

# MVMS: RNN BASED PRO-ACTIVE RESOURCE SCALING IN CLOUD ENVIRONMENT

### RIDDHI THAKKAR, DHYAN THAKKAR, AND MADHURI BHAVSAR $\ddagger$

Abstract. Cloud computing offers various services to its users, ranging from infrastructure, and system development environment, to software as a service over the internet. Having such promising services available over the internet consistently, it has become an ever-demanding facility. As a reliable services provider, a cloud service provider (CSP) needs to deliver its services seamlessly to users and is also required to optimally utilize the resources. Optimal resource utilization eliminates over and under-provisioning and improves the availability of cloud services. Therefore, it is a great need to have a model allowing CSP to systematize its resources to cater to customers' demands. Such a model should be computationally light and quick enough to produce effective results. In this work, a simple yet effective neural network-based resource prediction model named MVMS is proposed, which enables a CSP to predict the customer's resource demand in advance. The results show that compared to GRU, the proposed Multi-Variate Multi-Step (MVMS) model predicts the resources accurately. Thus, CSP can schedule the resources precisely and process real-time requests of users. Experiments on the bitbrains dataset indicate that the proposed MVMS resource prediction model is quick and accurate, with lower RMSE and MAE values.

Key words: Cloud Computing, RNN, Multi-step Resource Prediction, Elasticity, Multivariate Resource Prediction, Pro-Active scaling

## AMS subject classifications. 68M14, 68T07

1. Introduction. A new dawn of data eruption has taken place with the advancement in hardware and software technologies and the way such services are available to individuals or organizations. As a consequence, high demand for data computing and storage emerged. With seamless computing power, network connectivity, and storage capabilities, cloud computing has taken the lead in solving such issues by providing an on-demand, infinite pool of resources through the internet and charging them on a pay-per-usage basis.

Cloud computing has become the first choice of numerous industries, organizations, education sectors, social media, and many others. All such applications generate a big data deluge that must be processed in real-time, necessitating the elastic scaling of cloud resources. A CSP must allocate resources optimally in order to meet all demands seamlessly. Resource allocation requires to perform dynamically based on the velocity of the input data stream, which requires the elastic nature of resource scaling. Optimized resource allocation necessitates precise resource consumption prediction, which aids in proactive and real-time decision-making. The importance of resource prediction in the cloud is depicted in figure 1.1.

The volume of streaming data is highly fluctuating in a cloud environment. Thus, it has always been challenging to accurately predict resource utilization in such a dynamic environment. The existing approaches for resource prediction include statical methods and conventional neural networks. Examples of statistics-based classical time series forecasting methods include: Autoregression (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA) and its other variations, Simple Exponential Smoothing (SES), Holt's linear trend method, Holt-Winters Exponential Smoothing (HWES), and others [28]. Shyam et al. [1] utilized a bayesian model to predict long and short-term resource necessities for resource-intensive applications. Singh et al. [2] operated linear regressor (LR), ARIMA, and SVR models for predicting non-stationary workloads for web applications to reduce resource provisioning inconstancy. In their extended work [33], the authors proposed triangulation resource provisioning (TRP) approach to provide optimal resource utilization for an hourly billing cycle. Such classical approaches were unable to adequately

<sup>\*</sup>Institute of Technology, Nirma University, Ahmedabad, India (18ftphde29@nirmauni.ac.in)

<sup>&</sup>lt;sup>†</sup>Institute of Technology, Nirma University, Ahmedabad, India (20bec027@nirmauni.ac.in)

<sup>&</sup>lt;sup>‡</sup>Institute of Technology, Nirma University, Ahmedabad, India (madhuri.bhavsar@nirmauni.ac.in)

# Riddhi Thakkar, Dhyan Thakkar, Madhuri Bhavsar

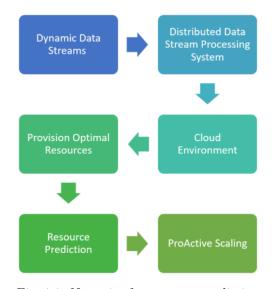


Fig. 1.1: Necessity for resource prediction

capture the non-linear patterns in time series data and were heavily reliant on the stationary nature of collected data. On the other hand, the inherent computational capabilities of neural networks allow the model to easily identify the intricate nonlinear relationship between the response variable and its predictor variables.

Cloud Service Providers need to continuously observe resource utilization to predict resource utilization, which is time-series data. Each point in time-series data is expressed by a time paired with one or more values, chronologically ordered. Due to its intrinsic nature as a time series, data may contain seasonal and nonseasonal cycles, different trends, missing values, outliers, and complex affinities among the variables. With the complex neuron architecture, deep neural networks can easily learn from time-series data and infer valuable information.

Karim [3] et al. proposed a hybrid RNN model to deal with the non-linearity of input data efficiently while foreseeing future CPU usage. The authors took into account a CPU usage parameter of historical workload. However, for multiple time-stamp predictions in advance, the proposed approach MVMS considers two parameters: CPU utilization and the CPU core. CPU cores are monitored along with the CPU utilization parameter as CPU allocation depends on the number of physical CPU cores available. Resource allocation for upcoming data streams can be accurately planned using resource prediction for multi time-stamps.

