A Weighted Ensemble of VAR and LSTM for Multivariate Forecasting of Cloud Resource Usage

Jyoti Shetty*, Karthik Cottur, G. Shobha, and Y. R. Prajwal

Department of Computer Science, RV College of Engineering, Bangalore, India; Email: {karthikcottur.cs16, Shobhag, prajwalyr.cs17}@rvce.edu.in (K.V., G.S., Y.R.P.)
*Correspondence: jyothis@rvce.edu.in (J.S.)

***Abstract*—Forecasting resource usage values of a cloud service has ample applications such as service performance management, auto-scaling, capacity planning, and so on. While univariate forecasting techniques are the focus of current research, multivariate forecasting is rarely explored. This research work focuses on multivariate forecasting of resource usage values believing that there exists interdependency among the features of the underlying system that must be considered while forecasting. At first, the interdependency among the attributes is verified using Granger causality tests. Then the research explores various forecasting approaches — univariate Multi-Layer Perceptron (MLP), univariate Long Short Term Memory (LSTM), multivariate Vector Autoregression (VAR), and multivariate stacked LSTM. Further based on the observations of performances of these models the research proposes an implementation of a weighted ensemble of VAR and LSTM models to forecast key cloud resource usage metrics. The models thus proposed are implemented and validated using the publicly available GWA-T-12 Bitbrains time series dataset. The results show that the multivariate models outperform univariate models with lesser Normalised Root Mean Square Error (NRMSE) values. Also, the multivariate stacked LSTM outperforms VAR and the proposed ensemble forecasting model with lesser NRMSE values within a range of 1–5% for various resources across different lag values.**

***Keywords*—multivariate forecasting, cloud resource usage forecasting, Long Short Term Memory (LSTM), stacked LSTM, Vector Autoregression (VAR), ensemble forecasting**

I. Introduction

Cloud computing is the amalgamation of service-oriented computing and utility computing paradigms where both hardware and software are provided as-a-service model [1]. The cloud computing environment is distributed, dynamic, shared, and elastic. These characteristics of the cloud makes cloud service performance unpredictable and vary over time. Thus, cloud service performance management is challenging and requires to have intelligent and informed management. Cloud service performance management involves operations such as capacity planning, auto-scaling, fault management, and so on. Currently, these performance management operations use a threshold-based reactive approach which is ineffective and time-consuming, affecting the quality of cloud services. For example, in Amazon Web Services (AWS) when a user sets the scale-out threshold policy to 85% of CPU utilization then as the CPU utilization of the service reaches 85% a new Virtual Machine (VM) is spawned, however, the quality of service may decrease by the time a new VM is spawned [2]. The disruption or decrease in performance of service can be avoided by forecasting resource usage of the service and auto-scaling policy based on the forecasted values. Such a proactive auto-scaling operation which can initiate the scaling operation before the threshold is reached can be called proactive [3], for instance, the current auto-scaling operation is into action once it detects the performance degradation, and the service performance gets affected until the auto-scaling operation is completed. With forecasting, the auto-scaling operation can be initiated well before the service performance hits low and thus prevent the performance degradation effect.

The performance of services running on the cloud is defined by multiple variables like CPU usage, Memory usage, Cache usage, Disk usage, etc. The values are collected over time to form multivariate time series data. Formally a multivariate time-series data T is a matrix on $m\ x\ n$ where m is the number of variables and n is the number of observations.

$$T = (t[m]_1, \ldots\ldots, t[m]_n),$$

$t[m]_i$ is a vector of m real-valued variables for n observations.

Such time-series data can be used to train statistical forecasting models to forecast future data. Univariate forecasting takes a single attribute as input for forecasting the corresponding future values of the attribute. A univariate model does not capture the interactions among various attributes that define the performance of service. But a resource usage value may not only depend on its past

Manuscript received September 1, 2022; revised November 12, 2022; accepted December 22, 2022; published March 28, 2023.

values but also depend on other resources' past values, i.e., there may exist a dependency relation between multiple resource usage values, which is ignored in univariate forecasting. Multivariate forecasting considers dependency among the features for forecasting future values. Hence it is proposed to use a multivariate model where multiple relevant attributes are taken as input to forecast multiple outputs simultaneously. Such a model is believed to capture the correlation among the variables to provide realistic forecasting.

