Received August 22, 2021, accepted September 3, 2021, date of publication September 17, 2021, date of current version September 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3113714

BHyPreC: A Novel Bi-LSTM Based Hybrid Recurrent Neural Network Model to Predict the CPU Workload of Cloud Virtual Machine

MD. EBTIDAUL KARIM[1], MIRZA MOHD SHAHRIAR MASWOOD[1], (Member, IEEE), SUNANDA DAS[2], AND ABDULLAH G. ALHARBI[3] (Member, IEEE)
[1]Department of Electronics and Communication Engineering, Khulna University of Engineering & Technology, Khulna 9203, Bangladesh
[2]Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna 9203, Bangladesh
[3]Department of Electrical Engineering, Faculty of Engineering, Jouf University, Sakaka 72388, Saudi Arabia

Corresponding author: Mirza Mohd Shahriar Maswood (shahriar@ece.kuet.ac.bd)

ABSTRACT With the advancement of cloud computing technologies, there is an ever-increasing demand for the maximum utilization of cloud resources. It increases the computing power consumption of the cloud's systems. Consolidation of cloud's Virtual Machines (VMs) provides a pragmatic approach to reduce the energy consumption of cloud Data Centers (DC). Effective VM consolidation and VM migration without breaching Service Level Agreement (SLA) can be attained by taking proactive decisions based on cloud's future workload prediction. Effective task scheduling, another major issue of cloud computing also relies on accurate forecasting of resource usage. Cloud workload traces exhibit both periodic and non-periodic patterns with the sudden peak of load. As a result, it is very challenging for the prediction models to precisely forecast future workload. This prompted us to propose a hybrid Recurrent Neural Network (RNN) based prediction model named BHyPreC. BHyPreC architecture includes Bidirectional Long Short-Term Memory (Bi-LSTM) on top of the stacked Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Here, BHyPreC is used to predict future CPU usage workload of cloud's VM. Our proposed model enhances the non-linear data analysis capability of Bi-LSTM, LSTM, and GRU models separately and demonstrates better accuracy compared to other statistical models. The effect of variation of historical window size and training-testing data size on these models is observed. The experimental result shows that our model gives higher accuracy and performs better in comparison to Autoregressive Integrated Moving Average (ARIMA), LSTM, GRU, and Bi-LSTM model for both short-term ahead and long-term ahead prediction.

INDEX TERMS Cloud computing, deep learning, recurrent neural network, time series analysis, virtual machine, workload prediction.

I. INTRODUCTION

With the introduction of a new wave of applications and services via the internet, a new dawn of information explosion is taken place. As a result, there is a huge surge in the requirement for data storage and data processing. Cloud computing brings forth the on-demand access of the network to a collective pool of configurable computing assets for profuse computing services (e.g., data storage, data processing) [1]. In this modern era of information technology, cloud computing acts as the framework for on-demand computing services. It avails us of a pay-as-you-go pricing architecture. The elasticity of resources is achieved by scalable virtualization of the cloud. Cloud provides us three different services based on three delivery structures. These are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [2].

Owing to its high computational power and network access, there are widespread use of cloud computing in countless industries, health care facilities [3], financial service corporations [4] and academic fields. As a result, cloud data center assets are expanding to meet the demand, which is subsequently increasing the cost of enterprise investment in data centers. A data center can become inefficient due to low resource utilization. For proper utilization of data center resources, we need to balance our loads. An immense

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Tong.

improvement in utilizing cloud resources and execution of tasks can be attained by the dint of precise forecasting [5]. Effective prediction of workload holds the key in enabling a system to take proactive and dynamic decisions, instead of taking reactive decisions. Consolidation of Virtual Machines (VMs) can be accomplished by merging and reallocating VMs into a tiny quantity of active physical servers. Another key hallmark of the cloud is the ability to relocate VM to a distinct hardware environment. This feature is called VM migration [6]. VM consolidation leads to a reduction in energy consumption by turning Physical Machines (PMs) having no VM to sleep state. Here, energy reduction occurs as PM in the sleep state consumes remarkably less energy compared to PM in active state [7]. Energy awareness tools are necessary to improve decision-making capability at both PM and VM levels. This can be done by accurately forecasting the workload pattern at VM level [8]. To circumvent the breach of Service Level Agreement (SLA), there is a necessity of precise prediction of PM usage in both VM consolidation and VM migration [9], [10]. Prediction of the trend of workload change also enables us to handle task scheduling proactively. Proactive scheduling by forecasting the load eventually assists in load balancing of cloud resources [11]. Hence, A robust future workload prediction in cloud computing architecture plays a crucial role in overall efficient resource assignment, reduction of energy consumption, load balancing, and task scheduling without breaching the SLA [12]–[14].

Prediction of workload is one type of time series forecasting. It has always been a tough challenge to provide precise future workload predictions in Cloud and Grid systems, with various time series data having different patterns and abrupt peaks at times. It is very likely that an algorithm can produce different under-and over-predictions within a single time series. In consequence, it becomes difficult to design a method using the results of such prediction [5]. Existing practices of prediction involve conventional methods of prediction using statistical models and neural networks. Nevertheless, there are several deficiencies in all methods while interacting with time series forecasting. The repetitive workload pattern of cloud resources with sudden peak [15] plays the key role in making accurate forecasting.

Among the various models which deal with time series forecasting, one common model is the statistical model. Statistical forecasting includes the usage of statistics to predict what will occur in the future, hinge on historical evidence. For intricate and non-linear time series forecasting, statistical approaches are inefficient and unreliable. Autoregression (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA), Simple Exponential Smoothing (SES), Holt-Winters Exponential Smoothing (HWES) and others are the example of statistics-based classical time series forecasting methods. Additionally, a neural network can also be used to predict the workload. The neural network model performs better in handling the complex observations compared to the statistical methods [16].

Uneven temporal structures, incomplete values, excessive noise, and complicated interrelationships across several variables characterize real-world time series data. Moreover, the trends and seasonality present in the temporal structure of the data make the prediction task more difficult. Deep learning techniques are equipped to deal with these problems as they can automatically learn and elicit features from raw and faulty data. As a result, the importance of deep learning techniques for the purpose of time series data forecasting, like CPU load in a cloud data center is immense. Recurrent Neural Network (RNN), a frequently used deep learning technique performs better than classical time series prediction models, as RNNs are able to process a series of inputs in deep learning and can retain their state until the next series of inputs are processed. Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM) are commonly used RNN for this purpose. Due to its ingenuity to process non-linear data, RNN based models are used in various time series forecasting applications, e.g., cloud workload prediction, stock prediction [17]–[19], and weather forecasting [20] etc.

An efficient combination of several methods can give rise to hybrid models. In this paper, a novel hybrid model is presented to forecast a vital cloud data center resource usage namely, Central Processing Unit (CPU) usage in VMs. The major contribution of our work in this paper is stated below:

- We propose a novel hybrid model namely BHyPreC, which combines a 1D convolution layer, Bi-directional Long Short Term Memory (Bi-LSTM), LSTM, and GRU with the best hyper-parameter settings to accurately predict multi-step-ahead workload. The ability of our model to efficiently extract features and to use both past and future data to learn the nature of workload patterns along with a combination of well-designed stacked LSTM and GRU units; gives us an upper hand in dealing with the nonlinear and complex data patterns.
- We apply a combined grid search technique of historical window size and train:test ratio to tune the model properly and to obtain the best possible set of window size and train:test ratio size. This set is responsible for generating the least error while predicting the CPU usage in VMs.
- We implement an overlapping sliding window approach technique across the entire training data. As a result, data is split into segments of equal lengths. Segmented data are then fed into the 1D-Convolution layer of our model. This helps in increasing the long-term prediction accuracy.
- We evaluate our model for four different evaluation metrics. The experimental result affirms the superiority of our model in comparison to traditional ARIMA and three other prevalent RNN based models trained with the same dataset.
- For analyzing each of these models, we carry out Cumulative Distribution Function (CDF), box plot, prediction

length, and error percentage rise analysis of each evaluated metric separately. It is done through a series of systematic analyses. For each analysis, the least value is obtained through training the model with an optimum set of grid searched window size and training:testing ratio.

We have arranged the rest of the paper in this following fashion. Related works are discussed in section II. Section III exhibits the complete architecture of our proposed model. Section IV demonstrates the analysis and metrics of the experimental results used for appraisal. In section V, we conclude and summarize the whole paper along with future research scopes.

II. RELATED WORKS

For predicting the workload in the cloud, several statistical, neural network-based, and deep learning-based methods have been proposed earlier. Time-based correlation within a group of VMs is used to predict changing workload patterns by using the Hidden Markov Model in [15]. In [21] we find that it is possible to use Markov chain analysis to predict how a broader system will respond when quality service assurances are not fulfilled. The expense of using the Markov chain is much less compared to that of test-beds and the computational costs are not influenced by the size of the modeling system. The Revised Simple Exponential Smoothing (RSES) method is generated by taking a new notion of shift arising probability into the context of the Simple Exponential Smoothing (SES) method in [22]. Here, weight coefficients of the forecasting system are assessed by using shift arising probability. The RSES method helps one to forecast time series with abrupt level changes more precisely. An autonomic three-dimensional resource provisioning procedure, equipped with resource-aware, SLA-aware, and user behavior-aware attributes has been proposed in [23]. In an attempt to grant both providence and flexibility, the suggested technique employed a Radial Basis Function (RBF) neural network. This method gives more attention to web user behavior in comparison to the Two-Phase Alarm (TPA), Double Exponential Smoothing (DES), and Latency-based One alarm (LAT-1Al) approach. To administer the elasticity attribute in the cloud environment, an elastic controller is designed in [24] that employs Colored Petri Nets (CPNs). The proposed model uses CPNs to provide a comprehensive approach for simulating arrival, controller, and host transitions.