1.1. Research Contributions. Following are the major contributions of the paper:

- In this work, an RNN-based Pro-Active resource scaling model, namely MVMS, is proposed which utilizes the LSTM approach with the most promising hyperparameter settings to accurately predict multi-step resource utilization in advance.
- For precise multistep prediction, MVMS is trained over multiple variables to capture a nonlinear correlation between historical values of dependent and independent variables. Thus, the model is able to extract the complex hidden patterns from historical data and accurately predict the future resource demand to handle incoming requests with negligible errors.
- The model has been tuned by executing various combinations of the batch size and train:test dataset splitting ratio and identified the best set of values for both of them. This set of hyperparameters generates optimal CPU usage for individual VMs and helps in improving the long-term accuracy of the model.
- The proposed model is evaluated against three different evaluation metrics. The results show that compared to the GRU model, the proposed Multi-Variate Multi-Step (MVMS) resource prediction model gives less error and predicts the resources accurately. The analysis of results establishes that the proposed approach is fast at generating highly accurate resource utilization prediction for multiple

timestamps, considering the multiple variables during the prediction phase.

**1.2. Organization.** The structure of this paper is systematized as follows. Section 2 disuses the related work. Section 3 discusses the motivation behind this work. Section 4 elaborates on technical and conceptual details behind the proposed work. Section 5 introduces the architecture of the proposed model and discusses the dataset along with its pre-processing. Section 6 presents the experimental setup and model evaluation results. The last section 7 concludes and summarizes the complete paper along with the scope of future work.

2. Related Work. The remarkable change in the trend of resource utilization in the recent technological era has made a drastic difference in resource prediction approaches. Borkowski et al. [4] proposed ANN-based model for accurate resource provisioning prediction. As the author applied the offline machine learning approach, the model couldn't accurately respond to unseen data patterns that may appear during real-time prediction. Singh et al. [2] utilized Linear Regressor (LR), ARIMA, and SVR models for predicting non-stationary workloads for web applications to reduce resource provisioning inconstancy. LR is utilized to linearly classify the workload and in cases where the non-linear model is unable to predict better results. LR and SVR predict the slow-scale workload, wherein ARIMA is used to predict the fast-scale workload.

Chen et al. [5] proposed a graph-based deep hybrid probabilistic forecasting framework named Graph Deep Factors (GraphDF), which consists of a relational local and global model. GraphDF aims to improve prediction accuracy and computational efficiency.

Rahmanian et al. [6] proposed learning automata (LA) based, ensemble resource utilization forecasting algorithm for precise resource prediction in a virtualized environment.

Malik et al. [7] predicted multi-resource utilization with a hybrid model named FLGAPSONN consisting of GA-PSO (genetic algorithm - particle swarm optimization) and FLNN (functional link neural network). The authors utilized a combination of GA and PSO algorithms for training the model with high accuracy and FLNN for resource prediction. For multivariate resource prediction, Xu et al. [8] utilized the sliding window approach named S-MTF for converting multivariate time series data into supervised time-series data for predicting future resource usage with a modified GRU model.

Prasad and Madhuri [30] proposed a resource monitoring approach with reinforcement learning and machine learning concepts. Thonglek et al. [9] used an LSTM model to accurately predict resource allocation for a given job. Two-layered LSTM discovers the trade-off between resource allocation and usage, and CPU and memory usage. Kumar et al. [10] predicted cloud workload to reduce operational cost and SLA violations with the evolutionary algorithm and artificial neural network (ANN). The evolutionary algorithm reduces the effect of initial parameter selection. It shows a significant improvement in value selection for mutation and crossover rates.

Mason *et al.* [11] implemented an evolutionary neural networks (NN) approach to predict CPU utilization to reduce energy efficiency while dynamically scaling the resources. However, It failed to deliver better accuracy while predicting multiple future steps. To predict resource utilization, Zhu et al. [12] proposed long-term short-term memory (LSTM) encoder-decoder network with an attention mechanism. The attention mechanism gives importance to the parameters having a high impact on prediction results by assigning them more weight. However, based on the attention-based LSTM model results, the authors concluded that the attention mechanism is not having any positive or negative impact on the performance of the model. Chudasama and Bhavsar [29] proposed deep learning and queuing theory-based short-term resource prediction approach. The proposed hybrid approach utilizes the Bi-directional LSTM model to predict resources for one hour based on historical resource utilization.

Zhang et al. [13] proposed CPW-EAMC, wherein CEEMDAN-PE-Wavelet (CPW) reduces noise in input data and ENN-Attention-MLP-Context (EAMC) predicts multidimensional output for physical machines. The approach proposed in this work is not suitable for real-time resource prediction as it requires complete knowledge of the effects of each dimension on hardware. Tran et al. [14] identified highly correlated features through a fuzzy selection approach to improve prediction accuracy. In this approach, at every prediction window, the relationship between parameters is identified, which is not possible when dealing with a real-time workload.

Song et al. [15] applied the LSTM model to predict single-step host load to efficiently schedule resource allocation and optimally utilize them. Karim et al. [3] used CPU utilization for resource prediction whereas the MVMS approach operates CPU utilization, available CPU core, and nonstationary timestamp as an input to the model and predicts multi-step CPU utilization in advance. In [3] authors used the Bitbrains [16] dataset for their model training and evaluation. In this work, the same dataset is evaluated for the training and validation of the model.

Yang et al. [31] proposed a dynamic and automatic resource admission control approach that precludes resource-ceasing situations due to the unavailability of resources. The proposed approach is implemented in Hadoop YARN framework. The authors stated that this is a novel work in the said framework.

Table 2.1 lists the methodology, dataset, model evaluation metrics, and error metrics used in the literature for resource prediction.