In multivariate forecasting there can be multiple independent variables and multiple dependent variables [4]. The commonly used approaches for multivariate time series forecasting algorithms are multivariate Long Short Term Memory (LSTM), multivariate Vector Autoregression (VAR), multi-layer perceptrons, and so on [4, 5].

Recurrent networks (RNN) use feedback connections to store activations, which are representations of recent inputs. As a result, RNN is better suited for short-term memory forecasting rather than long-term memory. Traditional backpropagation-through time results in exploding or vanishing gradients. The LSTM architecture is a gradient-based learning algorithm that gets rid of this problem by carrying over long-term dependencies [6]. The LSTM cell used is shown in Fig. 1.

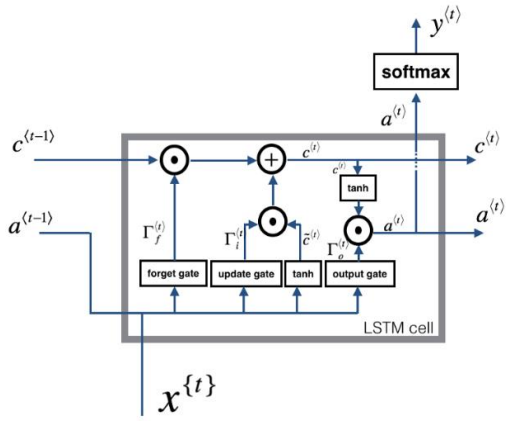

Figure 1. LSTM cell.

Three gates are used to decide whether a long-term dependency is passed on or a new dependency is passed on. The forget gate if activated increases the value of $c^{(t-1)}$ and carries the dependency while forgetting the current value which was calculated. The forget gate is as in Eq. (1).

$$\Gamma_f^t = \sigma\left(w_f\left[a^{(t-1)}, x^t\right] + b_f\right) \quad (1)$$

The update gate is as shown in Eq. (2)

$$\Gamma_u^t = \sigma\left(w_u\left[a^{(t-1)}, x^t\right] + b_u\right) \quad (2)$$

The update gate if high, updates $c^{(t)}$ with the current calculated value of $c^{(t)}$ given by Eq. (3) and Eq. (4)

$$\tilde{c}^{(t)} = \tanh\left(w_c\left[a^{(t-1)}, x^t\right] + b_c\right) \quad (3)$$

$$c^{(t)} = \Gamma_f^{(t)} o\, c^{(t-1)} + \Gamma_u^{(t)} o\, \tilde{c}^{(t)} \quad (4)$$

The output gate is combined with the current $c^{(t)}$ as shown in Eq. (5)

$$\Gamma_o^t = \sigma\left(w_o\left[a^{(t-1)}, x^t\right] + b_o\right) \quad (5)$$

The output gate is used for the calculation of $a^{(t)}$ shown in Eq. (6)

$$a^{(t)} = \Gamma_o^t\, o\, \tanh\left(c^{(t)}\right) \quad (6)$$

The value of $c^{(t)}$ carries the actual data required whereas $a^{(t)}$ determines which gate is activated and what happens to the value $c^{(t)}$.

