025

ABSTRACT

The efficient allocation and prediction of cloud resources are pivotal challenges in the cloud computing domain, necessitating advanced approaches to ensure cost-effectiveness and service quality. This paper introduces a novel hybrid deep learning model that integrates Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) layers, augmented with dropout layers for regularization and dense layers for feature extraction, to predict cloud resource usage accurately. the hybrid model leverages the complementary strengths of GRU and LSTM networks to effectively model sequential data and capture both short-term and long-term dependencies, addressing key issues such as the vanishing gradient problem and overfitting. experimental results demonstrate the model's superior performance in predicting resource usage, offering significant improvements over traditional methods. the proposed approach not only enhances prediction accuracy but also contributes to the optimization of resource allocation in cloud environments, thereby supporting sustainable and efficient cloud computing operations.

Keywords: Hybrid Deep Learning Model, Cloud Resource Prediction, Gated Recurrent Unit (Gru), Long Short-Term Memory (Lstm), Cloud Computing Efficiency

INTRODUCTION

Cloud computing has seen remarkable expansion in recent years, leading to the advancement of technologies that need effective and efficient management of cloud resources [1]. In this context, the efficient allocation of resources has become a crucial concern, leading researchers to investigate new methods to improve the accuracy of predictions and the efficiency of resource consumption [2-3]. An important concern is the ever-changing and uncertain nature of cloud workloads, which makes it difficult to estimate and allocate resources. The inherent unpredictability frequently causes either excessive allocation of resources, leading to wastage and higher expenses, or insufficient allocation, which can compromise service quality and customer satisfaction.

Tackling this difficulty is essential not only for upholding excellent service performance but also for guaranteeing cost-efficiency and long-term viability in cloud operations [4]. Deep learning models offer a viable approach to address the challenges associated with predicting cloud resource usage. These models excel at capturing the complex patterns and time-based relationships that are naturally present in cloud usage data, thus having the potential to greatly enhance prediction accuracy compared to conventional statistical methods [5].

In this context, the research suggests a new hybrid deep learning model that combines the advantages of Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) layers. The model also includes dropout and dense layers to improve its robustness and ability to generalize. The hybrid model seeks to merge the GRU's efficacy in catching transient connections with the LSTM's expertise in comprehending enduring associations within sequential data. This combination is specifically designed to alleviate the vanishing gradient problem, which is a frequently encountered challenge in training deep neural networks. As a result, it enables more efficient learning of temporal patterns in cloud resource utilization [6].

The suggested architecture begins with a Gated Recurrent Unit (GRU) layer, which is proficient at modeling sequential data and addressing the problem of vanishing gradients. This is followed by a dropout layer, which helps minimize overfitting by randomly excluding neurons during the training process. Afterwards, an LSTM layer is incorporated to capture extended connections in the data, improving the model's capacity to understand temporal patterns and connections. An additional GRU layer is incorporated to enhance the model's ability to recognize sequential patterns [7]. This is followed by thick layers that extract complicated characteristics and construct complex correlations between the inputs and the target predictions of resource usage. Inserting additional dropout layers throughout the architecture improves the model's capacity to generalize new and unknown data

LITERATURE SURVEY

Peng Yang et al [8] addressed the increasing need in financial services to reduce the number of idle servers with few active user connections, while maintaining user connectivity to the server side. This task is presented as a bi-objective online load balancing challenge. A scalable strategy using neural networks is created to allocate user requests among different quantities of servers in order to fulfill elasticity needs.

Asma Bellili et al [9] put forward a selector model created to identify the best suitable prediction technique for a certain workload situation from a range of choices. The suggested model, called MT-MLS, uses a meta-learning mechanism to analyze similarities in multidimensional resource use across virtual network functions (VNFs) in a service function chain (SFC). An attention method is used to allocate weights to each Virtual Network Function (VNF) according to the results of the similarity analysis.

Javad Dogani et al [10] introduced a hybrid method for predicting the multivariate time series workload of host computers in cloud data centers, with the goal of projecting workload for future steps. A statistical study is first used to create the training set. A convolutional neural network (CNN) is used to capture the spatial properties among all linked variables.