Paper	Model Used	Dataset	Model Evaluation Metrics	Error met- rics	Multi vari- able Pre- dic- tion	Multi Step Pre- dic- tion
[4]	ANN	Travis CI and GitHub	per-task duration of different tasks,	RMSD	×	×
[1]	Bayesian Model	AmazonEC2, Google CEData- Centers [17]	vCPU instance	MSE	×	×
[30]	Re-inforcement Learning, LSTM	Bitbrains [16]	CPU utilization, Disk read/write Through- put and Memory utilization	MAE, MSE, RMSE		×
[29]	Queuing the- ory, Bidirec- tional LSTM	Private Cloud dataset	workload of a web server	MAE, MSE, RMSE	×	
[2]	LR, ARIMA, SVR	ClarkNet, NASA	HTTP requests	MAE, MSE, RMSE, MAPE	×	×
[9]	LSTM	Googles cluster- usage trace [18]	Requested CPU and memory resource, Used CPU and mem- ory resource	-		×
[10]	Evolutionary Algorithm	NASA, Saskatchewan	Number of HTTP re- quests	RMSE	×	
[31]	Novel admis- sion control mechanism	Classic MapRe- duce	CPU Utilization	Makespan	×	×
[11]	Evolutionary Neural Net- works (NN)	PlanetLab work- load trace [19]	CPU utilization	MAE, MSE	×	
[14]	MF-LSTM	Googles cluster- usage trace [18]	CPU and Memory us- age	MAE		×

Table 2.1: Comparative analysis of state-of-the-art techniques for resource prediction and proposed approach

[15]	LSTM	Googles cluster- usage trace [18], Traditional Dis- tributed System [20]	CPU Usage	MSE, MSSE	×	
[3]	BHyPreC: Bi- LSTM Based Hybrid RNN	Bitbrains [16]	CPU Usage	MAE, MSE, RMSE, MAPE	×	
[5]	GraphDF: Graph Deep Factors	Google Trace, Adobe Workload Trace, Graph Construction	CPU Usage	MAPE	×	
[6]	Learning au- tomata	CoMon project [21]	CPU Usage	RMSD, error ratio, absolute Error	×	
[7]	FLGA PSONN	Google cluster workload traces [17]	CPU and memory Us- age	MAE		×
[8]	GRU	Alibaba, Google cluster workload traces [22]	CPU and memory Us- age	MSE, RMSE, MAPE		×
MVMS	Multi-variate, Multi-step LSTM	Bitbrains [16]	CPU utilization, CPU Cores	MAE, RMSE, MSE		

**3. Research Motivation.** Real-time stream processing applications running on the cloud receive an enormous amount of data, which needs to process in no-time. Such a scenario requires an infinite pool of processing capabilities available on the easy go, which requires the CSP to predict resource demand well in advance. The following subsection describes the motivation for our work category-wise.

**3.1. Multivariate analysis of time series data.** The cloud performance data contains multiple metrics like timestamp, CPU utilization, memory utilization, disk read-write throughput, and network received-transmitted throughput. While performing resource prediction and analyzing its behavior, it is necessary to consider the parameters that significantly contribute to the system performance. Thus, in this work, along with CPU utilization, CPU cores are considered, as CPU allocation from a physical machine completely depends on the availability of physical CPU cores.

**3.2.** Multi-step resource prediction. In the cloud, resource demand changes with the number of active users and the data streams spawned by the various processes. The resource utilization history of these events contains hidden trends and seasons. Unfolding and identifying the patterns in such data will enable the prediction of the multiple time steps in advance. MVMS forecasts multistep CPU utilization by locating hidden patterns in CPU utilization and CPU core usage. Knowledge of future resource demand enables CSP to avail resources well in advance. MVMS projects the resource utilization for 12 timesteps in the future with a 5-minute interval. CSP can effectively schedule its resources for the future data stream within this time frame.

**3.3.** Proactive scaling. As in the cloud environment, resource demand fluctuates very rapidly. The elastic nature of the cloud allows the scaling of resources in and out as per demand. Herein, multi-step-ahead resource prediction enables CSP to schedule resources in advance. Such practice delivers a swift service experience to cloud users and facilitates a CSP to efficiently utilize the resources, which also improves the energy efficiency of the overall data center.

**3.4.** Non-Stationary Time-series data. Time-series data may have nonstationarity among the parameters. As evident from figure 3.1, CPU utilization data is highly fluctuating in a given time frame. Such behavior

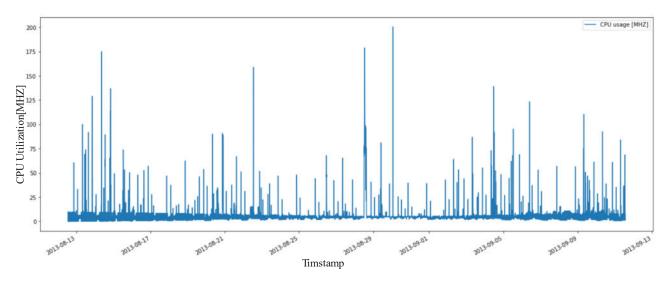


Fig. 3.1: CPU Utilization

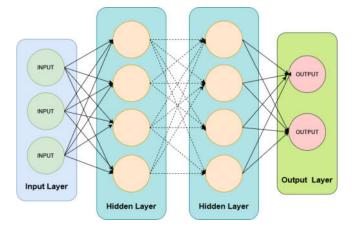


Fig. 4.1: A simplified block diagram of neural network [3]

of parameters significantly impacts the prediction efficiency and accuracy of any model. Thus, non-stationary data requires preprocessing and conversion to stationary data. In this work, the dataset is first transformed to stationary form, and then the proposed model is evaluated for resource usage prediction.

4. Background. This section elaborates on the core concepts of the proposed model. The subsections 4.1 introduce the neural network. Subsections 4.2 and 4.3 discuss LSTM and GRU, respectively.

