Revolutionizing Cloud Resource Allocation: Harnessing Layer-Optimized Long Short-Term Memory for Energy-Efficient Predictive Resource Management

Prathigadapa Sireesha[1*],Vishnu Priyan S[2], M. Govindarajan[3], Sounder Rajan[4] and V. Rajakumareswaran[5]

[1]Asia Pacific University of Technology & Innovation, Malaysia
[2]Biomedical Engineering, Kings Engineering College, Irungattukottai, Sriperumbudur, Chennai, India
[3]Department of General Engineering, Velalar College of Engineering and Technology, Erode, Tamilnadu, India
[4]Department of Artificial Intelligence and Data Science, Erode Sengunthar Engineering College, Thuduppathi, Tamil Nadu, India
[5]Dept of Computer Science and Design, Erode Sengunthar Engineering College, Thuduppathi, Tamil Nadu, India

Abstract

INTRODUCTION: This is the introductory text. Accurate data center resource projection will be challenging due to the dynamic and constantly changing workloads of multi-tenant co-hosted applications. Resource Management in the Cloud (RMC) becomes a significant research component. In the cloud's easy service option, users can choose to pay a fixed sum or based on the amount of time.
OBJECTIVES: The main goal of this study is systematic method for estimating future cloud resource requirements based on historical consumption. Resource distribution to users, who require a variety of resources, is one of cloud computing main objective in this study.
METHODS: This article suggests a Layer optimized based Long Short-Term Memory (LOLSTM) to estimate the resource requirements for upcoming time slots. This model also detects SLA violations when the QoS value exceeds the dynamic threshold value, and it then proposes the proper countermeasures based on the risk involved with the violation.
RESULTS: Results indicate that in terms of training and validation the accuracy is 97.6%, 95.9% respectively, RMSE and MAD shows error rate 0.127 and 0.107, The proposed method has a minimal training and validation loss at epoch 100 are 0.6092 and 0.5828, respectively. So, the suggested technique performed better than the current techniques.
CONCLUSION: In this work, the resource requirements for future time slots are predicted using LOLSTM technique. It regularizes the weights of the network and avoids overfitting. In addition, the proposed work also takes necessary actions if the SLA violation is recognized by the model. Overall, the proposed work in this study shows better performance compared to the existing methods.

Keywords: Long-short-term memory, cloud computing, energy-efficient resources, and resource forecasting

Received on 26 December 2023, accepted on 27 June 2024, published on 03 July 2024

doi: 10.4108/ew.6505

1. Introduction

Cloud computing has becoming more well-liked as a research area because of its great scalability, adaptability, and cost-efficiency [1]. The provision of several services, including web, multimedia, gaming, and IoT applications, has made extensive use of distributed cloud computing systems, such as fog computing, big cloud data centers, and edge computing due to their added advantages in integrating hardware, resource configuration, and service separation.

[*]Corresponding author. Email: sireesha.prathi@apu.edu.my

One of the most significant issues with cloud and edge computing systems is energy efficiency [2]. In the cloud's easy service option, users can choose to pay a fixed sum or based on the amount of time. Power consumption is a major factor in how much it costs to run cloud computing systems. Energy costs for data will rise if the current pattern holds.

In this field, Resource Management in the Cloud (RMC) becomes a significant research component. In earlier methods, the resource distribution evenness mechanism was utilized to distribute the available resources among the active applications. This strategy may result in resource overflow owing to high resource allocation levels above what is necessary and resource underflow due to low resource allocation levels above what is necessary.

Depending on the number of active clients, a running application's resource requirements will occasionally alter. Recent research has developed dynamic resource management with virtual systems to address resource underflow and overflow issues caused by evenness distribution. When allocating resources to apps, these systems take into account the server's available resources and the demands of the various programs.

Algorithms for unevenness and on-demand resource allocation are used by dynamic management systems to accomplish this. With the virtualization of cloud systems, this strategy will manage the resources in a dynamic way that is effective. It will also be possible to prevent SLA violations in a cloud environment by dynamically matching virtual requirements with physical resources.

It's possible for allocation attempts to fail or for a system to hang on occasion due to unexpectedly high future resource demands.

Resource distribution to users, who require a variety of resources, is one of cloud computing's main goals. For the model to function as a high-performance computing system, an efficient allocation technique is needed. Data centers that host cloud platforms are now being used by more people, thus it's imperative to create an allocation technique that offers higher-quality services to balance the rising and constantly shifting burden.

Due to the diversity of resources, the variety of workloads, the heterogeneity of resource capabilities, and the restrictions on there are problems with allocating resources in the cloud, such as where to put servers that are used for workload [4]. For running cloud computing systems, the most important factor is the energy efficiency. This is because energy efficiency can reduce the cost of carbon emissions.

Acceptable workloads and the server's processing power have been shown to have a greater impact on cloud server's energy expense. Therefore, using an effective prediction model, it is crucial to precisely estimate the workload and resource requirements for the future. Due to the time-varying and dynamic workloads of the multi-tenant co-hosted applications, predicting the resources needed for future will be difficult for data centers [5].