A pragmatic approach to complex resource provisioning for SaaS applications based on ARIMA model prediction is introduced in [25]. The precision of the future workload prediction is appraised by utilizing actual traces of Wikimedia Foundation web server requests. Accuracy of prediction also contributes to the efficient usage of resources. In [26], CloudInsight, a complex and highly variable cloud workload prediction system for online workload prediction is introduced. It incorporates a handful of local predictors and by interactively evaluating the appropriate weights of each local predictor, generates an ensemble prediction model for them. A Hierarchical Pythagorean Fuzzy Deep Neural Network (HPFDNN) is proposed in [27] to estimate the number of cloud resources available. Here, neural representation is used as a supplementary approach for clear interpretations of original results, beyond the use of fuzzy logic. Consumers can determine the number of cloud services to buy on the ground of the expectancy of a deep neural network. A prediction approach that is self-adaptive using the Ensemble model and Subtractive-Fuzzy Clustering-based Fuzzy Neural Network (ESFCFNN) is suggested in [28]. For proactive resource provisioning in a cloud environment, a self-learning fuzzy approach is implemented in [29]. To investigate the appropriate auto-scaling decisions, a decision-maker framework is proposed which uses the self-learning fuzzy technique. Here, we can see that the proposed approach can effectively allocate resources in comparison to the dynamic resource provisioning approach and a neural network-based proactive proposal. In [30], a hybrid resource apportion method based on the amalgamation of the automatic computing concept and the Fuzzy Analytical Hierarchy Process (FAHP) approach is proposed. It outshines other state-of-the-art resource provisioning approaches concerning virtual machine allocations, response time, and cost.

To address the resource provisioning issue, a hybrid approach is provided in [31], which clusters the workload submitted by end-users using the Imperialist Competition Algorithm (ICA) and K-means clustering. To identify scaling decisions for effective resource provisioning, a decision tree method is employed here. It performs better in comparison to K-means Pattern and Log-fuzzy methods. For workload estimation, an adaptive approach is introduced in [32], where workloads are grouped into various sections that are automatically allocated as per workload attributes for various prediction models. It performs better in comparison to ARIMA, Support Vector Machines (SVMs), and Linear Regression (LR). Performance comparison among three machine learning techniques such as ARIMA, Multi-Layer Perceptron (MLP), and GRU is studied in [33]. It shows that GRU performs better among the compared techniques. To construct an emerging framework to anticipate the VM workload, [34] introduces a modified Elastic Adaptive Neural Network (EANN), equipped with modified adaptive smoothing errors. Recurrent neural network with Back Propagation Through Time (BPTT) algorithm has better capability to precisely estimate host CPU usage in [35]. It also demonstrates how far into the future can an RNN model predict the workload accurately.

To transfer the dynamic workload into a cloud server, a Learning Automata (LA) based offloading approach is proposed in [36]. It employs an LSTM model to forecast future workload and a Reinforcement Learning (RL) approach to determine optimal scaling. Numerous models are stacked based on RNN and autoencoder in [5]. RNN is trained here to merge various prediction outcomes into one, whereas the autoencoder is used to construct an improved input representation characteristics layer created from the RNN predictions of the components. In [37], LSTM is introduced to predict data center machines CPU utilization and it is compared with an established conventional method named ARIMA. Attention-based encoder-decoder LSTM network is used in [38] to predict workload in a cloud environment. To increase the accuracy of forecasting, long prediction sequences are divided into smaller sequences by using the scroll prediction method. In [39], an LSTM-RNN based model is introduced to predict the cloud data center workload and it shows better prediction accuracy compared to the black hole and backpropagation method.

For predicting the workload in PM, a GRU-based model is introduced in [40]. In [41] a new model with an Encoder-Decoder network based on GRU (GRUED) is presented which includes two gated RNNs (GRUs) that function as a GRU encoder and GRU decoder duo. It can perform more precise multi-step-ahead host load prediction, compared to other RNN models. GRU-ES, which integrates the GRU model and the Exponential Smoothing (ES) system, is a hybrid approach for forecasting resource usage is presented in [42]. For both single-step prediction and multi-step prediction, the proposed hybrid approach outshines state-of-the-art techniques. A hybrid LSRU model which combines both LSTM and GRU along with a 1D convolution layer at the apex is proposed in [43]. It can forecast the use of CPU, disk, memory, and bandwidth of the cloud data center not just for short-term estimation, as well as for long-term estimation with sudden sharp peak prediction. In [44] an ensemble strategy, namely ESNemble is proposed to use the Echo State Network (ESN) to draw out and merge features obtained from different prediction algorithms to enhance the overall accuracy and efficiency. It is organized into four key stages. First, to pick features from various prediction algorithms, it uses nonlinear ESN reservoirs. Secondly, to avoid overfitting, the sizes of these extracted features are reduced. Third, the derived and reduced features are consolidated into a single matrix from the combined prediction algorithms. Finally, regression is applied from the consolidated matrix to produce ESNemble's final predictions. Based on Generative Adversarial Networks (GAN), a time series forecasting model known as Adversarial Sparse Transformer (AST) is proposed in [45]. To grasp the sparse attention map for time series forecasting, it incorporates a space transformer as a generator. A discriminator is also utilized alongside the generator to enhance the prediction accuracy at a sequence level. The authors used covariates along with the targeted univariate data to train their model. This model also addressed the error accumulation problem generated while forecasting.

In summary, the key motive behind the above-mentioned works is to reduce the error rate while predicting future cloud workload in single-step and multi-step ahead. Accurate prediction is necessary to assign resources more precisely when VM consolidation and migration is done. It can also help us to minimize the consumption of energy and handle task scheduling without violating SLA. For the above-mentioned reasons, we propose a hybrid method namely BHyPreC, which consists of different RNN methods to gain better precision in prediction by significantly reducing the prediction error rate. Our model can also predict more precisely the workload multi-step ahead in the future in contrast to other prevailing state-of-the-art statistical and RNN architectures. The comparative analysis of the related works in this field discussed so far, along with our proposed approach has been summarized in Table 1. This table analyzes each work based on six characteristics: (1) Utilized technique, (2) Performance Metrics, (3) Workload, (4) Prediction Window, (5) Preprocessing, and (6) Simulator.

III. OUR PROPOSED HYBRID RNN MODEL

This section illustrates and describes the entire procedure adopted in this paper. It is subdivided into seven sub-sections. Sub-section III-A summarizes our proposed model: BHyPreC, along with its visualization. The overview of the core concept of our model: neural network and recurrent neural network is stated in sub-section III-B and sub-section III-C, respectively. Then, a short introduction of the three key RNN units presented in our model namely LSTM, GRU, and Bi-LSTM is described in sub-section III-D, III-E, and III-F, respectively. There is also a key discussion of the training time complexity of our model in sub-section III-G.

A. PROPOSED ARCHITECTURE

The design overview of our proposed model is shown in Fig. 1. Here initially, input data are pre-processed using the 'MinMaxScalar' function. This function allows us to normalize the input data in the range of 0 to 1. The normalized data is then fed into a 1D-Convolutional Neural Network (CNN) layer by implementing a sliding window approach. A 1D CNN is extremely useful for extracting features from a fixed-length section of a larger dataset, where the locale of the feature in the section is less relevant. It is useful in time series analysis. This 1D CNN layer contains 64 output filters, kernel size of 5, stride size of 1, and activation function 'relu'.

The output of this CNN layer is then fed into a Bi-LSTM layer of 128 hidden memory units combining both forward and backward propagation. The output of Bi-LSTM is then passed through 2 successive LSTM layers and a GRU layer. Each of these layers contains 64 hidden memory units. Then, the output will be passed into an LSTM layer containing 50 hidden memory units. Finally, the output will be propagated through a fully connected neural network containing 1 layer. Here, Bi-LSTM uses both forward and backward information to update the hidden state in two-way directions. Again, both LSTM and GRU are two special variants of RNN having fewer parameters in comparison to Bi-LSTM.

Although, LSTM usually performs better in larger datasets and GRU outshines others in smaller datasets; a well-balanced combination of both the RNN units with better-tuned hyper-parameters yields a better result. The input time series data is split into successive history sequences,

TABLE 1. Summary of related works.