**4.1. Neural Networks.** A model mimicking a human mind and forming a network of artificial neurons is known as a neural network (NN) or an artificial neuronal network (ANN). A neural network is a series of algorithms designed to unfold the hidden pattern in a collection of data. ANN is composed of an input layer, number of hidden layers, and an output layer. Each layer contains multiple neurons that are connected to each other, having weights and biases. Figure 4.1 depicts a simplified block diagram of a neural network with three input nodes in an input layer, two hidden layers containing four and three neurons, respectively, and the output layer containing one neuron. The neural network learns the hidden patterns from training data and updates the associated weights to accurately predict the output for newly arrived data.

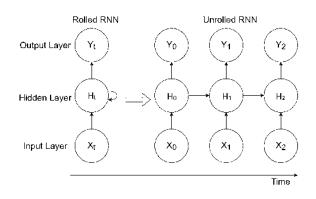


Fig. 4.2: Unrolled RNN [26]

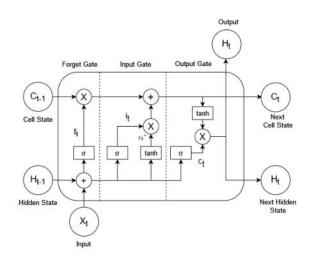


Fig. 4.3: LSTM Cell Architecture

A recurrent neural network (RNN) is a special kind of neural network that features sequential or time-series data. RNN differs by having a memory element, helping it to retain the states or information from previous sequences for predicting the output sequence. RNN is equipped with a feedback loop, where the output of the previous step is feedback to the network, which will be stored to leverage the output of the next step. Figure 4.2 depicts the unrolled RNN sequences, where  $X_t$  is the input values at time-stamp t,  $H_t$  is the hidden state values at time-stamp t, and  $Y_t$  is the output value.

There are two most common versions of RNN: Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM). This research work is formed on the foundation of LSTM architecture. We compared the results of the proposed approach with the GRU model.

**4.2. Long Short-Term Memory.** LSTM is a specialized neural network that accounts for memory effects in sequential data, like time series. LSTM efficiently addresses the back-propagation instability (vanishing gradient problem) found in traditional RNNs. Over longer sequences, the gradient loses parameter updates and becomes negligible in the vanishing gradient problem. LSTM addresses this issue by storing parameter updates for longer time sequence data. The LSTM cell architecture is depicted in figure 4.3.

A single LSTM cell contains three gates: input gate, output gate, and forget gate. Each of these gates contains the activation function, determining which output to preserve and which one to forget.

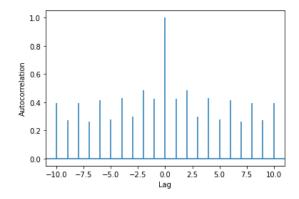


Fig. 5.1: Autocorrelation of CPU Utilization

With a new current input timestamp  $X_t$ , the LSTM cell updates its internal gates as follow [23]:

$$f_t = \sigma(W_f[H_{t-1}, X_t] + b_f)$$
(4.1)

$$i_t = \sigma(W_i[H_{t-1}, X_t] + b_i)$$
(4.2)

$$o_t = \sigma(W_o[H_{t-1}, X_t] + b_o)$$
(4.3)

$$c_t^{\sim} = \tanh(W_c[H_{t-1}, X_t] + b_c) \tag{4.4}$$

$$C_t = f_t * C_{t-1} + i_t * c_t^{\sim} \tag{4.5}$$

$$H_t = o_t * \tanh\left(C_t\right) \tag{4.6}$$

In MVMS, three LSTM units are stacked on top of each other. Through experimental tuning, each layer is operated with 100 memory cells.

**4.3. Gated Recurrent Unit.** GRU uses reset and update gate mechanism for updating the states of hidden neurons. The reset gate decides the amount of information omitted from the previous hidden state, and the update gate determines the amount of information from new input that should be used to update the hidden state. From the comparison between the performance matrices of the proposed model and GRU, it is derived that GRU is not suitable for time-series data with longer dependencies.

5. Multi-variate Multi-step Resource Prediction Model. The autocorrelation plots unfold the hidden patterns in the data and design an accurate prediction model [28]. Figure 5.1 shows the autocorrelation plot of CPU Utilization of a VM with 10 lags, visualizing the long-range dependency. The plot shows a high autocorrelation at lag 0, and then there is the alternate sequence of negative and positive spikes with a negative and positive lag. LSTM appears to be the best solution due to its inherent capacity to handle long-term dependency on volatile data. This section describes the proposed MVMS model and data set.

Figure 5.2 depicts the sequence of steps followed in the present work while developing the prediction model and producing the results.

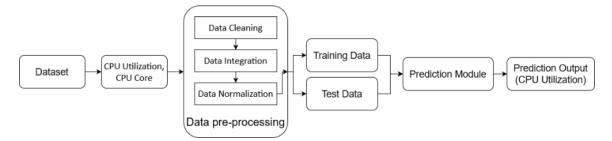


Fig. 5.2: Resource usage prediction sequence

**5.1. Dataset.** The proposed model is evaluated on a dataset on Bitbrains: a distributed data center, which hosts and manages the business computations [16]. The proposed model is evaluated on the fastStorage trace of the Bitbrains dataset. The fastStorage track comprises 1250 VMs interconnected via fast storage area network (SAN) storage devices. The file for each VM includes its performance metrics. The fastStorage dataset in total includes 5,446,811 CPU hours, holding 23,214 GB of memory with 5,501 cores.