To overcome these issues, this research suggests a novel, systematic method for estimating future cloud resource requirements based on historical consumption. A significant number of successful cases in the fields of speech recognition, object detection, and other fields have shown deep learning to be a cutting-edge technique for big data analysis.

Although there are a number of deep learning network-based estimation strategies for making advantage of cloud resources [4], [6], [7], and [8]. When very large time lags of undetermined duration exist between critical events, an LSTM network is well suited for learning from experience to categorize, analyze, and predict time series data [7]. The term LSTM refers to specific RNN for deep learning. Three or four "gates" are present in each LSTM block, and these gates are an exclusive RNN for employed to regulate how information enters and exits the memory of the block.

Utilizing a logistic function to generate a number between 0 and 1, these gates are created. To partially permit or refuse information to enter or leave the memory, multiplication is applied using this value. The required memory for estimating the output activation of the blocks, output gate has been used whereas input gate maintains value in the memory. As mentioned in [9], the amount of value flowing into and outside the memory has been controlled by the forget gate.

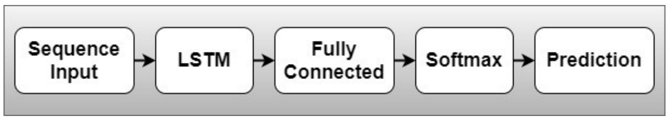

Figure 1. Illustrates a Simple LSTM Network Architecture for prediction

The input layer and LSTM layers are the fundamental layers of LSTM network and a sequence input layer is utilized to deliver the sequence data and the input layer into the network. The long-term relationships between the sequence data's time steps has been studied by the LSTM layer. The network's structure is shown in Figure 1 as a straightforward LSTM network for categorization. The first two layers of the network are input layer and an LSTM layers.

Class labels are predicted using the networks fully connected final layers of SoftMax. The factors listed below influenced the choice of LSTM. As suggested by its name, LSTM may at first pick up long-term dependence. Through learning, the neural network may also choose when to remove old material and for how long to keep it in memory. Useful data can be extracted by employing the end-to-end model, LSTM, as opposed to an autoencoder as a pre-recurrent feature layer. Finally, because of its excellent nonlinear generalization capacity, it can successfully represent highly noisy one-dimensional time series.

Existing deep learning-based approaches [9][2][13] use LSTM for prediction. Even though these techniques' results are best, however they have issues with poor accuracy and temporal complexity. As a result, choosing the appropriate network structure and precisely adjusting the hyper-parameters of the network are both necessary before training the model. As a result, our paper suggests a novel LSTM approach to address these issues.

Following is a list of this work's main contributions.

- To propose Layer Optimized Based on Long Short-Term Memory (LOLSTM), a more effective technique for allocating resources in the cloud.
- To control the level of risk brought on by SLA violations by keeping an eye on the QoS parameter.
- Tuning hyperparameters and hidden layers are done in order to train the suggested LOLSTM model. Resolving the vanishing gradient issue with the help of the rectified linear activation function (ReLU) improves the model's accuracy.
- In terms of validation accuracy, validation loss, training accuracy, and error rate, the proposed LOLSTM model outperforms current approaches.
- This paper structure is set up as follows: Section 2 examines relevant research in the areas of cloud resource allocation that is energy efficient. Section 3 describes the energy efficient resource allocation through the proposed LOLSTM based prediction of future resource requirement in cloud. Results and discussion in Section 4 demonstrate how effectively the suggested task was carried out. Section 5 discusses the study's conclusion.

2. Literature Review

Tseng et al. [1] proposed a multi-objective genetic algorithm (GA) to dynamically estimate resource use and energy consumption in cloud data centers. They created a multi-objective resource allocation optimization problem that takes into account how much CPU and RAM PMs and VMs consume as well as how much energy the data center uses. They failed to compare their work to other metaheuristics systems and failed to demonstrate their prediction accuracy.