VAR is an auto regressive model for multivariate data [7]. Each forecast in VAR is a linear function of its past lags as well as the past lags of all other variables. The VAR model of lag 1 for three variables $x_{t,1}$, $x_{t,2}$, $x_{t,3}$ can be represented as VAR (1), with the Eqs. (7)–(9):

$$x_{t,1} = \alpha_1 + \emptyset_{11}\, x_{t-1,1} + \emptyset_{12}\, x_{t-1,2} + \emptyset_{13}\, x_{t-1,3} + e_{t,1} \quad (7)$$

$$x_{t,2} = \alpha_2 + \emptyset_{21}\, x_{t-1,1} + \emptyset_{22}\, x_{t-1,2} + \emptyset_{23}\, x_{t-1,3} + e_{t,2} \quad (8)$$

$$x_{t,3} = \alpha_3 + \emptyset_{31}\, x_{t-1,1} + \emptyset_{32}\, x_{t-1,2} + \emptyset_{33}\, x_{t-1,3} + e_{t,3} \quad (9)$$

where $\alpha_1, \alpha_2, \alpha_3$ are constants, $\emptyset_{tx}$ are coefficients, and $e_{t,x}$ represent error terms.

Similarly for VAR (2), the lag 2 variables will be added to the above equations. In general, the multiple variables in the equation can be defined as Vectors, hence the name VAR as in Eq. (10).

$$\begin{bmatrix} x_{t,1} \\ x_{t,2} \\ x_{t,3} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} + \begin{bmatrix} \emptyset_{11}\emptyset_{12}\,\emptyset_{13} \\ \emptyset_{21}\emptyset_{22}\,\emptyset_{23} \\ \emptyset_{31}\emptyset_{32}\,\emptyset_{33} \end{bmatrix} \begin{bmatrix} x_{t-1} \\ x_{t-2} \\ x_{t-3} \end{bmatrix} + \begin{bmatrix} e_{t,1} \\ e_{t,1} \\ e_{t,1} \end{bmatrix} \quad (10)$$

The contributions of the proposed work are as follows

i. Statistical validation of the causal relationship between different time-series values of resource usage using Granger Causality tests.
ii. Implementation and evaluation of VAR, LSTM, and a weighted ensemble of VAR+LSTM multivariate forecasting models.

The further sections of the paper are organized as follows: first, we will discuss the method, and implementation, followed by the results, and finally the conclusion.

II. Literature Survey

Any forecasting approach aims to improve upon the accuracy while not overfitting the model. Various univariate and multivariate approaches have been used for forecasting resource usage values; however, a limited study is done on multivariate forecasting. The literature study shows that the multivariate approach can outperform the univariate approach sometimes [4]. This work is an extension of work done in [8], where an ensemble of

univariate forecasting models is used for forecasting the resource usage values in the cloud. This section reviews the previous work related to multivariate forecasting and forecasting using the LSTM model in the domain of virtual machine resource usage values such as CPU, memory, disk, network, and cache usage.

The very first work on multivariate resource usage forecasting is proposed in [4] using Dynamic Linear Model (DLM) and VAR model, the paper proposed to use ensemble models for improved accuracy [4]. The VAR and DLM require the time series to be stationary and memoryless. They are suitable for short-term forecasting. However, the cloud workload exhibits long-range dependencies i.e. the next step value depends on past lags in the series. The work in [9] compared Autoregressive Integrated Moving Average (ARIMA) and LSTM for CPU usage forecasting. The results show that LSTM accuracy outweighs ARIMA model accuracy. But the approach forecasts only CPU usage values instead of multiple other resource usage values. An LSTM and RNN based workload forecasting is proposed in [10] for CPU usage forecasting. The LSTM RNN Model was tested using three benchmark datasets and the empirical results obtained are about the mean squared error of 3.17×10^{-3}. Both the works of Kumar *et al.* [9], Janardhanan and Barrett [10] demonstrated that LSTM is better suited for long-term resource usage forecasting. However, both approaches are univariate but demonstrate the electiveness of LSTM for forecasting.

An ensemble model by combines VAR and LSTM for multivariate forecasting is designed in [11]. The VAR model is used to filter linear interdependencies among the multivariate time series and stacked LSTM is used to capture non-linear trends in residuals obtained from VAR model. This approach is short-term forecasting up to a lag order of 3 only.