After analyzing each VM file, it was observed that timestamps are not evenly distributed. Therefore, each VM file needs to be pre-proposed. Figure 5.3 shows the plot of unique timestamps in the dataset vs. the different timestamps, representing the uneven distribution of timestamps in a dataset. Section 5.2 describes the procedure followed for the dataset preprocessing. Once the dataset is evenly distributed, it is split into different train:test ratios and evaluated on the proposed model with various combinations of other hyperparameters.

5.2. Data pre-processing. The dataset contains the system information, e.g., CPU, memory, network, and disc read-write of 1250 VMs in the cloud. According to the author [16], the data is collected at a regular interval of 300 seconds. With the said time duration of data collection, there should be a total of 8640 entries for each VM. However, several timestamps are not collected at the interval of 300 seconds. These lagging or leading data entries will induce inefficiency in the training of the model with inaccurate prediction. Also, some VMs contain less than 5000 entries, while other VMs have more than 20,000 entries. Subsequently, such data entries lead to a different count of timestamps for each VM with an assorted combination of timestamps. With reference to the original work [16], it is required to have 63130 unique timestamps, starting at 1376314846, and ending at 1378906798 UNIX timestamps. With a total of 2591952 seconds and estimating the readings at 300 seconds apart, there should be a total of 2591952/300 = 8639.84 timestamps. According to [16], there are 8640 readings for each VM, and the timestamps are evenly distributed. However, the overall timestamp distribution is uneven, as depicted in figure 5.3. As there are missing and redundant entries recorded at irregular intervals, it requires processing the dataset before further usage.

The data pre-processing is carried out with the following steps:

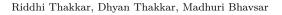
- Data Cleaning
- Data Integration
- Data Normalization

In the first step of data pre-processing, the following iterative equation is used to even the data at time t:

$$X_t = \begin{cases} \sum_{i=0}^{i=T} \sum_{\substack{u=min(T,i+s)\\u=max(0,i-s)}}^{u=min(T,i+s)} \frac{x_i - x_u}{i - u}, & \text{if } x_i \text{ is present} \\ \frac{\int_{i-u}^{i+u} x(v) \, dv\}}{2u}, & \text{if } x_i \text{ is not present} \end{cases}$$
(5.1)

Here, x is the uneven timestamp, X is the even timestamp

The above equation (5.1) removes duplicate readings and takes samples at an interval of 300 seconds. It returns a smoother and more consistent curve for parameters, across all VMs. With a weighted average, it synchronizes the values of all the columns for the missing timestamps found in multiple VMs. As all the columns have been converted to an even timestamp, they can now be further processed by the model.



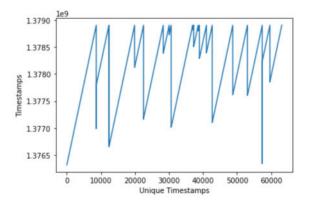


Fig. 5.3: Timestamp Distribution

Table 5.1: Hyper-parameters of the proposed model

Hyper-parameter	Value		
LSTM Layers	3		
Neurons in each LSTM Layer	100		
Dense Layer	1		
Neurons	12		
History window	60		
Prediction window	12		

After the data is sanitized and integrated for 1250 VMs with 8640 entries each, in the second step, a total of 10800000, all at the synchronized timestamps with even 300-second intervals, are aggregated in a single file.

In the third step, the dataset is normalized for faster convergence and efficient performance in a range of 0-1. The MinMax scalar [24] is used for normalization:

$$X\_std = (X - X\_min)/(X\_max - X\_min)$$
(5.2)

$$X \quad scaled = X \quad std * (max - min) + min \tag{5.3}$$

Here, min,  $\max$  = Feature Range

In the next step, the model is evaluated on the processed data.

**5.3.** Proposed Architecture and parameter selection. The LSTM layer consists of hidden layers with one or more neurons in each layer. A neuron is a signal processing unit that takes an input signal and uses an activation function to output a signal [11]. In MVMS, the inputs are the CPU cores and CPU utilization from historical data, and the output is the CPU utilization prediction for the multiple timestamps. By performing parameter sweeps manually and taking an educated guess, it was observed that a network with three stacked LSTMs with 100 neurons in each layer, followed by a single dense layer with 12 neurons, delivered the optimum performance. The observation revealed that more than three LSTM layers did not contribute to any improvement in performance and led to prolonged training time as a consequence of the additional weights to be trained. It was also learned that less than three LSTM layers are not delivering promising performance. The proposed RNN-based model, namely MVMS, is depicted in Figure 5.4, consisting of an input layer followed by three LSTM layers and a dense layer. Table 5.1 lists the hyper-parameters for the proposed model.

**History and Prediction window.** For the sake of brevity, rather than computing all the combinations of the history window with other parameters in the table, the history window value is borrowed from the work

MVMS: RNN based Pro-Active resource scaling in cloud environment

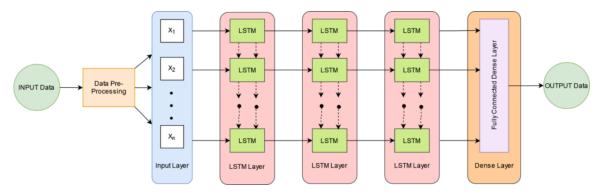


Fig. 5.4: The Proposed RNN based model

in [3] which significantly contributes to their optimal output. The proposed work predicts the CPU utilization for 12 timesteps in advance, providing significant information to CSP for preparing resources in advance to cater to data streams. Figure 5.5 depict the history window and prediction window for the proposed model. Multiple timestep forecasting directs optimal utilization of the cloud data center resources, increase the return of investment (RoI) for CSP, and also helps to reduce the carbon footprint [25].