Nguyen et al. [2] presented a prediction method based on Long Short-Term Memory Encoder-Decoder (LSTM-ED) for predicting the real load multi-step ahead and mean load over successive intervals. Because the time series data is internally represented, the LSTM-ED based technology expands the LSTM's memory capacity. It is essential to improve load prediction accuracy.
Song et al. improve the resource-utilization-aware energy efficient server consolidation method (RUAEE) by incorporating a resource reserve. The VM allocations can be changed by continuously calculating the ratio between resource allocation and reservation. Their migration and communication are expensive.
A deep reinforcement learning model has been demonstrated by Karthiban and Raj [4] for supporting users with resource allocation in green computing. The dimensionality issue renders conventional Q-learning models ineffective due to the exponential growth of the state space and they did not predict future demand of resources.
A new method was presented by Baig et al. [5] for automatically and adaptively selecting the optimum model to forecast the utilization of data center resources choose the optimal prediction model to apply to specific resource utilization observations acquired over a period of time, the suggested method creates a classifier based on statistical characteristics of previous resource usage. They did not enhance any machine learning techniques and not used any deep learning technique for workload prediction.
Chien et al. [6] integration of cloud computing with edge computing in an evolving network design reduces the transmission of unnecessary data and addresses bottleneck difficulties. For dynamic throughput prediction, they employed LSTM, and for resource allocation optimization, they used GARAA, a GA-based algorithm. They weren't very concerned with increasing prediction precision and speeding up deep learning.
Yu et al. [7] suggested an architecture for resource allocation based on self-organizing network-based matching slices. The dynamic traffic model of the multicast service in space-time is built using the multidimensional data and the efficient deep learning model known as LSTM (long short-term memory), which forms the basis for extra network resource allocation. They didn't enhance the LSTM.
Autoencoders were utilized by Zhang et al. [8] to forecast how much CPU time virtual machines will need. The authors' tensor rank decomposition method reduced training time by compressing the input parameters. It needs a lot of processing power. Deep learning-based service composition (DLSC) was suggested by Haytamy and Omara [9]. This system combines the long short-term memory (LSTM) network of deep learning and particle swarm optimization (PSO).
The PSO algorithm uses the results from the LSTM network to choose the best cloud service providers to work with in order to put together the necessary services and lower the consumer cost function. Using the LSTM network, precise predictions of the Cloud QoS values are made.
For a dynamic resource allocation process, Praveenchandar et al. [10] recommended improved work scheduling and the optimum power minimization strategy. Resource allocation efficiency can be obtained in terms of task completion and reaction time by utilizing prediction mechanisms and dynamic resource table updating algorithms. For workload prediction, they didn't apply any deep learning techniques.
Hussain et al. [11] looked at the procedure for finding and fixing SLA violations from a risk management perspective. As soon as an SLA was established, they provided a Risk Management-based Framework for SLA Violation Abatement (RMF-SLA), which entails SLA monitoring, violation prediction, and decision counseling. They used experiments to confirm and show that the framework is suitable for assisting cloud providers in avoiding service violations and fines.
Banerjee et al. [12] suggested a multi-step-ahead workload prediction method using machine learning methods. Based on this forecast, it is recommended to distribute resources so they can be used more efficiently, reducing the data center's overall energy consumption. Based on the actual Bit brains workload trace, they assessed the efficiency of our framework.

3. Methodology

The integration of numerous resources into a data center enables it to offer a range of resource services for cloud computing. How to deliver resources in a timely and accurate manner to meet user expectations is a significant concern. However, the resource demands of users fluctuate greatly and frequently change regularly. It's possible that the resource provision won't happen on time.

Additionally, since some physical resources are sometimes turned off to save energy, they may not always be enough to meet user requests. To ensure positive user experiences using cloud computing, it is crucial to offer proactive resource provision. Recent cloud systems are experiencing resource management issues as a result of unanticipatedly high asynchronous resource demands (CPU time, memory, networks, etc.).

Accurately predicting future resource demands is essential for supporting resource provision in advance. As a result, this work elaborately introduces the innovative technology known as Layer optimized based Long Short-Term Memory (LOLSTM). Here, the created LOLSTM approach makes predictions about future resource needs based on data from the past. For efficient resource allocation in the cloud, the CPU utilization, RAM, and workload are all included in the objective model.

Additionally, when training the model, both internal and external factors such as different types and quantities of VMs with different leasing costs are taken into account. External aspects include dynamic workload. Additionally, the LSTM model's training process optimizes the hyperparameters and hidden layer count. The proposed LOLSTM model's design is shown in Figure 2. Four layers of the proposed LOLSTM technique are as follows:

- The Input layer,
- The Multiple hidden layer,
- The Fully Connected layer, and
- The Output layer.

(i) The Input layer

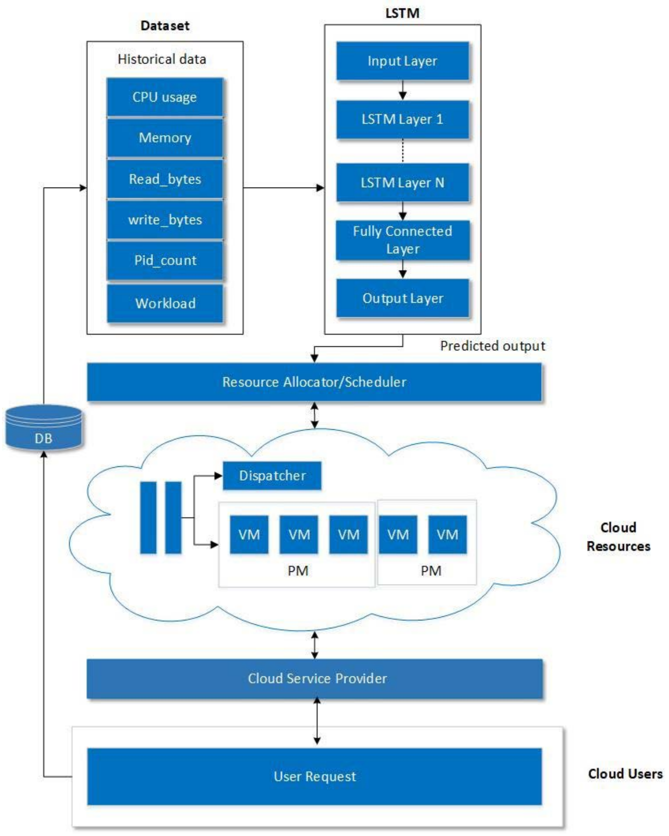

Figure 2. Illustrates the architecture of proposed LOLSTM technique

This is the first layer of LSTM model. The inputs for this layer are CPU usage time, memory, read_bytes, write_bytes, pid count and previous workload at different time interval which are formed using historical information. Let us assume the cloud structure with k Physical Machine's (PM) and b Virtual Machine's (VM) and they are denoted as $PM = \{PM_1, PM_2, .. PM_s \ldots.. PM_k\}; 1 \leq s \leq k$ and $VM = \{VM_1, VM_2, .. VM_a, \ldots.. VM_b\};\ 1 \leq a \leq b$ respectively. All PM will include dissimilar VMs.