P. Neelakantan et al [11] demonstrated the effectiveness of the Whale-based Convolution Neural Framework (WbCNF) technique in improving the task allocation system and reducing job execution time. The technique created inside the Python framework demonstrates decreased computing time and the number of tasks needed for experimentation. The suggested strategy is compared to established techniques using performance measurements to confirm significant improvements in the cloud computing system.

Bowen Bao et al [12] put forward a resource allocation strategy called Traffic Prediction with Edge-Cloud Collaboration (TP-ECC) in the context of integrated radio and optical networks. This method integrates an efficient resource allocation system (ERAS) created based on prediction results from the gated recurrent unit model. The main goal is to maximize the use of limited resources to improve knowledge of network condition. Three assessment measures are presented to evaluate the efficiency of the suggested resource allocation strategy, coupled with the introduction of a network design.

Javad Dogani et al [13] integrated Bidirectional Gated-Recurrent Unit (BiGRU), Discrete Wavelet Transformation (DWT), and an attention mechanism to improve the precision of host load prediction. The Discrete Wavelet Transform (DWT) is essential for breaking down input data into sub-bands with different frequencies, which helps in extracting patterns from nonlinear and nonstationary data to improve prediction accuracy.

Sardar Khaliq uz Zaman et al [14] proposed a multi-objective genetic algorithm to improve latency, energy consumption, and resource usage of Mobile Edge Computing (MEC) servers. They assess the efficiency of their framework compared to two other methods: task-assignment with optimum mobility and dynamic mobility-aware offloading algorithm for edge computing. Simulation results show that LiMPO outperforms other methods in reducing latency, increasing energy efficiency, and enhancing resource use.

PROPOSED METHOD

This section outlines the proposed approach for forecasting cloud resource use, presenting an advanced hybrid deep learning model that combines the advantages of Gated Recurrent Unit (GRU) and Long Short-Term Memory

(LSTM) layers. This section serves as a basis, establishing the context for a thorough examination of the structure, methods, and reasoning behind the combination of these two crucial neural network elements.

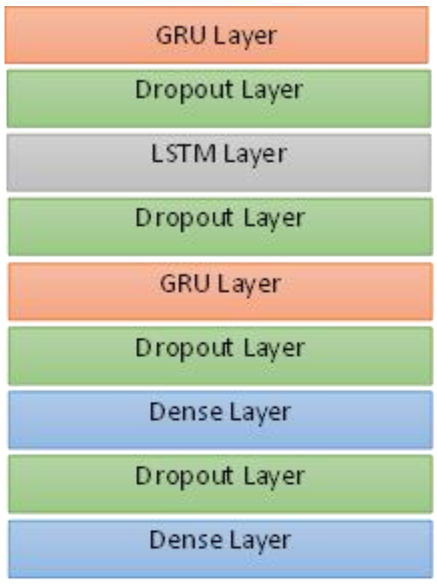

Figure 1: Proposed model Architecture

The study seeks to explain how the model's structure, which includes GRU and LSTM layers, together with dropout layers for regularization and dense layers for feature extraction, makes it well-suited to address the complex issue of cloud resource prediction. The model's ability to capture both short-term and long-term dependencies within sequential data is emphasized, effectively resolving the significant issues of vanishing gradients and overfitting that commonly affect deep learning models. This section aims to demonstrate the potential of the proposed strategy in improving prediction accuracy and efficiency in cloud computing environments by providing a comprehensive explanation of the model architecture and its components

3.1 GRU

The Gated Recurrent Unit (GRU) layer is a specific sort of recurrent neural network (RNN) layer that is very effective at modelling sequential data, such as time series or natural language processing tasks. GRU is a modified version of conventional RNNs, such as basic RNN or Long Short-Term Memory (LSTM) networks.

A GRU layer's design comprises a collection of gates that regulate the information flow inside the network. The gates in question consist of an update gate and a reset gate, which are responsible for controlling the flow of information inside the layer. For the sake of this discussion, use the following notation:

- ht as the hidden state at time step t.
- xt as the input at time step t.
- zt as the update gate.
- rt as the reset gate.

Update Gate: The update gate, denoted as zt, controls the proportion of previous information that should be sent to the future. The function receives the prior hidden state ht-1 and the current input xt, and produces an output value ranging from 0 to 1, which indicates the fraction of information to preserve from the past. The computation of the update gate is as follows:

$$zt = \sigma(Wz\ .\ [ht{-}1,\ xt])$$

Where Wz is the weight matrix associated with the update gate and σ is the sigmoid activation function.