An LSTM and Bidirectional LSTM for long-term resource usage forecasting is proposed in [12]. The paper compared LSTM performance with various other state-of-art approaches to find that LSTM yields better accuracy. The approach does not exploit the multivariate feature of the dataset. Tran and Nguyen *et al.* proposed a multivariate fuzzy time series forecasting model using LSTM [13]. To smooth the fluctuations the author proposes a fuzzification technique followed by LSTM Neural network modelling. However the stacked model increases the model complexity and computation requirement. The technique in [14] uses LSTM to identify dependencies among performance metrics like identifying the strongest performance predictors and identifying lagged/temporal dependencies. The author compares LSTM dependency results to Granger causality tests to verify the results. The results indicate that LSTM and Granger test results match. Further using the dependency information, the accuracy of forecasting is improved. The approach proposed in [15] used LSTM for mean host load prediction over consecutive intervals and actual workload multi-step-ahead prediction. The results were tested using two datasets with good accuracy. The approach demonstrates multi-step-ahead prediction using LSTM but not the multivariate characteristic.

It is clear from the analysis of earlier work that multivariate forecasting is a promising methodology for forecasting, and LSTM is a useful tool in that regard. As a result, the main focus of this study is LSTM-based multivariate forecasting.

III. Method

The dataset contains the performance metrics of 1,750 VMs from Bitbrains distributed datacenter [16]. The traces consist of VM performance metrics: CPU cores, CPU capacity provisioned, CPU usage, memory provisioned, memory usage, disk read throughput, disk write throughput, the network received throughput, and network transmitted throughput [16, 17]. The CPU usage, memory usage, disk write throughput, network received and network transmitted metrics are selected for the study. The reason for dropping other metrics is that they did not exhibit any variation in the values, i.e., they were constant values, for example, the number of CPUs provisioned was 2 for all the instances, thus will not contribute significantly to forecasting.

The proposed LSTM model is implemented using the Python Keras framework. A *series_to_supervised* function was used using the Pandas library to transform the dataset such that the values from the future time steps would be appended as outputs to each time step. Since prediction was happening for 11 attributes, for predicting 5-time steps, 55 extra columns would be appended to the dataset and likewise for 15-time steps and 25-time steps. Fig. 2 shows the output of the series to the supervised function.

Input				Output (1 time step)			
CPU usage	CPU usage	Memory c	Memory u	CPU usage	CPU usage	Memory c	Memory u
55.46666	2.133333	2097152	239072.8	32.93333	1.266667	2097152	272628
32.93333	1.266667	2097152	272628	39.86666	1.533333	2097152	247461.6
39.86666	1.533333	2097152	247461.6	32.93333	1.266667	2097152	222295.2
32.93333	1.266667	2097152	222295.2	36.39999	1.4	2097152	276822.4
36.39999	1.4	2097152	276822.4	38.13333	1.466667	2097152	218100.8
38.13333	1.466667	2097152	218100.8				

Figure 2. Output of series to supervised function.