Data resource requested by the user can be denoted as,

$$R = \left\{R_1, R_2, R_3, \ldots.. R_f\right\} \tag{1}$$

Data rate for each user can be represented as,

$$D = \left\{D_1, D_2, D_3, \ldots.. D_f\right\} \tag{2}$$

The history of the users (previously used resources) can be given as,

$$H = \{h_1, h_2, h_3, \ldots \ldots h_x\} \tag{3}$$

(ii) Multiple hidden layers

This layer has several LSTM layers, each of which has several LSTM blocks. In LSTM layers, the number of hidden layers and hyperparameters are controlled. Additionally, QoS monitoring will be done, and users will be assigned resources based on the outcomes of the predictions. The building blocks of a typical LSTM network are called cells. Both the concealed state and the cell state are carried over to the subsequent cell. The memory blocks, which are managed by three crucial parts known as gates, are in charge of recalling information. Three distinct input, output, and forget gates are shown on a single LSTM block in Figure 3. The LSTM uses three different gate mechanisms to monitor and safeguard data [14].

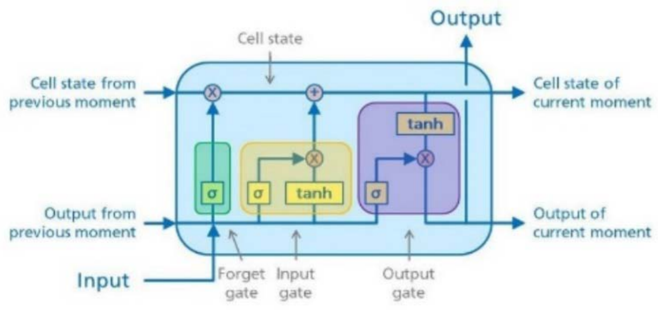

Figure 3. Shows the composition of one LSTM block

LSTM can be expressed mathematically as follows. Input gate outputs two values after receiving input $x_{t,}$and outputh_{t-1} from the previous time step.
Input gate:
$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$ (4)

$\tilde{C}_t = \tanh\,(W_C[h_{t-1}, x_t] + b_C$ (5)

Forget gate:
$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$ (6)

Cell state:
$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$ (7)
Output gate:
$o_t = \sigma(W_o[h_{t-1}, x_t + b_o)$ (8)
The LSTM unit's final output is:
$h_t = o_t * \tanh\,(C_t)$ (9)

The weight matrix 'W' and bias vector 'b' in the equation above represent the learnable parameters in each gate. The sigmoid function is (x), and * stands for dot formation. After each LSTM block, the Rectified Linear Unit (ReLU) has been employed because it is easier to train and performs better. In the neural network, activation functions play a big role, where output at a network node corresponds to input. It improves learning rate, enables the neural network to understand nonlinear dependencies, and helps it stay away from vanishing gradients.

In addition, there is no transform performed. For network predictions, At the output layer, linear activation functions are utilized. Every LSTM block uses ReLU because of its simple computations, linear nature, and simplicity of training. The ReLU function using x as the input vector looks like this:

$ReLU\,(x) = \max\,(0, x)$ (10)

Tuning the Hidden layers and Hyperparameters

Hidden layers are acknowledged as being a crucial component of neural network activity, particularly when addressing difficult conditions. This is only possible, though, if the network has a complete understanding of the issue without any restrictions due to under or over-fitting. Generally speaking, neural networks with several hidden layers will perform better on challenging and larger issues. Neuronal networks need more neurons and hidden layers in order to provide outcomes that are acceptable.

Some of the LSTM hyperparameters are the batch size, the number of units in a dense layer, the number of nodes and hidden layers, momentum, the number of epochs, the activation function, dropout, decay rate, and learning rate. In an effort to provide the most accurate predictions, the learning rate and epochs will be modified. A hyperparameter that controls how many iterations the learning algorithm performs on the entire training set is the number of epochs.

To prevent overfitting and boost the neural network's capacity for generalization, the suggested LOLSTM was trained for the advised number of epochs. For the proposed study, the model is validated and its output is checked using a fraction of the training data utilized in each training session. The epoch at which the model begins to overfit was determined by tracking the accuracy and loss on the training and test datasets.

The most important hyperparameter for maximizing deep neural network training is learning rate. Training is impacted by the learning rate's value. Neither convergent training nor divergent training will take place when the learning rate is high. Additionally, excessively poor learning rates are a result of insufficient instruction. As a result, selecting the appropriate learning will be critical for successful prediction. After selecting optimal hyperparameters and hidden layers, the required resources are predicted and allocated to the cloud users.