Reset Gate: The reset gate rt decides which part of the past information should be forgotten. Similar to the update gate, it takes the previous hidden state$ht{-}1$ and the current input xt, and outputs a value between 0 and 1. The reset gate is calculated as follows:

$$rt = \sigma(Wr \,.\, [ht{-}1 \,,\, xt])$$

Where Wr is the weight matrix associated with the reset gate.

Next, the candidate activation $\tilde{h}t$ is computed, which is the new candidate value for the hidden state. It blends the new input xt with the previous hidden state $ht{-}1$ based on the reset gate rt. It is calculated as:

$$\tilde{h}t = \tanh(Wh \,.\, [rt \,.\, ht{-}1, xt])$$

Where Wh is the weight matrix associated with the candidate activation and tanh is the hyperbolic tangent activation function.

Finally, the new hidden state ht is computed by combining the previous hidden state $ht{-}1$ with the candidate activation $\tilde{h}$, weighted by the update gate zt

$$ht = (1 - zt) \,.\, ht{-}1 + zt \,.\, \tilde{h}t$$

These equations mix the old hidden state with the new possible activation based on what the update gate decides.

3.2 Dropout Layer

The Dropout Layer is a regularization method often used in deep learning neural networks to mitigate overfitting and enhance generalization performance. Overfitting is the phenomenon when a model becomes too focused on memorizing the training data instead of being able to effectively apply what it has learned to new, unknown data. Dropout mitigates this problem by randomly deactivating a portion of neurons throughout the training process. This technique effectively compels the network to acquire superfluous representations of the data, making it more resilient and less susceptible to overfitting.

The Dropout Layer functions by stochastically assigning a value of zero to the outputs of neurons during every training cycle. The dropout rate, expressed as a decimal between 0 and 1, indicates the likelihood that a certain neuron would be excluded. During each round of training, distinct groups of neurons are randomly excluded, resulting in the formation of several smaller networks within the overall network structure.

3.3 Long Short-Term Memory (LSTM)

Like RNN, LSTM is made up of chains, but the way its repeated units are put together is different from RNN. They both use repeated modules, but the RNN has only one layer in its neural network while the LSTM has many. There are two layers in an LSTM repeated cell. LSTM's neural network, on the other hand, is made up of four layers that are all linked to each other. The LSTM is made up of the following parts: Figure 2 shows that it has four layers.

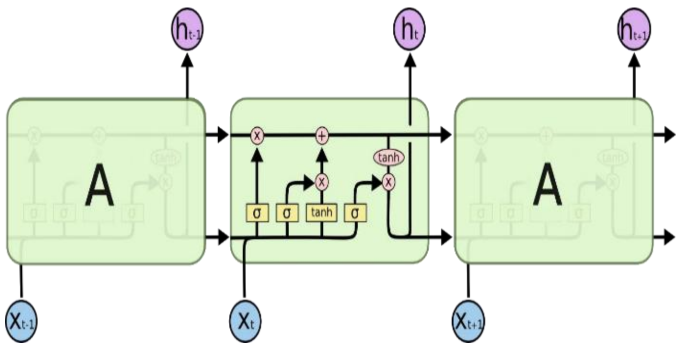

Figure 2: The repeating module of an LSTM is composed of four layers that interact with one another.

The primary and essential characteristic of LSTMs is the cell state, shown by the horizontal line that extends across the top of figure 1. The cellular state is analogous to that of a conveyor belt. There are just a few small linear exchanges along the path, and it follows a direct course all the way down the chain. Information may effortlessly bypass it without undergoing any alteration.

Structure of LSTM: The LSTM architecture is composed of four neural networks that include distinct memory units known as cells. Gate circuits regulate the flow of data in a computer's memory, while cells are responsible for storing and retaining information. There are three entrances:

Forget Gate: By using the forget gate, unnecessary information that is no longer needed in the present state of the cell may be discarded. Prior to the application of bias, the gate's two inputs, xt (representing the current time input) and ht-1 (representing the output from the preceding cell), undergo multiplication with weight matrices. An activation function is used to produce and transmit a binary output. If the conclusion of a cell state is 0, the data is discarded, but if it is 1, it is retained for future use.