The dataset was normalized to values [0, 1] and redundant columns were dropped from the input. The total 8225 examples were split into 6000 training examples and 2225 test examples. A 2-layer stacked deep LSTM network architecture was chosen with 275 layers each as shown in Fig. 3.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 1, 275)            309100
_________________________________________________________________
dropout_3 (Dropout)          (None, 1, 275)            0
_________________________________________________________________
lstm_4 (LSTM)                (None, 275)               606100
_________________________________________________________________
dropout_4 (Dropout)          (None, 275)               0
_________________________________________________________________
dense_2 (Dense)              (None, 275)               75900
=================================================================
Total params: 991,100
Trainable params: 991,100
Non-trainable params: 0
```

Figure 3. LSTM architecture with 5 inputs and 275 outputs.

The 275 outputs represent 25 time-step outputs for each of the 11 attributes. The model was chosen such that it was symmetric with the same number of cells in each layer, which provided good results.

The LSTM model was also compared with a standard Multilayer Perceptron (MLP) and VAR. The MLP model was chosen similarly to the LSTM architecture, i.e., the number of nodes in the two hidden layers was equal to 275 for the 25-time lag model. VAR took into consideration all 5 attributes for predicting each of the attributes and was implemented using the *statsmodels* library. It performs regression on itself based on the number of lags and variables taken into consideration.

The various forecasting models: multivariate symmetric LSTM, multivariate VAR, Univariate MLP, and Univariate LSTM are implemented and accuracy is measured for out-of-sample forecasts using NRMSE metric given by Eq. (11) and Eq. (12). NRMSE was used because it provides a normalized value so that attributes across different ranges of values can be compared via a common metric.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}{N}} \tag{11}$$

$$NRMSE = \frac{RMSE}{\max(x) - \min(x)} \tag{12}$$

where y_t is the actual value and $\hat{y}_t$ is the forecasted value, min(x) and max(x) are the minimum and maximum values in the dataset.

To further improve the model the research work proposes to implement a weighted ensemble of VAR and LSTM models. Where the weight determination method is: the weight assigned is inversely proportional to the error, i.e., NRMSE value, that is a model with a higher NRMSE value will be assigned less weight and vice versa. Further relative to other models error the weight is derived as follows. Based on the NRMSE values weights are assigned to forecasting models such that sum of all weights is equal to 1.

$$w_1 + w_2 + \cdots . + w_k = 1 \tag{13}$$

$$w_i \ \alpha \ \frac{1}{\sum_{j=0}^{k} w_j} \tag{14}$$

The weight assigned w_i is inversely proportional to the error NRMSE_i

$$w_i \ \alpha \ \frac{1}{\mathrm{NRMSE}_i} \tag{15}$$

From Eq. (14) and Eq. (15)

$$w_i = \frac{1}{\mathrm{NRMSE}_i \sum_{j=0}^{k} w_j} \tag{16}$$

From Eq. (15) substituting for w_j

$$w_i = \frac{1}{\mathrm{NRMSE}_i \sum_{j=0}^{k} \frac{1}{\mathrm{NRMSE}_j}} \tag{17}$$

where

$$\sum_{i=1}^{k} w_i = 1$$

After assigning the weights we combine the forecasts to generate the actual forecast as follows:

$$\widehat{y_f} = w_1 y_f^1 + w_2 y_f^2 + \cdots . + w_k y_f^k \tag{18}$$

where $f = 1, 2, \ldots, T$ number of forecasts from different models.

IV. Results and Discussion

This section discusses the implementation results, first, the Granger causality test results are discussed followed by accuracy values obtained of univariate and multivariate forecasting models are compared and inferred.