Ref.	Utilized Techniques	Evaluation Metric	Workload	Prediction Window	Preprocessing	Simulator
[5]	RNN+ Autoencoder	MAE, MAPE, RMSE, NRMSE	NASA HTTP traces, SHARCNet, WorldCup98	Multi step	Yes	Ubuntu 14.04.3 LTS, Graphic Processor
[15]	Co-clustering + Hidden Markov Model	Percentages of correct, under and over-prediction	Production Cloud Environment	Single step	No	X
[22]	RSES	MSE	Normally distributed stochastic data	Single step	No	X
[23]	RBF neural network	Response time, Cost, SLA Violation	NASA traces	N/A	No	CloudSim
[24]	CPN	CPU utilization, response time	Google cluster workload, Yahoo cluster workload, Wikipedia workload	N/A	No	CPN tools + CloudSim
[25]	ARIMA	RMSD, NRMSD, MAD, MAPE	Wikimedia Foundation	Multi Step	No	CloudSim
[26]	CloudInsight: A Council of Expert	NRMSE	Cluster Workloads, Web workloads, HPC workloads	Multi Step	No	Python 2.7 on Ubuntu 16.04 LTS
[27]	HPFDNN	Accuracy rate, Total Cost	OpenCloud cluster	Single Step	Yes	X
[28]	ESFCFNN	MSE, MAE, Prediction Error Index	Historical database	Multi step	Yes	X
[29]	MLE and LLR + Self learning fuzzy	MSE, Response time, Provisioned VM, Cost	Synthetic, RuneScape	N/A	Yes	CloudSim
[30]	FAHP	Response time, Cost, Number of VM, MSE, NMSE, MeAD, and R^2	ClarkNet, NASA, and synthetic workload traces	Single step	Yes	CloudSim
[31]	ICA/K-means clustering + Decision tree	Response time, cost, CPU utilization, elasticity	FIFA and ClarkNet workloads	N/A	Yes	CloudSim
[32]	Adaptive classified prediction scheme	Accuracy, Cumulative relative error	Google Cluster trace	Multi step	Yes	X
[33]	Comparison of ARIMA, MLP, and GRU	RMSE	NASA HTTP traces	Multi Step	Yes	Google Colaboratory
[34]	EANN	MSE, Load balance	Generated Workload	Single step	No	CloudSim
[35]	RNN	MAE, MSE	PlanetLab	Multi step	No	CloudSim
[36]	LA + RL + LSTM	Energy consumption, execution time, and CPU utilization, MAE, RMSE	Chicago open data portal, two synthetic workloads	Single Step	Yes	iFogSim
[37]	LSTM	RMSE	Google's cluster dataset	Multi Step	Yes	Google Cloud Platform
[38]	Attention Based LSTM Encoder-Decoder	MAE, RMSE, MAPE, RMSSE	Alibaba and Dinda workload traces	Multi step	Yes	X
[39]	LSTM + RNN	MSE	NASA HTTP traces, Calgary server, and Saskatchewan server	Multi step	Yes	Python on Intel Core I5-M520 processor
[40]	GRU	MSE, MAPE	Web server cluster	Multi step	Yes	Linux Centos 7.2
[41]	GRUED	MAE, MAPE, RMSE, RMSSE	Google clusters, Dinda workload traces	Multi step	Yes	UNIX system
[42]	GRU-ES	MSE, RMSE, MAPE	Google Cluster Trace	Multi step	Yes	Python on server, Intel Core i7-5930k processor
[43]	LSRU	MSE, MAE, RMSE, MAPE	Bitbrains traces	Multi step	Yes	Kaggle
[44]	ESNemble	MAE, MAPE, RMSE	CRAN dataset, EDGAR data set, Kyoto dataset	Multi step	No	X
[45]	AST	Normalized quantile loss	Electricity, traffic, wind, solar and M4-hourly datasets	Multi step	Yes	X
Proposed	**BHyPreC**	**MSE, RMSE, MAE, MAPE**	**Bitbrains traces**	**Multi step**	**Yes**	**Google Colaboratory**

Here, 'N/A' indicates 'Not Applicable'
'X' indicates 'Information not available'
Abbreviations: RMSD: Root Mean Square Deviation, NRMSD: Normalized Root Mean Square Deviation, MAD: Mean Absolute Deviation, NRMSE: Normalized Root Mean Square Error, HPC: High Performance Computing, MLE: Maximum Likelihood Estimation, LLR: Local Linear Regression, NMSE: Normalized Mean Square Error, MeAD: Median Absolute Deviation, RMSSE: Root Mean Segment Square Error.

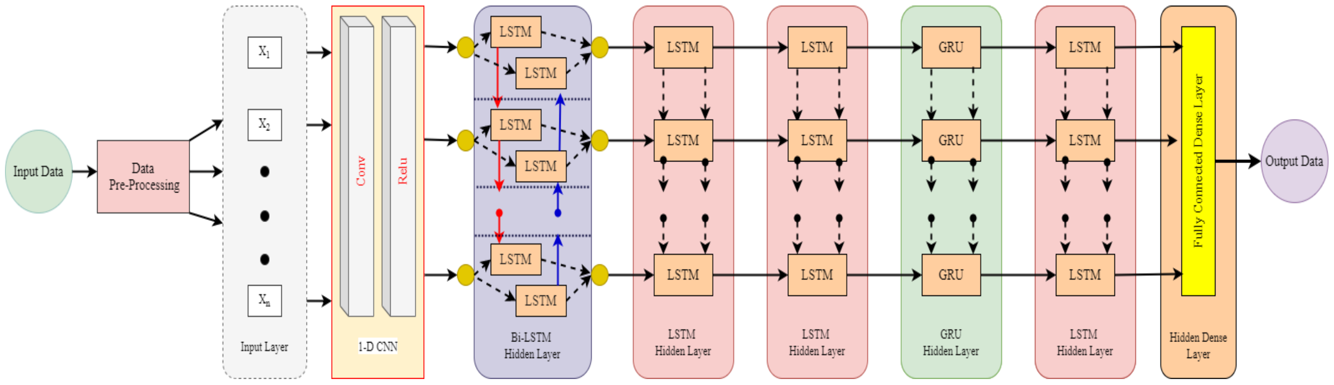


FIGURE 1. The proposed hybrid RNN model.

consisting of fixed size and each of the sequences is followed by a prediction sequence of the defined size. The proposed architecture utilizes the history and forecasting sequences as inputs and supervised outputs/labels, respectively.

The Summary of our proposed BhyPreC model is shown in Table 2. Here, 'HWS' indicates the History Window Size. For this experiment, we take HWS of 30, 60, 90, and 120 successively in a grid search approach. Here, the 'Dense' layer represents the fully connected neural network. There is a total of 196,875 trainable parameters in our BHyPreC model. Table 3 shows the hyper-parameters utilized in our proposed BHyPreC model. The batch size, convolution filter size, kernel size, and output neuron layer of each RNN unit used in our model are tuned through a rigorous trial and error method approach. The learning rate, β_1, β_2, and ϵ of the Adam optimizer are set according to [38]. We iterate each of our models 100 times.

TABLE 2. Summary of the proposed BhyPreC model.

Layers	Output Shape	Parameters
1D-Convolution	(None, HWS, 64)	384
Bi-LSTM	(None, HWS, 128)	66048
LSTM	(None, HWS, 64)	49408
LSTM	(None, HWS, 64)	33024
GRU	(None, HWS, 64)	24960
LSTM	(None, 50)	23000
Dense	(None, 1)	51
Total Trainable Parameters		196,875

TABLE 3. Hyper-parameters of the proposed BHyPreC model.

Hyper-parameters	Values
History Window Size (HWS)	{30,60.90,120}
Batch Size	64
Hidden Bi-LSTM Layer Neurons	128
Hidden 1st LSTM Layer Neurons	64
Hidden 2nd LSTM Layer Neurons	64
Hidden GRU Layer Neurons	64
Hidden Final LSTM Layer Neurons	50
Convolution Filters	64
Kernel Size	5
Learning Rate	0.001
1st Moment Exponential Decay Rate, β_1	0.9
2nd Moment Exponential Decay Rate, β_2	0.999
Numerical Stability, ϵ	10^{-7}
Iteration	100

B. NEURAL NETWORK

Inclined by the biological neural network, a neural network or commonly known as Artificial Neural Network (ANN) is an array of algorithms that are designed to detect hidden relationships in a collection of data. It can adjust to evolving data, resulting in the best possible outcome without requiring the output requirements to be redesigned. Each node in ANN is known as a neuron, forms the basic backbone of the neural network. Fig. 2 exhibits a simplified view of a neural network with two input nodes, one output node, and two underlying hidden layers in between them.

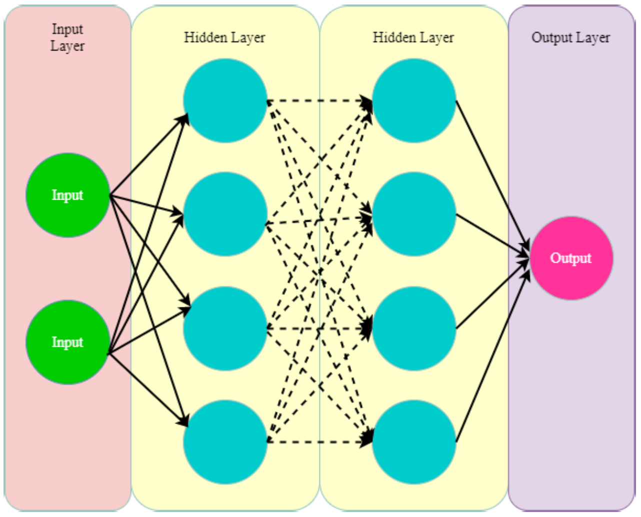


FIGURE 2. A simplified neural network block diagram [46].

Each artificial neuron receives inputs and generates a single output that can be transmitted to multiple other neurons. A neuron blends data input with a set of coefficients, or weights and incorporates bias terms. To generate the output, the weighted sum is transmitted through a nonlinear

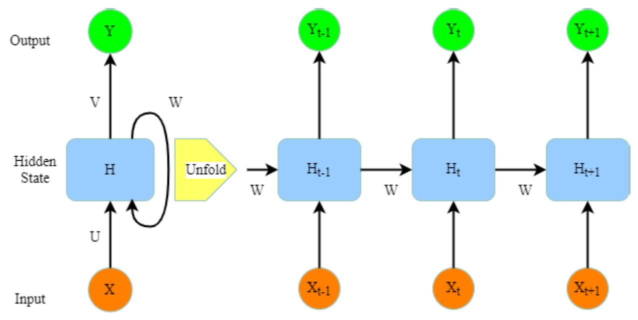


FIGURE 3. Recurrent neural network with its unfold architecture [47].

activation function. If the output of this aggregated weighted sum exceeds the neuron threshold, the neuron is activated and the signal passes to the next layer neuron input. The signal in the network passes through each successive layer until it reaches the output layer. It is commonly known as a feed-forward network. The network trains the data to identify the patterns present within the data and adjust the associated weights to predict the output for a brand new set of similar data. Weight and bias in each neuron and edge are optimized using the backpropagation technique, which employs gradient descent [46]. The last part of the BHyPreC model incorporates a fully connected dense neural network having a single output layer neuron to calculate the final prediction.