To distribute the resource in cloud, the VM is selected based on bandwidth, processor, memory and Million Instructions Per Second (MIPS) and it is given as,

$P_{s,a} = B_{s,a}, M_{s,a}, L_{s,a}, H_{s,a}, T_{s,a}$ (11)

Where $B_{s,a}$ denotes the required number of processor by a^{th} VM contained in s^{th} PM, $M_{s,a},$ denotes the required memory by a^{th} VM contained in s^{th} PM, , $L_{s,a}$ is the MIPS of a^{th} VM contained in s^{th} PM, , $H_{s,a}$ denotes the required bandwidth by a^{th} VM contained in s^{th} PM and $T_{s,a}$ denotes the frequency scaling of a^{th} VM contained in s^{th} PM.

The load of VM can be expressed as,

$L_{s,a} = \sum_{d=1}^{n} \frac{(V_b+V_m+V_l+V_h+V_t)*X}{max(V_b+V_m+V_l+V_h+V_t)*n} * \frac{1}{N}$ (12)

Where V_b, V_m,V_l,V_h, V_h and V_t represents a processing element, memory units, bandwidth component, MIPS element and frequency component ina^{th} VM respectively and n denotes total number of users.

$$X = \begin{cases} 1, & if d^{th} workload run by a^{th} \text{ VM} \\ 0, & Otherwise \end{cases} \quad (13)$$

The load of PM can be denoted as,

$$L_s = \sum_{a=1}^{f} L_{s,a} \quad (14)$$

(iii) Fully Connected layer

This layer is in charge of determining when a SLA has been violated by comparing the expected QoS value to the threshold value. Using a dynamic threshold technique based on median absolute deviation (MAD), SLA violations are managed. The standard deviation is less prone to outliers and is more stable than the MAD [15]. Based on historical server use data, MAD generates new threshold values. Furthermore, MAD values for a given dataset $x_1, x_2, \ldots . x_n$ can be created using the following approach.

$$MAD = median(|x_a - median(X_b)|) \quad (15)$$

here $X_b$stands for prior CPU consumption and x_a stands for current CPU utilization. The threshold value is represented as

$$TH = 1 - Z \times MAD \quad (16)$$

In the equation above, the safety parameter Z is utilized to regulate how the threshold mechanism behaves. SLA violation can be identified based on above defined threshold TH.An SLA breach occurs if the CPU utilization approaches or exceeds the TH, and the next layer will estimate the severity of the risk violation. If the CPU utilization is within the defined TH, no action will be taken.

User level experience can be denoted as,

$$E = T_1 D - T_2 D \quad (17)$$

Where T_1 and T_2 are the constants.

Delay D can be expressed as,

$$D = \sum_{a=1}^{b} \frac{t_a}{k_w} + H_a \quad (18)$$

Where k_w denotes writing speed in hard disk, t_a denotes file size and H_a denotes initialization period of VM.

The factors that make up each job are memory size m_size_i input data size (I_i), CPU utilization (C_i), use of memory for security (m_sec_i), use of memory for scheduling (m_sche_i), using the CPU for security (C_sec_i) using the CPUfor scheduling (C_sche_i).

C_{td} , C_m, $C_B$stands for the transferring costs for task i CPU, memory, and input data size, respectively. The weighted costs for each task, w_{td}, w_m, w_b result in various minimization cost factors.

The cost minimization factor [16] can be expressed as,

$$\min [(C_{td} \times w_{td}) + (C_m \times w_m) + (C_b \times w_b)] (19)$$

$$C_{td} = \sum_{i=1}^{n} I_i \quad (20)$$

$$C_m = \sum_{i=1}^{n} m\_size_i + \sum_{i=1}^{n} m\_sec_i + \sum_{i=1}^{n} m\_sche_i \quad (21)$$

$$C_B = \sum_{i=1}^{n} C_i + \sum_{i=1}^{n} C\_sec_i + \sum_{i=1}^{n} C\_sche_i \quad (21)$$

A single task requires N time slots to be completed as follows:

$$N_x = \frac{C_x}{\tau f_x} \quad (23)$$

Where C_x denotes total number of CPU cycles; $f_x$denotes the operating frequency and τ denotes the allocated time slot for user x.

(iv) Output layer

The LOLSTM model's initial evaluation of the severity of the risk of SLA violation is done in the final layer. Then, using the softmax function, the necessary appropriate actions to be taken are classified according to the risk. The softmax function can be given as,

$$\sigma(\vec{z})_i = \frac{e^{Z_i}}{\sum_{j=1}^{K} e^{Z_j}} \quad (24)$$

where the zi values signify how serious a danger of SLA violation it is. By guaranteeing that the output values of the function add up to 1, the normalization factor at the formula's core creates an adequate probability distribution. The Softmax function is used to translate the training results in this case between 0 and 1.

4. Result and Discussion

The execution of the specified task was assessed using the following criteria.