The Granger causality tests were performed using python statsmodels library. The Granger causality test between two-time series helps determine if one series can be used to predict the other [18]. If a time series X-Granger-Causes-Y then past values of X help predict the value of Y above and beyond the information contained in past values of Y. Table I shows the Granger causality test results.

TABLE I. Granger Causality Test Results

Granger causality	5 Lag	15 Lag	25 Lag
CPU usage →Disk write	$p < 0.005$	$p < 0.005$	$p < 0.005$
Disk write → CPU usage			
CPU usage →Memory usage			
Memory usage → CPU usage			
Memory usage → Disk write			
Disk write →Memory usage			

As the P value for all series is $p < 0.005$, it is interpreted that each time series contributes to the prediction of other time series values. Thus, multivariate forecasting takes into account the hidden interaction among the features, unlike univariate forecasting.

The various forecasting models: MLP, Univariate LSTM, multivariate symmetric stacked LSTM, multivariate VAR, and the weighted ensemble of VAR and LSTM models are implemented. The accuracy of these models is measured for out-of-sample forecasts using the NRMSE metric given by Eq. (11) and Eq. (12). NRMSE was used because it provides a normalized value so that attributes across different ranges of values can be compared via a common metric. Table II shows the NMRSE values for CPU usage, from Table II, it can be observed that the proposed algorithm has an acceptable error and better accuracy for different lag values of 5, 15, and 25.

Further observation of the results in Table II can be inferred that multivariate stacked LSTM in comparison with univariate LSTM has a lesser NRMSE value in a range of 3–90% for various resources across different lag values, and thus multivariate approach provides better accuracy compared to univariate approach. Further, the multivariate stacked LSTM in comparison with VAR and the proposed ensemble forecasting has a lesser NRMSE value in a range of 1–5% for various resources across different lag values, and thus multivariate stacked LSTM approach provides better accuracy compared to VAR the proposed ensemble forecasting.

The MLP model does decently well in predicting values, but for the most part, it is not as accurate as standard approaches used for time series forecasting. The Univariate LSTM approach takes only the variable, which is being predicted, into consideration for training, not

accounting for other variables, and again falls short of Multivariate parts. Multivariate LSTM can capture long-term dependencies while predicting outputs and also accounts for all the variables while doing so. VAR also does reasonably well on the dataset due to the fact the selected attributes are stationary and repeated every few time steps. The weighted ensemble VAR+LSTM model does not show much variation as only two models, VAR and LSTM are used. The ensemble approaches perform better when there are multiple heterogeneous base predictors.

TABLE II. OUT-OF-SAMPLE FORECAST NRMSE VALUES

Model Type	Univariate MLP	Univariate LSTM	Multivariate stacked-LSTM	Multivariate VAR	Multivariate Weighted Ensemble VAR + LSTM
5 lags					
Network received	2.806	0.994	**0.395**	0.537	0.435
Network transmitted	6.693	6.494	6.382	6.422	**6.365**
CPU usage	1.543	1.499	**0.851**	0.916	0.871
Disk write throughput	2.293	0.774	0.566	**0.537**	0.545
15 lags					
Network received	2.816	1.276	**0.386**	0.537	0.422
Network transmitted	5.674	6.226	6.396	6.422	**6.361**
CPU usage	2.423	0.853	**0.822**	0.915	0.846
Disk write throughput	2.852	1.077	0.539	0.537	**0.536**
25 lags					
Network received	0.454	0.592	0.415	0.537	**0.365**
Network transmitted	6.579	6.423	**6.309**	6.422	6.352
CPU usage	1.045	1.688	**0.846**	0.916	0.869
Disk write throughput	1.256	1.314	0.556	**0.537**	0.539

Figs. 4–7 show the graph of the out-of-sample forecast of the proposed symmetric LSTM model for various resource types.

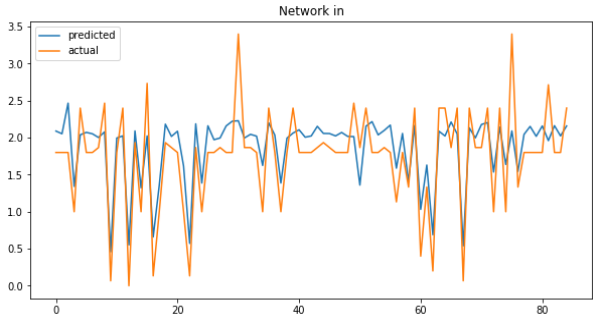