C. RECURRENT NEURAL NETWORK

RNN is a special type of ANN algorithm which is convenient for handling the data that are sequential in nature such as time-series data. RNN is equipped with a feedback loop and it can process input sequences of variable length by utilizing their internal state or memory. In this case, the output from step $n-1$ is fed back into the network to leverage the outcome of step n, and so on for each subsequent step. This network style can be used for our cloud workload prediction role by forecasting the next value based on previous load values. RNN model consisting of one hidden layer with its spread out shape is demonstrated in Fig. 3, where X_t is the input data at time t, H_t is the hidden state at that time step, and Y_t is the output of the RNN [47].

Historical cloud workload data such as CPU usage value will be used as X_t in our forecasting. From Fig. 3, Hidden state H_t of the RNN depends on the previous hidden state values and the current time step output. H_t can be calculated as:

$$H_t = \alpha_H \left(UX_t + WH_t + b_H\right) \tag{1}$$

The output state Y_t for the input X_t can be computed depending on the hidden state H_t at time step t as follows:

$$Y_t = \alpha_Y \left(VH_t + b_Y\right) \tag{2}$$

Here, α_H and α_Y are the non-linear activation function such as *tanh*, *ReLU*, or *sigmoid* function. Again, U, V, W and b are the parameter matrices and vector, respectively. Unlike a typical deep neural network, RNN employs the same array of parameters matrices and vectors throughout all stages. It significantly reduces the parameter numbers that RNN needs to learn.

D. LONG SHORT-TERM MEMORY

Vanishing gradient and short-term memory problems are most prevalent in RNN due to backpropagation through time technique. This issue can be addressed by using a special kind of RNN, known as LSTM. It is also adept of learning dependencies of long-term. Fig. 4 shows the overview of an LSTM cell.

An LSTM cell has three gates namely input gate, forget gate, and output gate. All of the gates having a sigmoid activation function that acts as a filter and decides which information to keep and which one to forget. Besides the gates, LSTM has the following components [47]:

- X_t is the input data at time step t.
- H_{t-1} is the hidden state at time step $t - 1$ or from previous time step. It serves as short-term memory in LSTM.
- C_{t-1} is the cell state at time step $t-1$. It is responsible for long-term memory in LSTM. It is updated at two points inside LSTM.
- σ is the sigmoid activation function.
- *tanh* is the activation function that gives output in the range of 1 to -1.
- f_t is the output of the sigmoid function of forget gate which decides whether to forget or remember the previous time step memory $C_t - 1$.
- i_t is the output of the input sigmoid gate which controls what new input data information to add to the cell state.
- $c_t^{\sim}$ is the candidate cell state which is the result of the non-linear transformation of the external input data X_t using *tanh*.

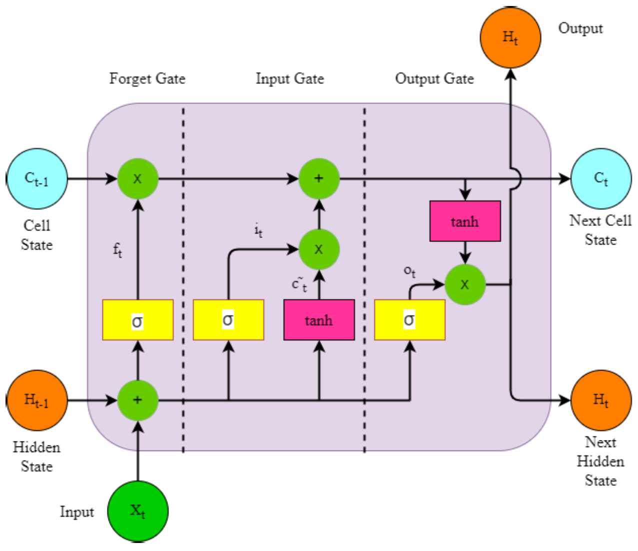


FIGURE 4. A LSTM cell architecture [5].

- o_t is the output of the sigmoid activation of the vector created by the pointwise addition of previous hidden state $H_t - 1$ and external input X_t. It determines what portion of the new cell state data should pass over to output H_t and hidden state H_t.
- C_t is the next cell state.
- H_t is the output of the LSTM cell and acts as the next hidden state.

With each new external input data X_t, a LSTM cell block undergoes the following internal gate updates [47]:

$$f_t = \sigma\left(W_f\,[H_{t-1}, X_t] + b_f\right) \tag{3}$$
$$i_t = \sigma\left(W_i\,[H_{t-1}, X_t] + b_i\right) \tag{4}$$
$$o_t = \sigma\left(W_o\,[H_{t-1}, X_t] + b_o\right) \tag{5}$$
$$c_t^{\sim} = tanh\left(W_c\,[H_{t-1}, X_t] + b_c\right) \tag{6}$$

Finally, an LSTM updates the next cell state C_t and output H_t by using the results of internal gates and candidate cell state as follows:

$$C_t = f_t \otimes C_{t-1} \oplus i_t \otimes c_t^{\sim} \tag{7}$$
$$H_t = o_t \otimes tanh\,(C_t) \tag{8}$$

In our proposed architecture, we used three LSTM units. Through proper tuning, we choose the memory units as 64 for two (in the left side of GRU) and 50 for last LSTM units (in the right side of GRU).

E. GATED RECURRENT UNIT

GRU is the advanced version of RNN equipped with a gating mechanism. It is capable of preventing vanishing gradient problem that takes place in traditional RNN. It is comparable with LSTM but with fewer training parameters. As a result, it is faster than LSTM with lesser training time and has less memory consumption. It exposes all the hidden states without any control. For a shorter sequence of input data, GRU is a more suitable version of RNN. Fig. 5 shows the architecture of the GRU unit.

Unlike LSTM, GRU has two gates namely, reset gate and update gate. Update gate combines the functionality of forget gate and input gate of LSTM. It aids the model in identifying how much prior data (from previous time steps) can be moved on to the future. On the other hand, the reset gate is used to determine how much data from the past can be erased. GRU also removes the cell state that is present in LSTM and transfer data over the hidden state. The components presented in the GRU architecture unit are:

- X_t is the external input data at time step t.
- H_{t-1} is the hidden state at previous time step $t - 1$.
- r_t is the reset gate vector which after point-wise multiplication with H_{t-1} controls the amount of past information to forget.
- z_t is the output of the update gate sigmoid function which determines the amount of information of the previous hidden layer that will pass along to the next hidden state.
- $\hat{h}_t$ is the candidate activation vector which uses reset gate vector r_t to preserve the past relevant information.
- Y_t is the output for the input data X_t at time step t.
- H_t is the next hidden state value and it is the same as the Y_t.

Each internal gate and candidate activation vector inside a GRU cell for each new input data at time t undergoes the following updates [42]:

$$z_t = \sigma\left(W_z X_t + U_z H_{t-1} + b_z\right) \tag{9}$$
$$r_t = \sigma\left(W_r X_t + U_r H_{t-1} + b_r\right) \tag{10}$$
$$\hat{h}_t = tanh\left(W_h X_t + U_h\,(r_t \otimes h_{t-1}) + b_h\right) \tag{11}$$

The final output of the GRU cell unit and the next hidden layer can be obtained by manipulating the update gate vector. The final output is updated as follows [42]:

$$H_t = z_t \otimes \hat{h}_t + (1 - z_t) \otimes H_{t-1} \tag{12}$$

In our proposed architecture, illustrated in Fig. 1, a GRU layer is incorporated in between LSTM layers. We select the output neuron layer of our GRU unit after proper tuning. It is set to 64 units.

F. BIDIRECTIONAL LONG SHORT-TERM MEMORY

Bi-LSTM is an exclusive variant of RNN that consists of two LSTM layers combined together. Here, one LSTM unit processes inputs in the forward direction, and the other unit processes inputs in the backward direction. It is an extension of traditional LSTM, which preserves both past and future information. This results in a substantial amount of increment of the information accessible to the network. Thus, the network can better understand the context. Fig. 6 shows the block diagram of Bi-LSTM network. Each LSTM block shown in Fig. 6 follows the internal mechanism introduced in sub-section III-D.

The forward layer LSTM in Bi-LSTM at time step t takes into account the input data sequence X_t and past hidden state value $\overrightarrow{h}_{t-1}$. Then, it calculates the present hidden state value $\overrightarrow{h}_t$. The hidden state value is updated and processed

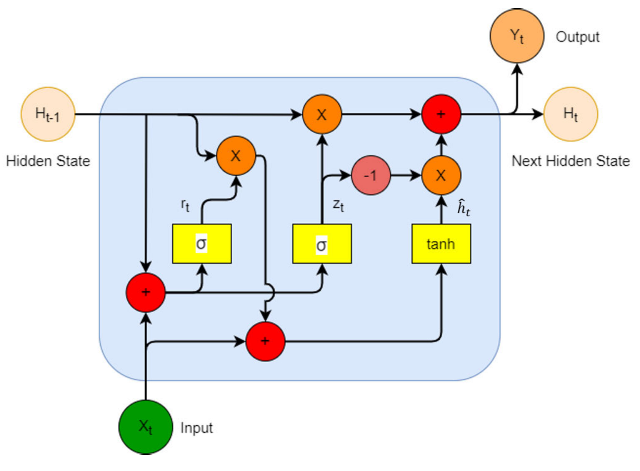


FIGURE 5. Architecture of gated recurrent unit cell [42].