(i) Prediction performance over 5 minutes interval

Table 1. Shows the estimated CPU usage with actual observed values

Methods	Time period (5 mins interval)												
	11.00	11.05	11.10	11.15	11.20	11.25	11.30	11.35	11.40	11.45	11.50	11.55	12.00
Observed data	452	460	475	485	495	499	510	517	532	545	548	550	558
SVM	437	445	465	455	454	460	485	500	498	570	514	565	570
RNN	428	428	432	550	460	450	485	590	570	518	504	589	580
CNN	460	450	480	499	490	510	515	510	528	540	545	560	545
Proposed LOLSTM	450	459	470	484	494	501	508	518	532	544	545	551	557

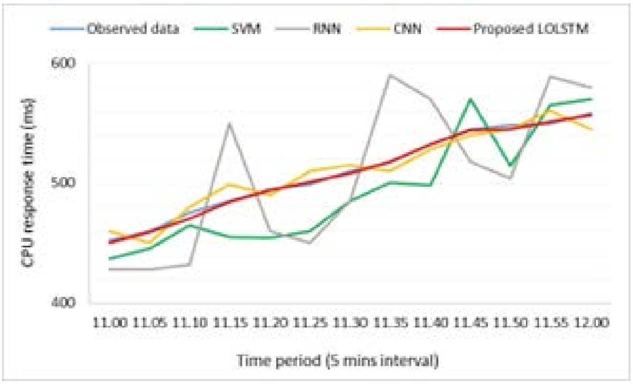

Figure 4. Performance of CPU response time over 5 minutes interval

In comparison to RNN, SVM, and CNN, the proposed LOLSTM produces the greatest overall results. The results of the predicted and observed values are given for 5-min intervals and units are measured in millisecond (ms). Figure 4 shows that, when compared to other current methodologies, the proposed LOLSTM model predicts the resource value approximately closer to the observed value.

(ii) Error rate

MAD and RMSE are used to estimate the correctness of the model. Table 2 shows the error rate for various suggested and existing approaches.

Table 2. Error rate for various prediction methods

Prediction methods	RMSE	MAD
RNN	0.458	0.292
SVM	0.309	0.254
CNN	0.213	0.224
Proposed LOLSTM	0.127	0.107

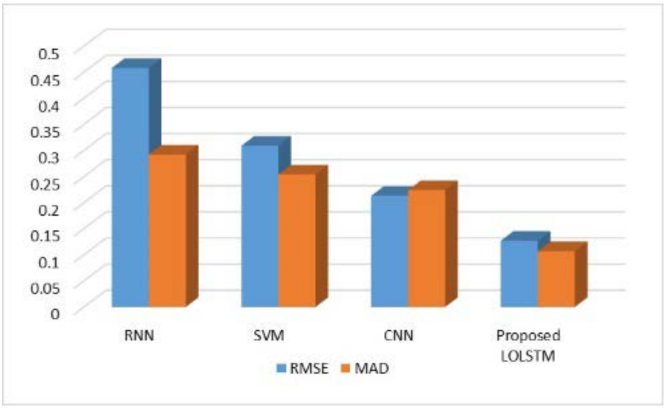

Figure 5. Error rate for proposed and existing methods

Figure 5 shows how often suggested and existing techniques fail. RMSE and MAD is comparatively low for the proposed LOLSTM technique when compared to RNN, SVM and CNN techniques. The suggested model is optimized for efficiency by selecting the appropriate hyperparameter values during training. The number of hidden layers in the model is another factor that can be optimized for high accuracy and low error rate.

(iii) The Receiver Operating Characteristics (ROC)

The ROC curve appears to be a significant signal for categorizing and identifying issues. The signal from the noise is separated using a probability curve known as a ROC by contrasting the TPR against the FPR at various threshold levels.

TPR, which is often referred to as sensitivity, is a gauge of how accurately the negative class is computed. The specificity, or FPR, determines how much the model has overestimated the negative class. The ROC curves for the recommended and current approaches are shown in Table 3.

(iv) Receiver Operating Characteristics (ROC)

The probability curve known as the ROC curve serves as the fundamental statistic for location and classification issues. Additionally, the ROC curve is employed to separate the noise signals in order to compare FPR and TPR at various threshold values. TPR nothing but the True Positive Rate is a sensitivity measure used for approximation of negative class accurately. Whereas the FPR which is the False Positive Rate determines the extent to where the negative class has been overestimated by the proposed model.

Table 3. ROC curves for the proposed and existing methods are displayed.

False Positive Rate	True Positive Rate			
	RNN	SVM	CNN	Proposed LOLSTM
0	0	0	0	0
0.05	0.83	0.87	0.89	0.95
0.1	0.84	0.88	0.9	0.96
0.2	0.85	0.88	0.91	0.97
0.3	0.85	0.89	0.92	0.98
0.4	0.86	0.9	0.93	0.99
0.5	0.87	0.91	0.94	0.99
0.6	0.88	0.92	0.94	0.99
0.7	0.89	0.92	0.95	0.99
0.8	0.9	0.93	0.96	0.99
0.9	0.9	0.93	0.97	0.99
1	0.91	0.94	0.98	0.99