Figure 4. Network received throughput forecast.

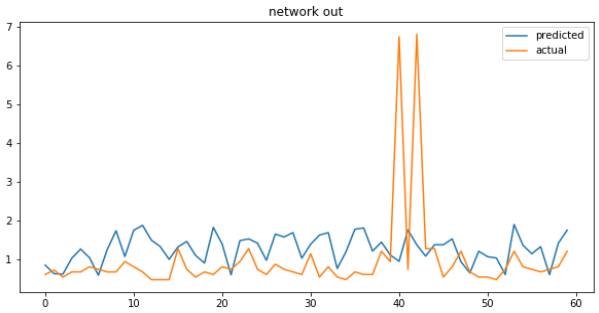

Figure 5. Network transmitted throughput forecast.

Fig. 4 compares the actual and predicted network received throughput values in bytes/second. It is evident from the graph that the model not only predicts the trend but is able to forecast with good accuracy. Fig. 5 compares the actual and predicted network transmitted throughput values in bytes/second, the model predicts the trend accurately but there is a variation in the predicted values resulting in NRMSE value of 6.309 (>1). Fig. 6 compares the actual and predicted disk write throughput values in MegaBytes/second, the model is able to predict the trend and forecast the values accurately with NRMSE value 0.556. Fig. 7 compares the actual and predicted CPU usage in percentage utilization, the graph shows that the model is able to predict the trend and forecast values with NRMSE value of 0.846. Thus, it is evident that multivariate stacked LSTM model is able to predict the trend and forecast usage values for multiple resources accurately.

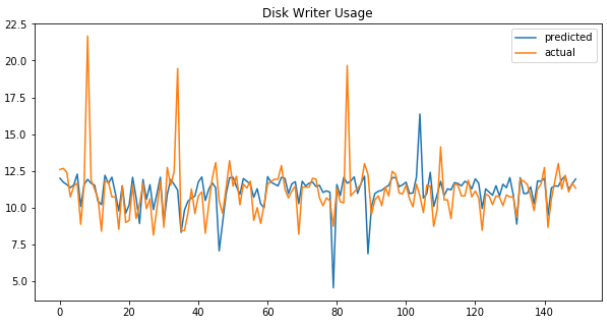

Figure 6. Disk write throughput forecast.

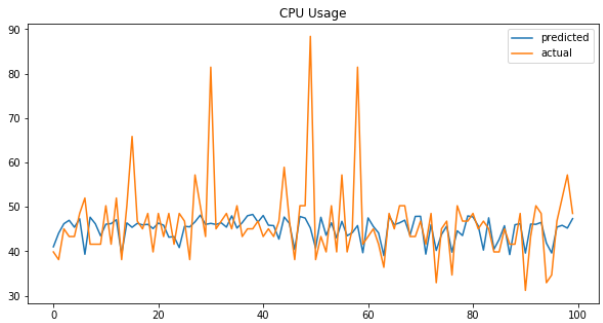

Figure 7. CPU usage forecast.

V. Conclusion

This research proposed the application of a symmetric LSTM model for forecasting resource usage values focusing on multivariate monitoring data. The knowledge of future resource usage information helps in cloud service and performance management, which is important from a cloud service provider perspective. The production dataset from GWA-T-12 Bitbrains is used in this research. At first, the causality relationships among various resource usage metrics are demonstrated using Granger causality tests. Then the LSTM model is then configured for long- and short-term forecasting, the results show that accuracy is better compared to other univariate and multivariate approaches. Furthermore, multivariate forecasting using LSTM, VAR, and weighted ensemble VAR+LSTM model is designed and implemented, and the result shows that the multivariate model using LSTM is able to forecast with better accuracy compared to other multivariate and univariate approaches. Although the proposed model is forecasting the trend of data very well however the variation in data is not captured well. The future work is to improvise the model to capture the variations in the data by tuning the model/parameters further.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Jyoti Shetty proposed the problem, carried out the study, designed the methodology with guidance from Shobha G. Karthik Cottur worked on implementing the stacked LSTM, MLP and VAR models. Prajwal Y implemented the weighted LSTM model. Jyoti Shetty further drafted the paper and the results. All authors had approved the final version of the paper.

Acknowledgment

The authors wish to thank RV College of Engineering for its support and encouragement during the research.

References

[1] G. Cicotti, L. Coppolino, S. D'Antonio, and L. Romano, "Big data analytics for QoS prediction through probabilistic model checking," arXiv:1405.0327, 2014. http://arxiv.org/abs/1405.0327

[2] Amazon Elastic Container Service. Developer Guide. [Online]. Available: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-configure-auto-scaling.html

[3] J. Shetty, B. S. Babu, and G. Shobha, "Proactive cloud service assurance framework for fault remediation in cloud environment," *International Journal of Electrical & Computer Engineering*, vol. 10, no. 1, pp. 987–996, 2020.

[4] J. S. Hirwa and J. Cao, "An ensemble multivariate model for resource performance prediction in the cloud," in *Proc. IFIP International Conference on Network and Parallel Computing*, *Lecture Notes in Computer Science*, Springer, 2014