**5.4.** Dataset Split Analysis. To perform a fair model evaluation and to determine whether there is overfitting or underfitting, the dataset is divided into three different train:test combinations.

- 60:40
- 70:30
- 80:20

In the above m:n split ratios, m% of the complete dataset is used for training, and the remaining n% is used for testing.

5.5. Optimal Batch Size. For the selection of batch size, it is essential to subtract the history and prediction window sizes from the total count of timestamps, for accurate multistep prediction. As per the discussion in section 5.1 and 5.2, each VM contains exactly 8640 unique timestamps with a history window of size 60 and a prediction window of 12 steps. Figure 5.5 depicts the history window and prediction window distribution. The history window and prediction windows overlap at the current value of the timestamp. Thus, adding one extra value makes the final count of timestamps for each VM 8569, which will be useful for prediction. If the batch size is not a perfect divisor of 8569, then in the last batch of the VM, there will be an overlapping of data from the next VM, which would cause the model to not comprehend the spike from the VM change, and hence reduce the performance. The factors of 8569 are 1, 11, 19, 41, 209, 451, 779, and 8569. As the smaller batch size increases the computation time as well as leads to a lower amount of information extracted per batch, and a batch size value lower than the number of neurons does not contribute more to model convergence. Therefore, factors 1, 11, 19, and 41 are not considered for the batch size, and 209, 451, 779, and 8569 were selected as different batch sizes. Now, with different combinations of train:test ratio and batch size, the proposed model is iterated 10, 25, and 50 times.

6. Results & Discussion. This section covers the evaluation of the proposed model. It first describes the metrics used to validate the performance of MVMS in section 6.1. Section 6.2 elaborates the results of the model, and section 6.3 discusses the time complexity of the model.

6.1. Evaluation Metrics. Performance metrics represent how well the model learned from trained data and how accurately it performed during testing. Therefore, to quantify the model performance and how much to improve it, mean absolute error (MAE), mean squared error (MSE), and root mean squared Error (RMSE) [32] was calculated. In this work, performance metrics illustrate the accuracy of CPU utilization predicted by MVMS and GRU. Equations 6.1, 6.2, and 6.3 are the formula used to derive the MAE, MSE, and RMSE,

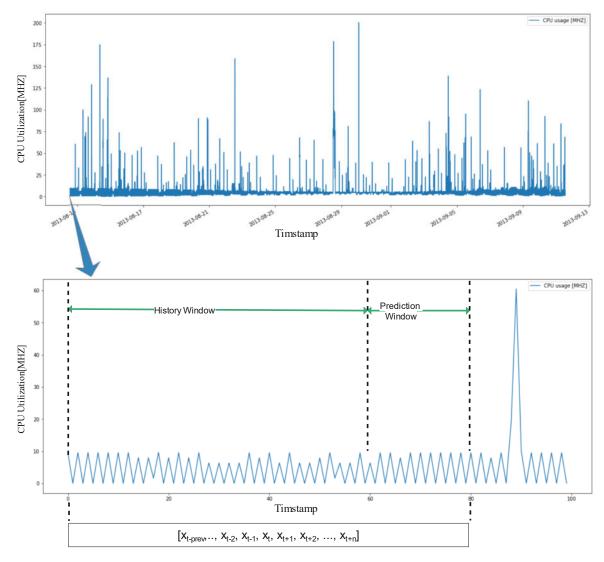


Fig. 5.5: History and Prediction Window

respectively. The lower the value of the error, the better the model performance.

$$MAE = \left(\frac{1}{n}\right)\sum_{i=1}^{n} |y_i - x_i|$$
(6.1)

$$MSE = \left(\frac{1}{n}\right) \sum_{i=1}^{n} (y_i - x_i)^2$$
(6.2)

$$RMSE = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n} (y_i - x_i)^2}$$
(6.3)

MVMS: RNN based Pro-Active resource scaling in cloud environment

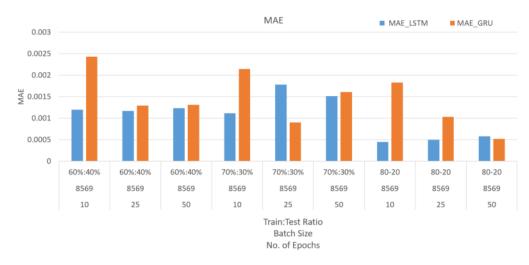


Fig. 6.1: Comparison between MAE of MVMS and GRU with 3 dataset splitting ratio

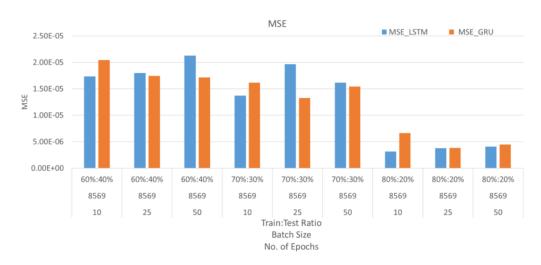


Fig. 6.2: Comparison between MSE of MVMS and GRU with 3 dataset splitting ratio

**6.2.** Performance Evaluation. MVMS is trained and validated using the Bitbrains dataset. The model is trained with various batch sizes and splitting ratios of a dataset to obtain the optimal prediction output.

Initially, the model is evaluated on a dataset split ratio of 60%:40%, and batch size of 8569 over 10, 25, and 50 epochs. Then, the same batch size and number of epochs are implemented over 70%:30% split ratio. A subtle improvement with a bit negative fluctuations in performance metrics is observed. As shown in figure 6.1, the average value of MAE for 70%:30% is high as compared to 60%:40%, whereas in figures 6.2, 6.3 for MSE, and RMSE it is lower than 60%:40% for MVMS and GRU.