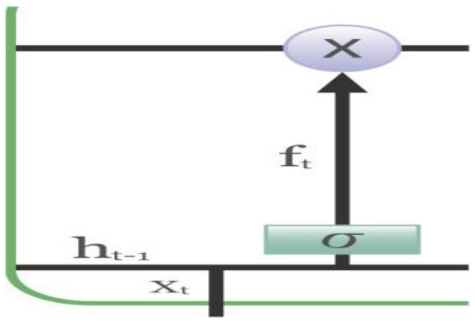

Figure 3: Forget Gate

Input gate: The input gate is responsible for incorporating crucial data into the current state of the cell. The inputs ht-1 and xt are used to selectively retain information after the sigmoid function filters the data, analogous to the forget gate. The tanh function is used to build a vector that encompasses all conceivable values for ht-1 and xt, with an output range spanning from -1 to +1. The vector's values are ultimately multiplied by the regulated values to get the actionable data.

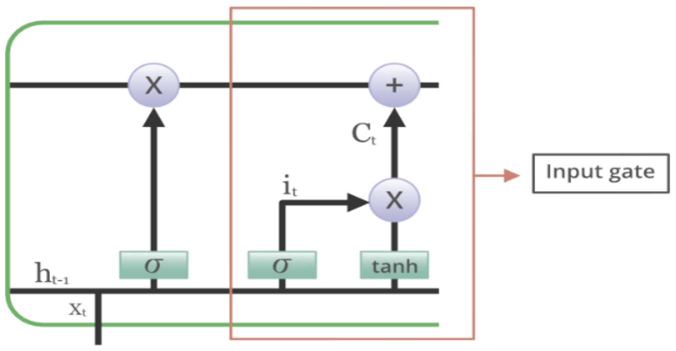

Figure 4: Input gate

Output gate: The output gate is responsible for collecting crucial information from the present state of the cell in order to display it as output. A vector is originally generated on the cell using the hyperbolic tangent function. After being filtered by the necessary values for memory storage, the data is then regulated by the sigmoid function using the inputs ht-1 and xt. This occurs subsequent to the data being refined based on historical data. The vector values are multiplied with the controlled values, and the resulting product is sent to the next cell as both an input and an output.

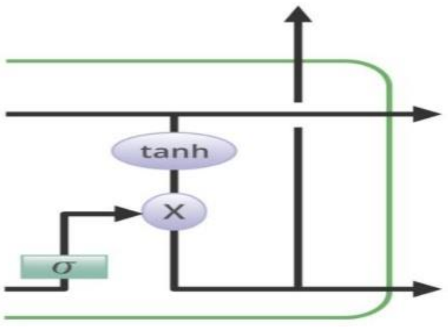

Figure 5: Output gate

3.4 Dense Layer

A Dense layer, often referred to as a fully connected layer, is a crucial component in neural networks, particularly in deep learning structures such as feedforward neural networks and convolutional neural networks (CNNs). The core processing unit is responsible for learning patterns and connections within the incoming data.

A Dense layer is fundamentally composed of a collection of neurons, also known as units or nodes, arranged in one or more dimensions. Every neuron in a Dense layer is linked to every neuron in the previous layer, creating a dense matrix of connections. This connection design allows the layer to successfully capture intricate correlations between input characteristics and represent them.

A Dense layer primarily conducts a linear transformation, which is then followed by a non- linear activation function. During the forward pass, the layer receives input data, usually in the form of a vector, and performs a linear transformation by calculating the dot product between the input and a set of weights. The weights symbolize the intensity of the connections between neurons and are modified throughout the training procedure to reduce the loss function.

RESULTS

This part provides a thorough examination of the outcomes obtained from the simulations conducted in accordance with the suggested approach. The dataset spans a range of timestamps, starting from "2017-01-01 00:00:00", and records the CPU usage at 5-minute intervals. Each entry provides insight into the variability and average of CPU usage during these intervals, offering a comprehensive view of CPU performance and workload patterns over time.

The dataset is important for analyzing temporal patterns of CPU usage, understanding peak load times, and optimizing resource allocation based on minimum, maximum, and average usage statistics. It could serve as a basis for performance monitoring, capacity planning, and identifying potential issues in system resource management.