[5] A. K. Nayak, K. C. Sharma, R. Bhakar, and H. Tiwari, "Short-term Wind Speed Forecasting Using Multi-Source Multivariate RNN-LSTMs," in *Proc. 2021 9th IEEE International Conference on Power Systems (ICPS), Kharagpur, India*, 2021, pp. 1–6. Doi: 10.1109/ICPS52420.2021.9670251

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Journal of Neural Computation*, vol. 9, issue 8, pp. 1735–1780, 1997.

[7] J. H. Stock and M. W. Watson, "Vector autoregressions," *Journal of Economic Perspectives*, vol. 15, no. 4, pp. 101–115, 2001.

[8] J. Shetty and G. Shobha, "An ensemble of automatic algorithms for forecasting resource utilization in cloud," in *Proc. 2016 Future Technologies Conference (FTC)*, 2016, pp. 301–306.

[9] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, vol. 125, pp. 676–682, 2018.

[10] D. Janardhanan and E. Barrett, "CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models," in *Proc. 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, Cambridge, 2017, pp. 55–60.

[11] S. Ouhame and Y. Hadi, "Multivariate workload prediction using vector autoregressive and stacked LSTM models," in *Proc. the New Challenges in Data Sciences: Acts of the Second Conference of the Moroccan Classification Society (SMC'19)*, ACM, New York, NY, USA, 2019.

[12] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *Proc. 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Bhubaneswar, 2017, pp. 1–6.

[13] 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.

[14] S. Y. Shah, Z. Yuan, S. Lu, and P. Zerfos, "Dependency analysis of cloud applications for performance monitoring using recurrent neural networks," in *Proc. 2017 IEEE International Conference on Big Data*, Boston, MA, 2017, pp. 1534–1543.

[15] B. Song, Y. Yu, Y. Zhou, *et al.*, "Host load prediction with long short-term memory in cloud computing," *Journal of Supercomputing*, vol. 74, pp. 6554–6568, 2018.

[16] The Grid Workloads Archive. GWA-T-12 Bitbrains. [Online]. Available: http://gwa.ewi.tudelft.nl/datasets

[17] S. Shen, V. Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *Proc. the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2015.

[18] J. M. McCracken, "Exploratory causal analysis with time series data," in *Exploratory Causal Analysis with Time Series Data*, Morgan & Claypool, 2016.

Jyoti Shetty is an assistant professor of Computer Science and Engineering Department, RV College of Engineering, Bengaluru, India. She has 16 years of teaching and 2 years of industry experience. Her specialization includes data mining, machine learning and cloud computing. She has published research papers in reputed journals and conferences. She has also executed sponsored projects funded by various agencies nationally and internationally. She was the recipient of awards such as the SAP Award of excellence from IIT Bombay for demonstrating ICT in education in 2016 and the HPCC Systems Mentor Badge Award in 2021 for providing guidance and direction towards the successful completion of intern open source projects.

G. Shobha is a professor of Computer Science, and Engineering Department, R.V College of Engineering, Bengaluru, India. She has teaching experience of 28 years. Her specialization includes data mining, machine learning, and image processing. She has published more than 150 papers in reputed journals/conferences. She has also executed sponsored projects worth INR 200 lakhs funded by various agencies nationally and internationally. She is a recipient of various awards such as the Career Award for young teachers 2007-08 constituted by the All India Council of Technical Education, Best Researcher award from Cognizant 2017, GHC Faculty Scholar for Women in Computing in 2018, IBM Shared University Research Award in 2019, HPCC Systems community recognition award 2020.

Karthik Cottur is a computer science student who graduated from RVCE in 2020. He is interested in building cool automation using basic coding and some machine learning.
He is currently working at a data pipeline startup called Fivetran.