Thus, we further updated the dataset split ratio and evaluated the model with the same parameters over 80%:20% ratio, and observed the refinement in performance parameters. By comparing the results for all three dataset splitting combinations, the model performs optimally with 80%:20% ratio as shown in figures 6.1, 6.2, and 6.3.

After identifying the optimal dataset split ratio, the model is iterated over other batch sizes 779, 451, and 209. Figures 6.4, 6.5, and 6.6 show the MAE, MSE, and RMSE for 8569, 779, 451, and 209 batch sizes

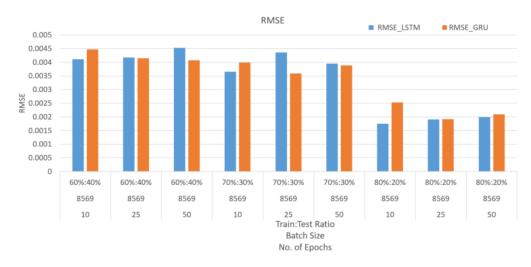


Fig. 6.3: Comparison between RMSE of MVMS and GRU with 3 dataset splitting ratio

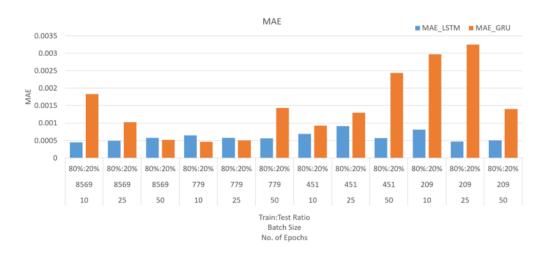


Fig. 6.4: Comparison between MAE of MVMS and GRU for all batch sizes

with 80%:20% dataset splitting ratio. It is deduced that by reducing the batch size, the model under-fits, and performance is reduced. From the performance parameters, we can notice that with the batch size of 8569 and 80%:20% ratio, the model performed optimally, and with the batch size 8569 and 70%:30% ratio, the performance of the model deteriorated. It is observed that the larger the batch size and the data split ratio, the model delivers the optimal performance, whereas with decreasing the dataset splitting ratio, the result decays.

The results show that the proposed approach performs better than the GRU model for all batch sizes and dataset splitting ratios. This work predicts the CPU utilization for multiple timestamps in the future, but GRU failed to maintain the values of hidden neurons for a longer time period and hence failed to predict the CPU utilization with high accuracy.

The proposed neural network architecture for time series prediction of multi-step, multi-variate, and multiagent is inexpensive to train with exceptional results. Here, as the proposed model can predict the resources for individual VMs too, we referred to it as a multi-agent prediction model. The fewer layers in the proposed model reduce the computation complexity, which leads to a faster computation time while accurately converging to MVMS: RNN based Pro-Active resource scaling in cloud environment

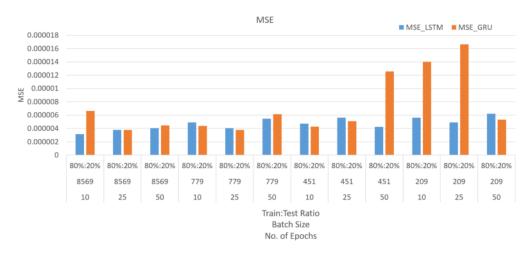


Fig. 6.5: Comparison between MSE of MVMS and GRU for all batch sizes

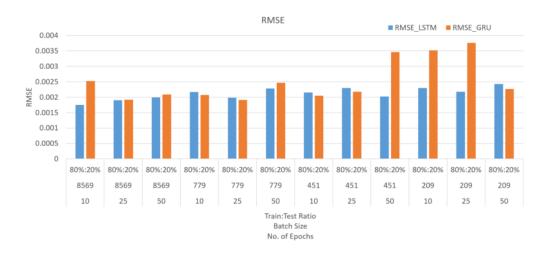


Fig. 6.6: Comparison between RMSE of MVMS and GRU for all batch sizes

the optimal results.

**6.3. Training Time Complexity.** The total time required to train a model depends on the number of training parameters. However, it also depends on the computing environment in which it is executed. As the experiment was conducted on high-end configuration systems, the proposed model took approximately 3 minutes for each epoch with the best configuration. Therefore, it converges in about 30 minutes to the optimal result. However, with the smaller batch size, it took an average of 12 minutes per epoch. Figure 6.7 shows the training time for various batch sizes and epochs. It is observed from the graph that the smaller the batch size, the larger the training time for any combination of dataset splitting ratios.

7. Conclusion and Future Work. The elastic nature of the cloud facilitates users with seamless access to resources on demand. A CSP needs to have enough resources available to provide the required services to the clients. Such action requires having knowledge of resource requirements in advance, allowing CSP to optimally operate the available resources. The proposed RNN-based architecture for time series prediction of multiple-variate and multi-step is inexpensive to train, with exceptional results. For validating the proposed model,



Fig. 6.7: Training Time

MAE, MSE, and RMSE are assessed. The proposed approach outperformed the GRU model by accurately predicting the resources. From the performance parameter values, it is observed that the model with more training data and a larger batch size converges very quickly to the optimal result.

In this work, the correlation between CPU cores and CPU utilization parameters is considered for resource prediction. However, based on the nature of a problem, the correlation among various parameters can be identified using machine learning approaches dynamically.