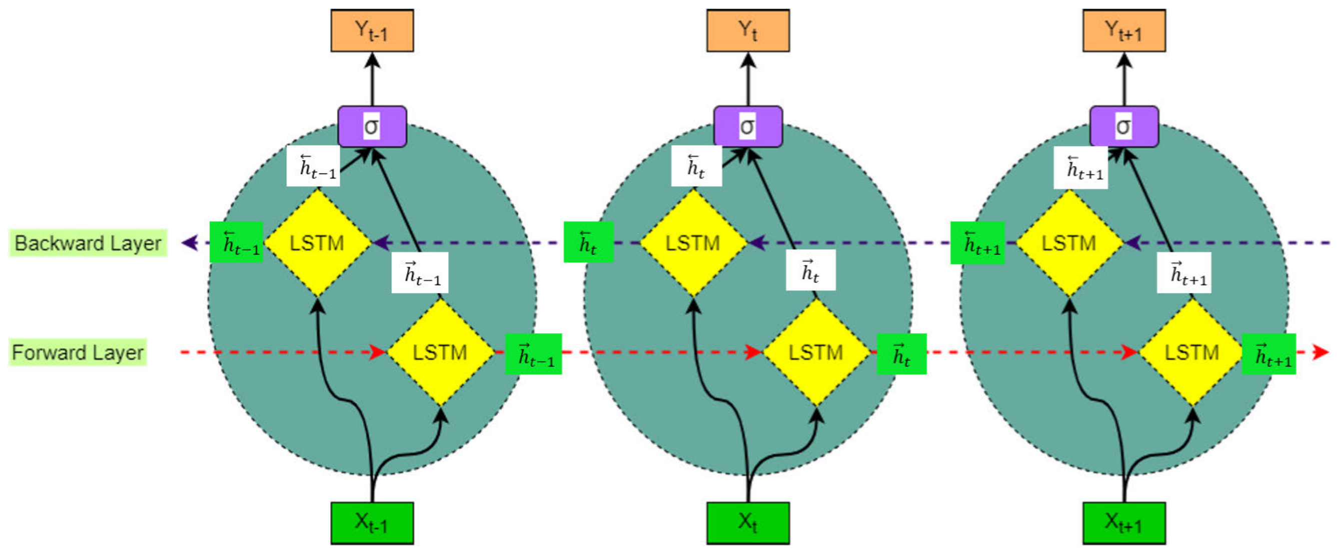


FIGURE 6. A bidirectional long short-term memory unit [47].

following the internal equation described below:

$$\overrightarrow{f}_t = \sigma\left(\overrightarrow{W}_f\left[\overrightarrow{h}_{t-1}, \overrightarrow{X}_t\right] + \overrightarrow{b}_f\right) \quad (13)$$
$$\overrightarrow{i}_t = \sigma\left(\overrightarrow{W}_i\left[\overrightarrow{h}_{t-1}, \overrightarrow{X}_t\right] + \overrightarrow{b}_i\right) \quad (14)$$
$$\overrightarrow{o}_t = \sigma\left(\overrightarrow{W}_o\left[\overrightarrow{h}_{t-1}, \overrightarrow{X}_t\right] + \overrightarrow{b}_o\right) \quad (15)$$
$$\overrightarrow{\tilde{c}}_t = tanh\left(\overrightarrow{W}_c\left[\overrightarrow{h}_{t-1}, \overrightarrow{X}_t\right] + \overrightarrow{b}_c\right) \quad (16)$$
$$\overrightarrow{C}_t = \overrightarrow{f}_t \otimes \overrightarrow{C}_{t-1} \oplus \overrightarrow{i}_t \otimes \overrightarrow{\tilde{c}}_t \quad (17)$$
$$\overrightarrow{h}_t = \overrightarrow{o}_t \otimes tanh\left(\overrightarrow{C}_t\right) \quad (18)$$

Overall, a forward layer LSTM output hidden layer equation for time step t can be summarized as:

$$\overrightarrow{h}_t = f\left(\overrightarrow{X}_t, \overrightarrow{h}_{t-1}, \overrightarrow{\square}\mathbf{LSTM}\right) \quad (19)$$

Here, $\overrightarrow{\square}\mathbf{LSTM}$ indicates the internal operation of present state forward LSTM unit. Again, backward layer in Bi-LSTM updates its hidden state value $\overleftarrow{h}_t$ by taking into consideration the future hidden state value $\overleftarrow{h}_{t+1}$ and present input data sequence X_t. All the internal updates that takes place inside a backward layer are as follows:

$$\overleftarrow{f}_t = \sigma\left(\overleftarrow{W}_f\left[\overleftarrow{h}_{t+1}, \overleftarrow{X}_t\right] + \overleftarrow{b}_f\right) \quad (20)$$
$$\overleftarrow{i}_t = \sigma\left(\overleftarrow{W}_i\left[\overleftarrow{h}_{t+1}, \overleftarrow{X}_t\right] + \overleftarrow{b}_i\right) \quad (21)$$
$$\overleftarrow{o}_t = \sigma\left(\overleftarrow{W}_o\left[\overleftarrow{h}_{t+1}, \overleftarrow{X}_t\right] + \overleftarrow{b}_o\right) \quad (22)$$
$$\overleftarrow{\tilde{c}}_t = tanh\left(\overleftarrow{W}_c\left[\overleftarrow{h}_{t+1}, \overleftarrow{X}_t\right] + \overleftarrow{b}_c\right) \quad (23)$$
$$\overleftarrow{C}_t = \overleftarrow{f}_t \otimes \overleftarrow{C}_{t-1} \oplus \overleftarrow{i}_t \otimes \overleftarrow{\tilde{c}}_t \quad (24)$$
$$\overleftarrow{h}_t = \overleftarrow{o}_t \otimes tanh\left(\overleftarrow{C}_t\right) \quad (25)$$

In short, the equation for output of hidden state for the backward LSTM layer can be summarized as follows:

$$\overleftarrow{h}_t = f\left(\overleftarrow{X}_t, \overleftarrow{h}_{t+1}, \overleftarrow{\square}\mathbf{LSTM}\right) \quad (26)$$

Here, $\overleftarrow{\square}\mathbf{LSTM}$ represents the overall internal operation of the backward LSTM layer [47]. The final output of the hidden state combines the hidden state of both the forward and backward layer of a Bi-LSTM network along with an activation function. As a result, the impact of past and future data can be traced to the output of a Bi-LSTM network. It has more parameters in comparison to LSTM and GRU networks. Due to its capability to handle complex data, Bi-LSTM networks are often used in the field of time series prediction [48]–[50]. The Bi-LSTM layer used in our BHyPreC model has in total of 128 output layer neurons, considering both forward and backward propagation. Here, the output unit size is selected after proper tuning. Bi-LSTM unit plays a key role in enhancing the long-term prediction accuracy of our model due to its ability to retain both past and future data.

G. TRAINING TIME COMPLEXITY

The total number of training parameters of a model dictates the time needed for a model to be trained. It is also dependent on the computational power of the model simulator. For training our model, we used the Google Colaboratory platform. It is a cloud-based Jupyter notebook environment that saves its notebooks on Google Drive. In total, we trained our model for 36 different cases. The total average training time of our model is 75.68 minutes. The maximum training time of our model is 104.08 minutes and the minimum training time is 20.46 minutes. The maximum training of 104.08 minutes is required for training our model with the ''mean absolute percentage error'' loss function for 'HWS' of 120 and 75% training data. For 'HWS' of 30, 75% training data size, and "mean absolute error" loss function, our model requires a minimum of 20.45 minutes training time. However, in practice, devices with very high computational capacity will be used that will certainly reduce the training time.

IV. RESULT ANALYSIS

The result analysis section is sub-divided into the nine subsections. Together, they describe the dataset to be trained, narrate the evaluation metrics briefly, illustrate the detailed experimental result of our proposed model, and compare the performance of our model with other relevant forecasting models, along with an appropriate statistical test. This section also demonstrates the performance comparison of all the models with the variation of prediction length.

A. DATASET DESCRIPTION AND PRE-PROCESSING

In this work, a distributed data center commonly known as Bitbrain is used to collect large-scale and long-term traces of real data [51]. The dataset includes performance metrics for 1,750 virtual machines from Bitbrains' distributed DC, which specializes in controlled hosting and business computing for businesses. It manages computational capacity using generic VMware provisioning frameworks such as Dynamic Resource Scheduling and Storage Dynamic Resource Scheduling.

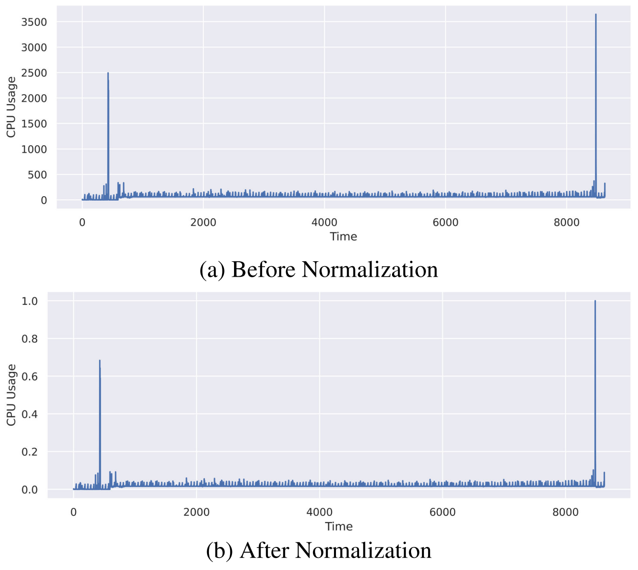
Each file includes the VM performance metrics. These files are grouped by tracks: fastStorage and Rnd. FastStorage consists of 1250 VMs affiliated with SAN storage gadgets, and Rnd is composed of 500 VMs affiliated with either faster Storage Area Network (SAN) or relatively slower Network Attached Storage (NAS) gadgets. The layout of each file is row-based, and each row reflects the observation of performance metrics containing 11 columns. In total, this dataset contains data spanning over 5,446,811 CPU hours, having 23,214 GB memory with 5,501 cores.

After some pre-processing raw dataset is converted into a DataFrame. From this DataFrame, we take 'CPU Usage [MHz]' values for this workload prediction task. For faster convergence and learning, the CPU usage data is normalized in the range of 0 to 1 using the 'MinMaxScaler' function. Fig. 7 shows the CPU usage value over a period of 43,155 minutes. Here, the data are sampled within successive 5 minutes time windows. As a result, the time interval between successive two points in the X-axis of Fig. 7 is of 5 minutes.