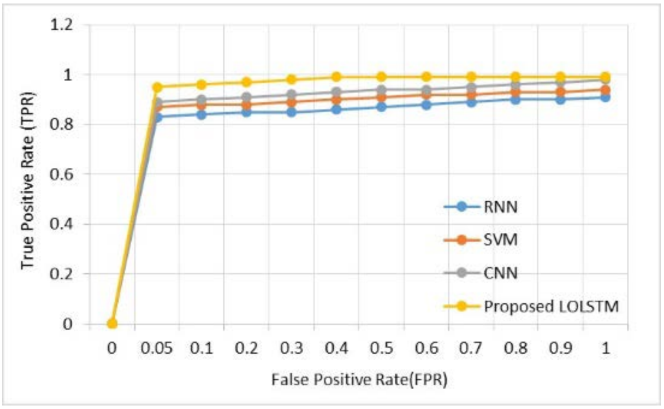

Figure 6. ROC curve for proposed and current methods

The ROC curve for both recommended and established approaches is displayed in Figure 6. When compared to previous works, the suggested work accurately predicts the need for resources, according to the ROC curve that is more pronounced in the upper left corner.

(i) Training and Validation loss

The loss function, which calculates the model's prediction error, is a crucial component of neural networks.

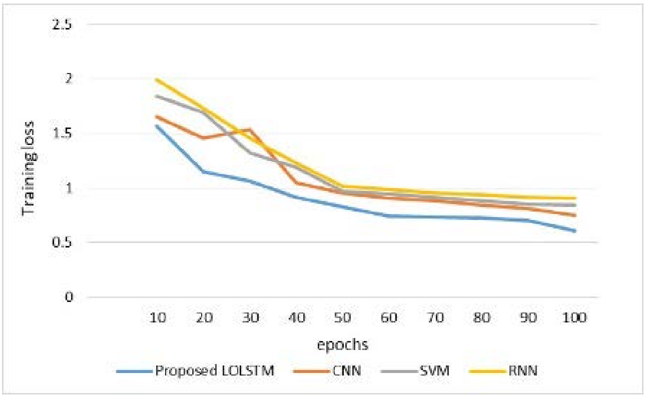

Figure 7. Shows the training loss comparison

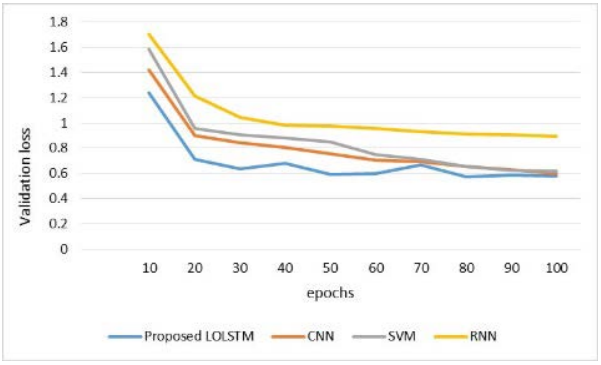

Figure 8. Illustrates the Validation loss comparison

Figure 7 and 8 illustrates the graphical performance of proposed LOLSTM and other techniques such as SVM, RNN, and CNN during the training and validation stages. The suggested LOLSTM's training and validation losses at epoch 100 are 0.6092 and 0.5828, respectively. The proposed method has a minimal training and validation loss when compared to previous methods.

Also, Validation loss of proposed LOLSTM is low when compared to training loss which indicates the model is fitting to new data and there is no overfitting problem. The model generalizes well for all type of new data.

(ii) Training and Validation accuracy

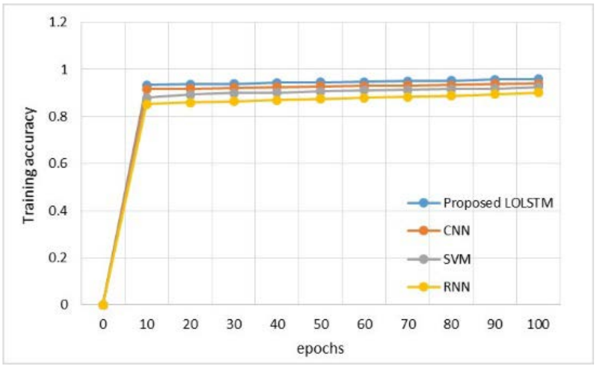

Figure 9. Comparison of training accuracy

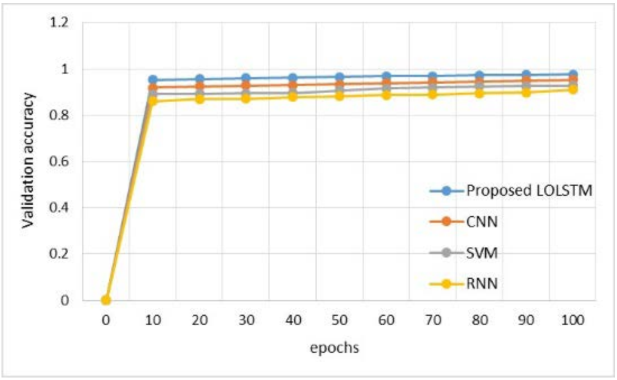

Figure 10. Illustrates the comparison of Validation accuracy

Moreover, Figures 9 and 10 accurately show how the planned LOLSTM performed during the training and validation phases. At epoch 100, validation accuracy is 97.6% and training accuracy is 95.9%. Using the LOLSTM, higher training and validation accuracy scores were obtained than with the other methods. The suggested LOLSTM model is trained by adjusting hyperparameters and hidden layers. Hyperparameters like learning rate and epochs will be adjusted in this effort in order to produce the best prediction results.
The suggested model is validated using the training data throughout each training session, and the results are validated. The proposed LOLSTM model predicts the resource value approximately closer to the observed value. When compared to previous works, the suggested work accurately predicts the need for resources, according to the ROC curve that is more pronounced in the upper left corner. Validation loss of proposed LOLSTM is low when