#### REFERENCES

- G. K. Shyam and S. S. Manvi, "Virtual resource prediction in cloud environment: a bayesian approach," Journal of Network and Computer Applications, vol. 65, pp. 144–154, 2016.
- [2] P. Singh, P. Gupta, and K. Jyoti, "Tasm: technocrat arima and svr model for workload prediction of web applications in cloud," *Cluster Computing*, vol. 22, no. 2, pp. 619–633, 2019.
- [3] M. E. Karim, M. M. S. Maswood, S. Das, and A. G. Alharbi, "Bhyprec: a novel bi-lstm based hybrid recurrent neural network model to predict the cpu workload of cloud virtual machine," *IEEE Access*, vol. 9, pp. 131476–131495, 2021.
- [4] M. Borkowski, S. Schulte, and C. Hochreiner, "Predicting cloud resource utilization," in Proceedings of the 9th International Conference on Utility and Cloud Computing, pp. 37–42, 2016.
- [5] H. Chen, R. A. Rossi, K. Mahadik, S. Kim, and H. Eldardiry, "Graph deep factors for forecasting with applications to cloud resource allocation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 106–116, 2021.
- [6] A. A. Rahmanian, M. Ghobaei-Arani, and S. Tofighy, "A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment," *Future Generation Computer Systems*, vol. 79, pp. 54–71, 2018.
- [7] S. Malik, M. Tahir, M. Sardaraz, and A. Alourani, "A resource utilization prediction model for cloud data centers using evolutionary algorithms and machine learning techniques," *Applied Sciences*, vol. 12, no. 4, p. 2160, 2022.
- [8] M. Xu, C. Song, H. Wu, S. S. Gill, K. Ye, and C. Xu, "Esdnn: Deep neural network based multivariate workload prediction approach in cloud environment," arXiv preprint arXiv:2203.02684, 2022.
- K. Thonglek, K. Ichikawa, K. Takahashi, H. Iida, and C. Nakasan, "Improving resource utilization in data centers using an lstm-based prediction model," in 2019 IEEE International Conference on Cluster Computing (CLUSTER), pp. 1–8, IEEE, 2019.
- [10] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Generation Computer Systems*, vol. 81, pp. 41–52, 2018.
- [11] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host cpu utilization in the cloud using evolutionary neural networks," *Future Generation Computer Systems*, vol. 86, pp. 162–173, 2018.
- [12] Y. Zhu, W. Zhang, Y. Chen, and H. Gao, "A novel approach to workload prediction using attention-based lstm encoderdecoder network in cloud environment," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–18, 2019.
- [13] Y. Zhang, F. Liu, B. Wang, W. Lin, G. Zhong, M. Xu, and K. Li, "A multi-output prediction model for physical machine resource usage in cloud data centers," *Future Generation Computer Systems*, 2022.
- [14] N. Tran, T. Nguyen, B. M. Nguyen, and G. Nguyen, "A multivariate fuzzy time series resource forecast model for clouds

using lstm and data correlation analysis," Procedia Computer Science, vol. 126, pp. 636-645, 2018.

- [15] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," The Journal of Supercomputing, vol. 74, no. 12, pp. 6554–6568, 2018.
- [16] S. Shen, V. Van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," pp. 465–474, IEEE, 2015.
- [17] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the third ACM symposium on cloud computing*, pp. 1–13, 2012.
- [18] J. Wilkes, "More google cluster data," Google research blog, Nov, 2011.
- [19] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.
- [20] P. A. Dinda, "Load traces on unix systems.."
- [21] K. Park and V. S. Pai, "Comon: a mostly-scalable monitoring system for planetlab," ACM SIGOPS Operating Systems Review, vol. 40, no. 1, pp. 65–74, 2006.
- [22] "More google cluster data. google research blog."
- [23] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," Physica D: Nonlinear Phenomena, vol. 404, p. 132306, 2020.
- [24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [25] R. Thakkar, R. Trivedi, and M. Bhavsar, "Experimenting with energy efficient vm migration in iaas cloud: Moving towards green cloud," in *International Conference on Future Internet Technologies and Trends*, pp. 56–65, Springer, 2017.
- [26] Wikipedia contributors, "Recurrent neural network Wikipedia, the free encyclopedia.". [Online; accessed 15-May-2022]
- [27] Wikipedia contributors, "Feedforward neural network Wikipedia, the free encyclopedia.". [Online; accessed 14-May-2022].
   [28] R. J. Hyndman and G. Athanasopoulos, Forecasting: principles and practice. OTexts, 2018.
- [29] Chudasama, V. & Bhavsar, M. A dynamic prediction for elastic resource allocation in hybrid cloud environment. Scalable Computing: Practice And Experience. 21, 661-672 (2020)
- [30] Prasad, V. & Bhavsar, M. Monitoring and prediction of SLA for IoT based cloud. Scalable Computing: Practice And Experience. 21, 349-358 (2020)
- [31] Yang, Z., Bhimani, J., Yao, Y., Lin, C., Wang, J., Mi, N. & Sheng, B. AutoAdmin: automatic and dynamic resource reservation admission control in Hadoop YARN clusters. *Scalable Computing: Practice And Experience.* 19, 53-68 (2018)
  [32] Holden, K., Peel, D. & Thompson, J. Economic forecasting: an introduction. (cambridge university Press, 1990)
- [33] Singh, P., Gupta, P. & Jyoti, K. Triangulation resource provisioning for web applications in cloud computing: A profit-aware approach. Scalable Computing: Practice And Experience. 20, 207-222 (2019)

*Edited by:* Katarzyna Wasielewska *Received:* Jun 16, 2022 *Accepted:* Mar 3, 2023