B. EVALUATION METRICS

To precisely evaluate the prediction accuracy of our model, various evaluation metrics such as Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) are analyzed. Each evaluation metric projects the model performance. Here, lower metrics value represents better prediction accuracy.

The average of squared differences between original and predicted sequences of an estimator is represented by the MSE. It is defined as:



(a) Before Normalization

(b) After Normalization

FIGURE 7. CPU usage over time.

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(y_k - \overline{y_k})^2 \tag{27}$$

RMSE is a frequently used metric of evaluation that measures the square root of average squared differences between initial and expected sequences. It is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(y_k - \overline{y_k})^2} \tag{28}$$

A very commonly used metric for prediction in time series data is the MAE. It is the average difference between original and predicted values and is defined as:

$$MAE = \frac{1}{n}\sum_{k=1}^{n}|y_k - \overline{y_k}| \tag{29}$$

MAPE is another efficient tool for evaluating the prediction accuracy of the forecasting model. It is defined as:

$$MAPE = \frac{100}{n}\sum_{k=1}^{n}\frac{|y_k - \overline{y_k}|}{y_k} \tag{30}$$

Here, in equations 27, 28, 29, and 30, y_k and $\overline{y_k}$ stands for original and predicted sample values for the k^{th} number of sample, respectively. Again, n denotes the total number of samples.

C. OPTIMUM WINDOW AND DATA SPLIT SIZE ANALYSIS

At first, our model shown in Fig. 1 is trained and later validated using the Bitbrains dataset described in section IV-A. To obtain optimum history window size, we trained our model for four distinct window widths. For each unique window size, we split our dataset into four training-testing split ratios. In total, we obtain a set of 16 different combinations of window size and data split ratio. Initially, the proposed model is compiled using the "mean squared error" loss function for the above set of 16 different combinations. We evaluate the performance of these trained models and calculate MSE and RMSE. Again, for each set of 16 unique combinations, we compile our model using "mean absolute error" and "mean absolute percentage error" loss functions and calculate MAE and MAPE for model evaluation respectively. In total, our proposed model is trained for 36 unique combinations. As a result, we obtained 48 different error values.

To analyze the performance of our prediction model with respect to other established and advanced prediction models, we conducted a systematic training and validation process. We individually trained and validated LSTM, GRU, Bi-LSTM, ARIMA, and Bayesian Information Criterion (BIC) plus smooth filter algorithm [52] using the same dataset and for the same set of unique combinations of parameters. As a result, each model, except ARIMA and BIC plus smooth filter is trained and validated for 48 unique combinations to obtain optimum window and train:test split size. ARIMA belongs to the group of traditional time series prediction models. ARIMA model does not require history window size to train the dataset. Again, BIC plus smooth filter algorithm in [52], chooses the best possible traditional time series model based on the minimum BIC value among six traditional time series models. These six prediction models are Error Trend Seasonal (ETS), SES, HWES, DWES, Automatic ARIMA, and Trigonometric Exponential Smoothing State-Space model with Box-Cox transformation, ARMA errors, Trend and Seasonal Components (TBATS). Again, Savitzky-Golay (SG) and Moving Average (MA) smooth filters are applied to the data points separately. The BIC of the selected model with no smooth filtering, SG filtering, and MA filtering of data points are compared and the combination that gives minimum BIC is selected. This algorithm also does not require history window size to train the dataset. LSTM, GRU, and Bi-LSTM are based on deep learning techniques and require history window size to train the dataset. For performing the above experiments, we used Keras in python and the TensorFlow library backend. We chose the google colaboratory platform for training all of our models and used Adam optimizer to determine the optimal learning rate for compiling the models. For each training and validation, we used a batch size of 64 and an epoch size of 100. Table 4 shows the MSE, RMSE, MAE, and MAPE of CPU usage prediction for varying window size and train:test ratio. Here, 'Window = 30' represents training a model with history window size 30 and so on. Again, '65:35' denotes the splitting of the dataset and taking 65% of data for training and 35% data for testing purposes and so on. Here, in the case of BIC plus smooth filter algorithm SES model with SG filter is selected for the train:test ratio of 65:35 and TWATS model with SG filter is selected for all other train:test ratio.

Analyzing the Table 4, we can see that for varying the window size and ratio of training and testing data, prediction error in terms of MSE, RMSE, MAE, and MAPE varies. The lowest evaluated prediction error for each model with optimum window size and the training-testing split ratio is shown in Table 5 (shown in bold). It is obtained after analyzing the performance metrics of each model for all possible combinations shown in Table 4. After inspecting Table 5, we can see that for each of the evaluation metrics our proposed model leads to the least prediction error in comparison to other state-of-the-art prediction models. In each scenario, the performance of ARIMA is much poorer compared to other forecasting models. In the case of MSE and RMSE, our model gives the least error of forecasting for window size '60' and split ratio '65:35'. However, the lowest MAE is obtained in our model for a window size of 30 and a split ratio of 65:35. Here, the split size ratio remains the same as that for the previous metrics, but history window size changes. Again, the window size of 120 and a split ratio of 70:30 yield to lowest MAPE error.

D. ERROR VALUE COMPARISON

The pictorial representation of the lowest forecasting errors for the best possible combination of window size and a split ratio of each model is shown in Fig. 8. To fit the error values properly in this figure, we multiply MSE by 10 and MAPE by 0.001. As by nature, the MSE value is quite small and the MAPE value is large in comparison to other errors. This certain modification helps us to obtain a purposeful bar chart to properly compare the performance of all the models. Here, we can clearly see that for each of the prediction error evaluations, our proposed BHyPrec model yields the lowest possible error (Green bar), and ARIMA (Light blue bar) gives the highest forecasting error. Therefore, we can say that our model performs better in predicting CPU usage in VMs and the performance of ARIMA is poor in comparison to other deep learning approaches.

Again, another approach to compare the error value is the calculation of the percentage increase or decrease of prediction error in other state-of-the-art forecasting models in comparison to our proposed model. It can be calculated by using the following equation, where Y_p is the error value of our proposed model, and Y_c indicates the error value of the compared model.

$$X_c = \frac{(Y_c - Y_p) * 100}{Y_p} \tag{31}$$

Here, for this experiment, we use LSTM, GRU, Bi-LSTM, and ARIMA as the baseline models to compare. Positive X_c indicates a percentage increase of error value of the compared model with respect to our proposed model and negative X_c indicates the exact opposite. Table 6 shows the percentage increase or decrease of forecasting error in other compared models in comparison to our proposed model for each evaluation metrics.

From this Table 6, we clearly observe that percentage error value increases in all the models in comparison to our proposed model. Here, we take into consideration the best-case scenario for all the models. Therefore, we can say that our

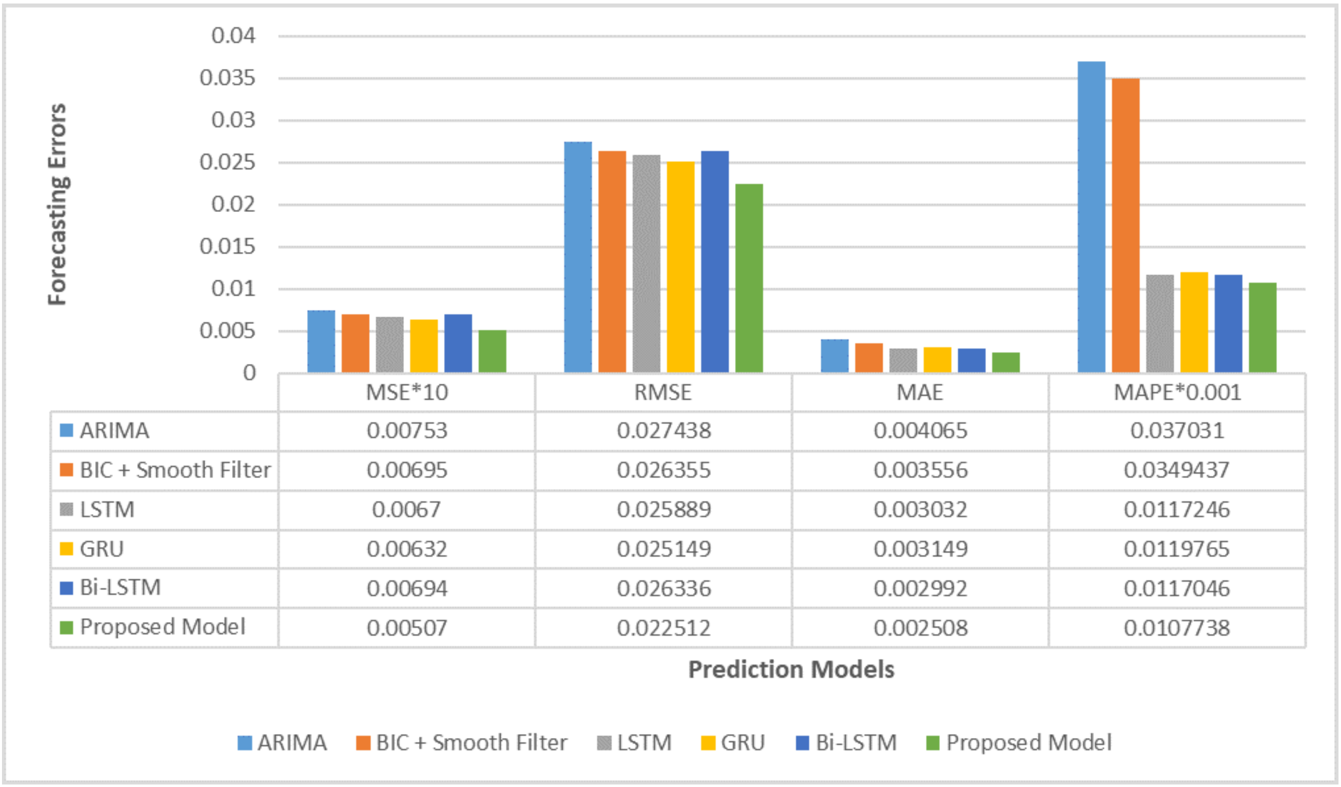
TABLE 4. Performance analysis of all the models for CPU usage prediction.