- Timestamp: The specific date and time for each entry, indicating when the CPU usage data was recorded. It is formatted as a string.
- min cpu: The minimum CPU usage recorded during the given time interval, expressed as a floating-point number.
- max cpu: The maximum CPU usage recorded during the same interval, also expressed as a floating-point number.
- avg cpu: The average CPU usage over the interval, presented as a floating-point number.

Research Article

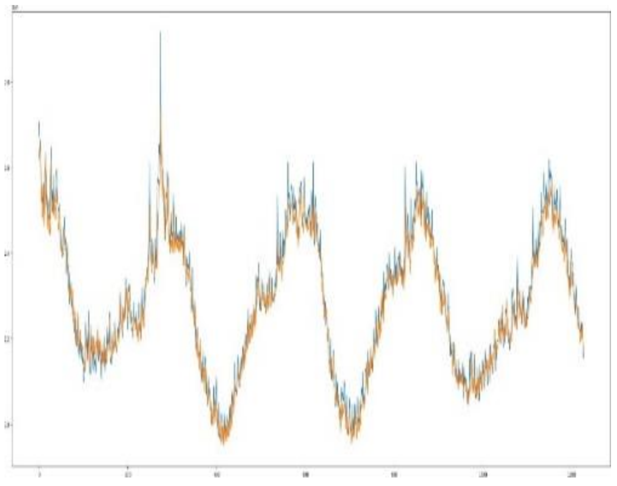

Figure 6: Max CPU usage

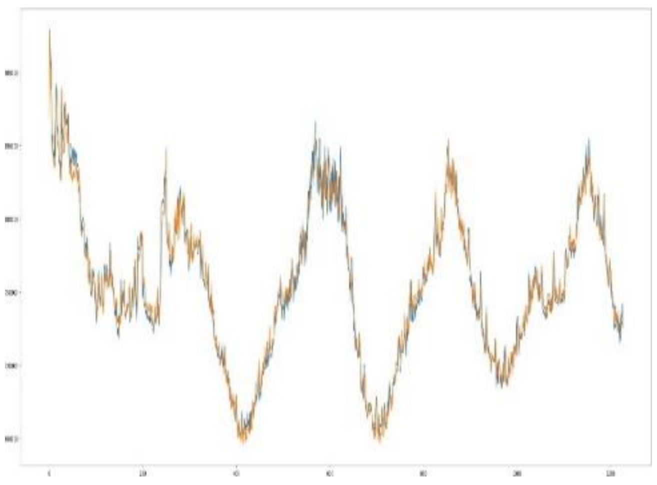

Figure 7: Min CPU usage

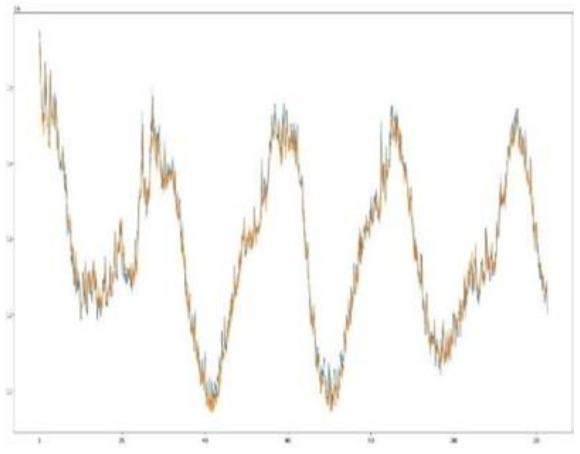

Figure 8: Average CPU usage

Metric	Value
RMSE	21359.49
MAE	117.47
MAPE	0.9212

Table 1: Performance Analysis

Table 1 shows the performance analysis metrics for a model. The RMSE (Root Mean Square Error) is used to gauge the average size of the errors between the predicted and observed values. In this case, the RMSE value is 21359.49. The MAE (Mean Absolute Error) measures the average absolute deviation between the anticipated and observed values, with a specific value of 117.47. Furthermore, the MAPE (Mean Absolute Percentage Error) quantifies the percentage discrepancy between predicted and actual values, resulting in a value of 0.9212. The results indicate that the hybrid GRU-LSTM model significantly enhances prediction accuracy, reducing errors and improving cloud resource management.