compared to training loss which indicates the model is fitting to new data and there is no overfitting problem. Using the LOLSTM, higher training 95.9% and validation accuracy 97.6% scores were obtained than with the other methods.

Conclusion

In this work, the resource requirements for future time slots are predicted using LOLSTM technique. By include hidden layer adjustment and hyperparameter optimization during training, the suggested model's accuracy is increased. It regularizes the weights of the network and avoids overfitting. In addition, the proposed work also takes necessary actions if the SLA violation is recognized by the model. Using the metrics MAD, validation loss, ROC curve, RMSE, training and validation accuracy, and training loss, evaluation process of the proposed model was done. The model's accuracy of 97.6% is 2.3%, 2.3%, 4.7%, and 6.6% higher than that of CNN, SVM, and RNN, respectively. Additionally, it was discovered that this proposed work had a 5% higher efficiency than the current works.

Acknowledgements

The author would like to thank the editors and reviewers for their work. No specific grant was given to this research by funding organizations in the public, private, or not-for-profit sectors.

References

[1] Tseng F-H, Wang X, Chou L-D, Chao H-C, Leung VCM. Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm. IEEE Syst J. 2018; 12(2):1688–1699.

[2] Nguyen HM, Kalra G, Kim D. Host load prediction in cloud computing using Long Short-Term Memory Encoder–Decoder. J Supercomput. 2019; 75(11):7592–7605.

[3] Song T, Wang Y, Li G, Pang S. Server consolidation energy-saving algorithm based on resource reservation and resource allocation strategy. IEEE Access. 2019; 7:171452–171460.

[4] Karthiban K, Raj JS. An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm. Soft Comput. 2020; 24(19):14933–14942.

[5] Baig S-U-R, Iqbal W, Berral JL, Erradi A, Carrera D. Adaptive prediction models for data center resources utilization estimation. IEEE Trans Netw Serv Manag. 2019; 16(4):1681–1693.

[6] Chien W-C, Lai C-F, Chao H-C. Dynamic resource prediction and allocation in C-RAN with edge artificial intelligence. IEEE Trans Industr Inform. 2019; 15(7):4306–4314.

[7] Yu P, Zhou F, Zhang X, Qiu X, Kadoch M, Cheriet M. Deep learning-based resource allocation for 5G broadband TV service. IEEE Trans On Broadcast. 2020; 66(4):800–813.

[8] Zhang Q, Yang LT, Yan Z, Chen Z, Li P. An efficient deep learning model to predict cloud workload for industry informatics. IEEE Trans Industr Inform. 2018; 14(7):3170–3178.

[9] Haytamy S, Omara F. A deep learning based framework for optimizing cloud consumer QoS-based service composition. Computing. 2020; 102(5):1117–1137.

[10] Praveenchandar J, Tamilarasi A. RETRACTED ARTICLE: Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing. J Ambient Intell Humaniz Comput. 2021; 12(3):4147–4159.

[11] Hussain W, Hussain FK, Hussain O, Bagia R, Chang E. Risk-based framework for SLA violation abatement from the cloud service provider's perspective. Comput J. 2018; 61(9):1306–1322.

[12] Banerjee S, Roy S, Khatua S. Efficient resource utilization using multi-step-ahead workload prediction technique in cloud. J Supercomput. 2021; 77(9):10636–10663.

[13] Rendyk. 2021 Tuning the hyperparameters and layers of neural network deep Learning. Analytics Vidhya. [accessed 2023 Aug 25]. https://www.analyticsvidhya.com/blog/2021/05/tuning-the-hyperparameters-and-layers-of-neural-network-deep-learning.

[14] Xiao H, Sotelo MA, Ma Y, Cao B, Zhou Y, Xu Y, Wang R, Li Z. An improved LSTM model for behavior recognition of intelligent vehicles. IEEE Access. 2020; 8:101514–101527. doi:10.1109/access.2020.2996203. http://dx.doi.org/10.1109/access.2020.2996203.

[15] Gul B, Khan IA, Mustafa S, Khalid O, Khan A ur R. CPU–RAM-based energy-efficient resource allocation in clouds. J Supercomput. 2019; 75(11):7606–7624.

[16] Anoop S, Singh JAP. RETRACTED ARTICLE: Multi-user energy efficient secured framework with dynamic resource allocation policy for mobile edge network computing. J Ambient Intell Humaniz Comput. 2021; 12(7):7317–7332.