Methods	Window Size	Train:Test Ratio	MSE	RMSE	MAE	MAPE
ARIMA	N/A	65:35	**0.000753**	**0.027438**	**0.004065**	37.3729
		70:30	0.000874	0.029569	0.004354	37.1983
		75:25	0.001048	0.032369	0.004325	39.4395
		80:20	0.001299	0.036044	0.005233	**37.031**
BIC + Smooth Filters [52]	N/A	65:35	**0.000695**	**0.026355**	**0.003556**	41.8746
		70:30	0.000803	0.028333	0.004082	39.6015
		75:25	0.000953	0.030869	0.005137	**34.9437**
		80:20	0.001187	0.034452	0.005582	35.7016
LSTM	Window=30	65:35	0.001023	0.03199	0.003189	12.2261
		70:30	0.001041	0.032266	0.003857	12.1969
		75:25	0.000885	0.029752	0.003579	12.3824
		80:20	0.001317	0.036294	0.004232	13.0709
	Window=60	65:35	0.000962	0.031016	0.003705	11.812
		70:30	0.001052	0.032438	0.00325	12.163
		75:25	0.001442	0.037972	0.00326	12.6109
		80:20	0.001449	0.03807	0.004551	12.8973
	Window=90	65:35	0.000808	0.028432	0.003038	13.0719
		70:30	0.00072	0.026825	0.003468	12.6183
		75:25	0.000977	0.031256	0.004745	13.0822
		80:20	0.001384	0.037199	0.004524	13.2347
	Window=120	65:35	0.000797	0.028239	**0.003032**	**11.7246**
		70:30	**0.00067**	**0.025889**	0.003354	12.4235
		75:25	0.001139	0.033752	0.003716	13.0995
		80:20	0.001342	0.036631	0.005137	13.2246
GRU	Window=30	65:35	0.000774	0.027821	0.003252	12.3917
		70:30	0.001286	0.035866	0.003411	13.0974
		75:25	0.00132	0.03633	0.003934	12.9049
		80:20	0.001326	0.036409	0.00416	13.991
	Window=60	65:35	0.000712	0.026691	0.004224	12.2636
		70:30	**0.000632**	**0.025149**	0.003172	12.4737
		75:25	0.001365	0.036945	0.00418	12.9735
		80:20	0.001513	0.038903	0.004225	13.4912
	Window=90	65:35	0.000911	0.03019	0.003456	12.0025
		70:30	0.000807	0.028415	0.003695	**11.9765**
		75:25	0.001037	0.032204	0.00357	13.87
		80:20	0.001836	0.042845	0.004597	14.4913

*Continued on next page

TABLE 4. *(Continued.)* Performance analysis of all the models for CPU usage prediction.

Method	Window Size	Train:Test Ratio	MSE	RMSE	MAE	MAPE
	Window=120	65:35	0.000772	0.027779	**0.003149**	12.4653
		70:30	0.001174	0.034268	0.003833	12.289
		75:25	0.002093	0.04575	0.003817	12.6512
		80:20	0.001447	0.038036	0.004766	13.9903
Bi-LSTM	Window=30	65:35	0.000819	0.028613	0.003305	12.0119
		70:30	0.001084	0.032928	0.003319	12.2173
		75:25	0.000882	0.029692	0.003968	12.3019
		80:20	0.000821	0.028662	0.00398	13.6177
	Window=60	65:35	0.000788	0.028077	0.003331	**11.7046**
		70:30	**0.000694**	**0.026336**	0.003405	11.7091
		75:25	0.001075	0.032792	0.004423	11.914
		80:20	0.001319	0.036318	0.004489	13.6163
	Window=90	65:35	0.000936	0.0306	0.003485	12.0244
		70:30	0.000752	0.027428	0.003391	12.3091
		75:25	0.000873	0.029542	0.003576	12.8671
		80:20	0.001087	0.032965	0.004351	13.1198
	Window=120	65:35	0.000716	0.026758	**0.002992**	12.0802
		70:30	0.000868	0.029454	0.003564	11.8903
		75:25	0.001166	0.034152	0.003526	14.207
		80:20	0.001178	0.034322	0.004591	13.4428
Proposed Model	Window=30	65:35	0.000565	0.023766	**0.002508**	11.1799
		70:30	0.000577	0.024021	0.002832	12.3343
		75:25	0.000877	0.02962	0.002883	12.3688
		80:20	0.000887	0.029783	0.003517	12.2959
	Window=60	65:35	**0.000507**	**0.022512**	0.002802	11.1101
		70:30	0.000648	0.025462	0.003026	13.0751
		75:25	0.000788	0.02807	0.003156	11.7641
		80:20	0.000908	0.030133	0.003559	13.507
	Window=90	65:35	0.00057	0.023866	0.002661	12.537
		70:30	0.000687	0.026207	0.002677	12.2912
		75:25	0.000777	0.027881	0.00327	10.8557
		80:20	0.000834	0.028881	0.003846	12.4713
	Window=120	65:35	0.000837	0.028928	0.00266	12.2044
		70:30	0.00072	0.026825	0.002786	**10.7738**
		75:25	0.000775	0.027843	0.003086	12.706
		80:20	0.000937	0.030609	0.003415	13.3193



	MSE*10	RMSE	MAE	MAPE*0.001
ARIMA	0.00753	0.027438	0.004065	0.037031
BIC + Smooth Filter	0.00695	0.026355	0.003556	0.0349437
LSTM	0.0067	0.025889	0.003032	0.0117246
GRU	0.00632	0.025149	0.003149	0.0119765
Bi-LSTM	0.00694	0.026336	0.002992	0.0117046
Proposed Model	0.00507	0.022512	0.002508	0.0107738

FIGURE 8. Comparative error value analysis of different prediction models.

TABLE 5. Summary of lowest error value of each model.

Errors	Methods	Optimum Window Size	Optimum Split Ratio	Lowest Error Value
MSE	ARIMA	N/A	65:35	0.000753
	BIC [52]	N/A	65:35	0.000695
	LSTM	120	70:30	0.00067
	GRU	60	70:30	0.000632
	Bi-LSTM	60	70:30	0.000694
	Our Model	60	65:35	**0.000507**
RMSE	ARIMA	N/A	65:35	0.027438
	BIC [52]	N/A	65:35	0.026355
	LSTM	120	70:30	0.025889
	GRU	60	70:30	0.025149
	Bi-LSTM	60	70:30	0.026336
	Our Model	60	65:35	**0.022512**
MAE	ARIMA	N/A	65:35	0.004065
	BIC [52]	N/A	65:35	0.003556
	LSTM	120	65:35	0.003032
	GRU	120	65:35	0.003149
	Bi-LSTM	120	65:35	0.002992
	Our Model	30	65:35	**0.002508**
MAPE	ARIMA	N/A	80:20	37.031
	BIC [52]	N/A	75:25	34.9437
	LSTM	120	65:35	11.7246
	GRU	90	70:30	11.9765
	Bi-LSTM	60	65:35	11.7046
	Out Model	120	70:30	**10.7738**

TABLE 6. Error percentage increase/decrease analysis of the compared models with respect to our proposed model.

Compared Models	MSE	RMSE	MAE	MAPE
LSTM	+32.1499%	+15.0009%	+20.8931%	+8.82511%
GRU	+24.6548%	+11.7138%	+25.5582%	+11.1632%
Bi-LSTM	+36.8836%	+16.9865%	+19.2983%	+8.6395%
BIC+Filters	+37.0809%	+17.0709%	+41.7863%	+224.34%
ARIMA	+48.5207%	+21.8817%	+62.0813%	+243.714%

model significantly minimizes the error in predicting VMs' CPU usage and can precisely predict the future workload. A high percentage rise of error in ARIMA indicates that our model performs better in comparison to the traditional time series prediction model. In terms of MAPE, the percentage rise in error for ARIMA and BIC + Filters are huge. Our model not only outperforms the classical models but also performs much better compared to other relevant deep learning approaches investigated in this work. Our hybrid model outshines all other RNN based models equipped with the best possible combinations and its margin of error minimization is quite high.