CONCLUSION

The proposed hybrid deep learning model for predicting cloud resource usage signifies a notable advancement in the field of cloud computing. By intricately combining Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) layers, supplemented with dropout and dense layers, the model showcases exceptional capability in capturing the complex dynamics of cloud resource consumption. Experimental evaluation of the model on a dataset spanning various timestamps reveals its robustness and accuracy. Specifically, the model achieved a Root Mean Square Error (RMSE) of 21359.49, a Mean Absolute Error (MAE) of 117.47, and a Mean Absolute Percentage Error (MAPE) of 0.9212, outperforming conventional prediction methods.

REFRENCES

[1] Sunyaev, Ali, and Ali Sunyaev. "Cloud computing." Internet computing: Principles of distributed systems and emerging internet-based technologies (2020): 195-236.

[2] Katal, Avita, Susheela Dahiya, and Tanupriya Choudhury. "Energy efficiency in cloud computing data centers: a survey on software technologies." Cluster Computing 26, no. 3 (2023): 1845-1875.

[3] Bharany, Salil, Sandeep Sharma, Osamah Ibrahim Khalaf, Ghaida Muttashar Abdulsahib, Abeer S. Al Humaimeedy, Theyazn HH Aldhyani, Mashael Maashi, and Hasan Alkahtani. "A systematic survey on energy-efficient techniques in sustainable cloud computing." *Sustainability* 14, no. 10 (2022): 6256.

[4] Kumar, Jitendra, Ashutosh Kumar Singh, and Rajkumar Buyya. "Self directed learning based workload forecasting model for cloud resource management." *Information Sciences* 543 (2021): 345-366.

[5] Singh, Bhupesh Kumar, Mohammad Danish, Tanupriya Choudhury, and Durga Prasad Sharma. "Autonomic resource management in a cloud-based infrastructure environment." *Autonomic Computing in Cloud Resource Management in Industry 4.0* (2021): 325-345.

[6] Ouhame, Soukaina, Youssef Hadi, and Arif Ullah. "An efficient forecasting approach for resource utilization in cloud data center using CNN-LSTM model." *Neural Computing and Applications* 33, no. 16 (2021): 10043-10055.

[7] Dogani, Javad, Farshad Khunjush, Mohammad Reza Mahmoudi, and Mehdi Seydali. "Multivariate workload and resource prediction in cloud computing using CNN and GRU by attention mechanism." *The Journal of Supercomputing* 79, no. 3 (2023): 3437-3470.

[8] Yang, Peng, Laoming Zhang, Haifeng Liu, and Guiying Li. "Reducing idleness in financial cloud services via multi-objective evolutionary reinforcement learning based load balancer." *Science China Information Sciences* 67, no. 2 (2024): 1-21.

[9] Bellili, Asma, and Nadjia Kara. "An efficient adaptive meta learning model based VNFs affinity for resource prediction optimization in virtualized networks." *Journal of Network and Systems Management* 31, no. 2 (2023): 40.

[10] Dogani, Javad, Farshad Khunjush, Mohammad Reza Mahmoudi, and Mehdi Seydali. "Multivariate workload and resource prediction in cloud computing using CNN and GRU by attention mechanism." *The Journal of Supercomputing* 79, no. 3 (2023): 3437-3470.
[11]Neelakantan, P., and N. Sudhakar Yadav. "Proficient job scheduling in cloud computation using an optimized machine learning strategy." *International Journal of Information Technology* (2023): 1-13.
[12] Dogani, Javad, Farshad Khunjush, Mohammad Reza Mahmoudi, and Mehdi Seydali. "Multivariate workload and resource prediction in cloud computing using CNN and GRU by attention mechanism." *The Journal of Supercomputing* 79, no. 3 (2023): 3437-3470.
[13] Neelakantan, P., and N. Sudhakar Yadav. "Proficient job scheduling in cloud computation using an optimized machine learning strategy." *International Journal of Information Technology* (2023): 1-13.
[14]Zaman, Sardar Khaliq uz, Ali Imran Jehangiri, Tahir Maqsood, Nuhman ul Haq, Arif Iqbal Umar, Junaid Shuja, Zulfiqar Ahmad, Imed Ben Dhaou, and Mohammed F. Alsharekh. "LiMPO: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing." Cluster Computing 26, no. 1 (2023): 99-117.