E. STABILITY ANALYSIS

To analyze the stability and harmony of our proposed model, we take into account the CDF of our model along with other compared models. CDF is an accurate indicator to depict the distribution of the error values of models. CDF of distribution always approaches 1 and it acts as a non-decreasing function. The more quickly the CDF curve reaches 1 and more it is

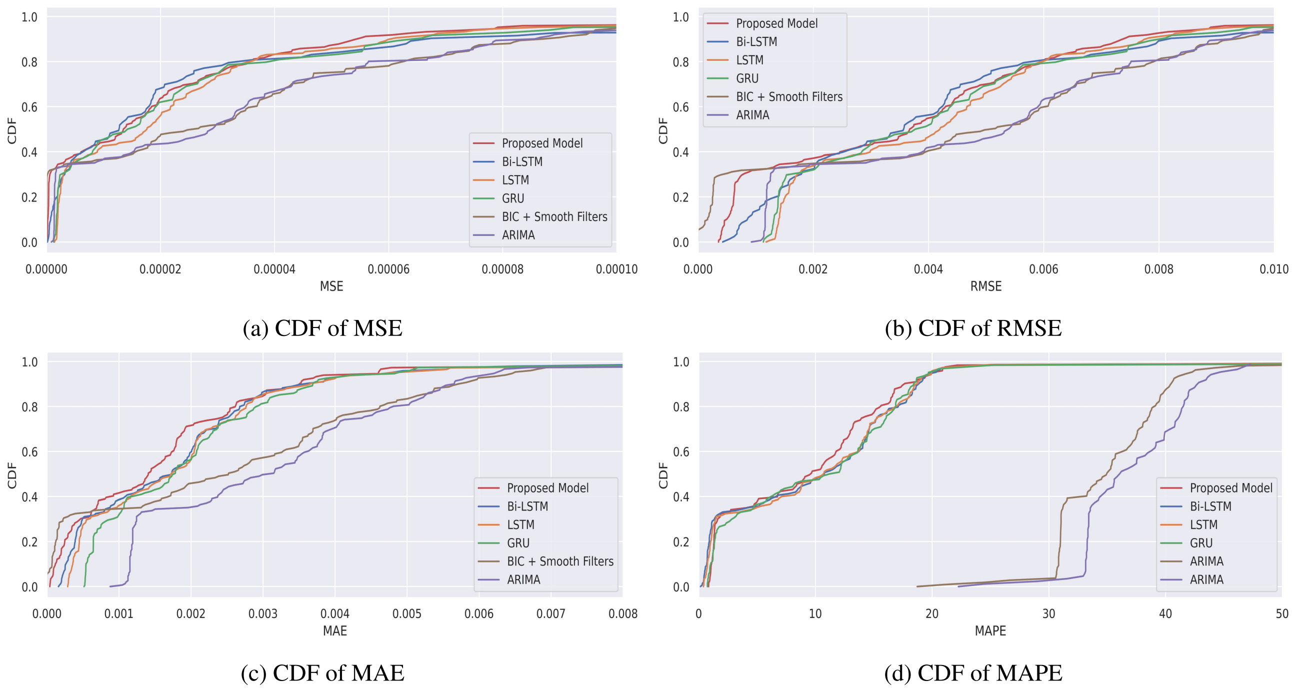


FIGURE 9. **CDF analysis of different prediction models.**

closer to the vertical axis, the higher is the stability of the model. It results in more accurate forecasting of workload.

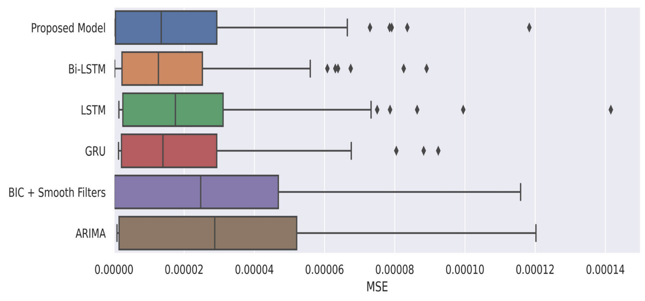
From Fig. 9, it can be observed that almost all the error values are distributed exponentially. The best-case scenario is considered here for each model. The CDF curve of ARIMA and BIC + Smooth Filters lies slightly below the CDF curve of other models. ARIMA CDF curve is farthest away from the vertical axis. The CDF value corresponding to MSE value of 0.00006 in Fig. 9a, RMSE value of 0.006 in Fig. 9b, and MAE value of 0.004 in Fig. 9c for ARIMA and BIC + Smooth Filters are quite less than 1 in comparison to other CDF value of RNN models. Again by analyzing Fig. 9d, we can clearly see that the CDF curve reaches 1 at a larger error value and its distance from the vertical axis is the highest. It indicates that ARIMA and BIC + Smooth Filters perform worse compared to other models for all the evaluation metrics, especially for MAPE. The non-linear nature of the data leads ARIMA and BIC + Smooth Filters to generate higher errors while predicting future CPU usage.

Although, the CDF curves of all the RNN models are very close to each other, in each case our proposed model reaches the maximum value of 1 faster than others. Again our proposed models' CDF curve lies closer to vertical axis almost throughout the graph in 9c and 9d. Although in 9a and 9b in some portions the 'Bi-LSTM' curve lies closer to the vertical axis, overall our proposed models' curve lies closer to the y-axis and reaches maximum value quicker. As a result, we can say that the proposed model predicts the workload most accurately and it is most stable among the studied models. The performances of other RNN models are almost similar in nature. They perform better than ARIMA and BIC + Smooth Filters, but show slightly less stability and predict error less effectively than our proposed model.

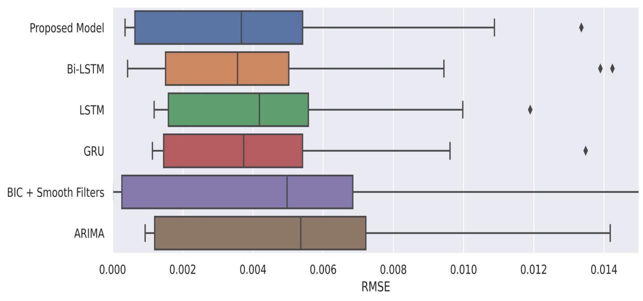
F. ERROR DISTRIBUTION ANALYSIS

Box plot analysis of different prediction models with the best possible combination for each type of evaluated error is expressed in Fig. 10. The distribution and skewness of the error data of different prediction models are visually represented through the box plots. Here, some outlier values presented in the ARIMA and BIC + Smooth Filters models are omitted for better visualization of the box plot. The median, upper quartile, whiskers, and maximum score of our suggested models are smaller than the other four prediction models as shown in Fig. 10c and Fig. 10d. It indicates that our model has better prediction accuracy than all other models for MAE and MAPE analysis. Here the performance of ARIMA and BIC + Smooth Filters are much poorer than other RNN models, indicating supremacy of RNN models in predicting non-linear time series data with sudden peak values.

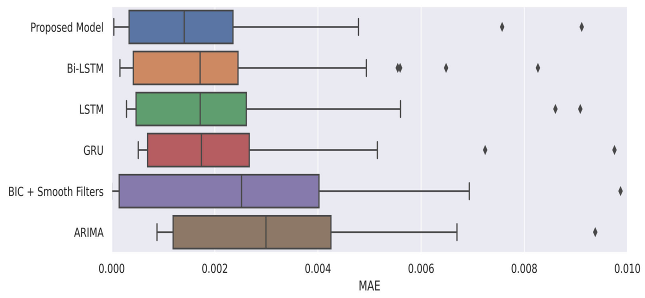
In Fig. 10a and 10b, the median value of Bi-LSTM is slightly smaller than our proposed model. However, the minimum value and first quartile value of Bi-LSTM are larger than the proposed hybrid model. It clearly indicates that the prediction accuracy of our model is better than Bi-LSTM. Analyzing median, first quartile, and third quartile values, we can see that rest of the prediction models also shows poor prediction accuracy in comparison to our model in terms of MSE and RMSE analysis. Again, ARIMA shows the worst performance.



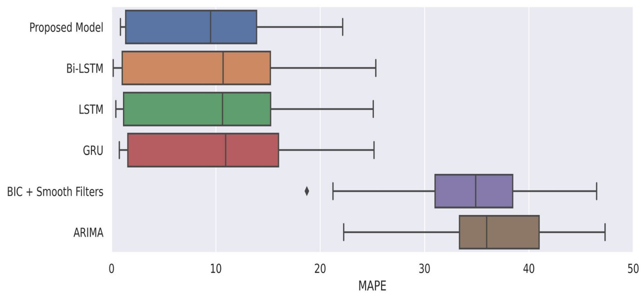
(a) Box plot of MSE



(b) Box plot of RMSE



(c) Box plot of MAE



(d) Box plot of MAPE

FIGURE 10. Box plot analysis of different prediction models.

G. PREDICTION LENGTH ANALYSIS

The analysis of the effect of prediction model performance with the increment of the forecasting step is also a key performance indicator. The model which gives less forecasting error for large prediction steps is more suitable in multi-step workload prediction. Knowledge of the precise future workload patterns will provide us an upper hand in VM migration and consolidation tasks without violating SLA. It will also allow us to undertake efficient task scheduling and ensure the reduction of energy consumption through proper allocation and utilization of cloud resources.

Fig. 11 portrays the effect of prediction length or steps on different forecasting models for each of the evaluation metrics. Here, the interval between each successive prediction step is of 5 minutes. Therefore, an error corresponding to the prediction step of '1200' indicates '100 Hours' ahead CPU usage workload prediction error. The model which gives the least amount of error throughout the entire prediction length and which exhibits minimal error due to the variation of prediction step is most suitable in multi-step ahead prediction. Fig. 11 clearly shows that error increases with the increase of prediction steps. From Fig. 11a and 11b, we can see that MSE and RMSE value increase, respectively, with the increment of prediction length for the LSTM, GRU, and ARIMA models. In the case of Bi-LSTM and our proposed model, the rate of prediction error increase is small due to their ability to extract temporal features by learning workload pattern change of both past and future. ARIMA brings out the most forecasting error with an increase in prediction length. Both MSE and RMSE increase drastically in cases of ARIMA and BIC + Smooth Filter models, with the increases of prediction steps. Thereafter, ARIMA and BIC + Smooth Filter models turn out to be the least suitable for multi-step ahead prediction.

Analyzing Fig. 11c and Fig. 11d, we can say that our model generates the least error while predicting future CPU usage and the performance of ARIMA is the worst. In the case of the ARIMA and BIC + Smooth Filters model, the margin of MAPE is substantially higher for the future time-ahead prediction in comparison to RNN and hybrid models. Again with the increase of prediction length, MAE increases dramatically in ARIMA. In both cases, our model can carry out precise predictions with generating the least error. The effect of prediction length variation is also minimum in the case of our model. In each of the above scenarios shown in Fig. 11, by generating the least amount of error for the time ahead prediction, our model appears to be the most suitable candidate for multi-step ahead forecasting.

H. MULTI-STEP AHEAD FORECASTING

Our suggested hybrid model not only predicts single-step prediction precisely but also can accurately foresight long time ahead CPU usage pattern variations. A long-time-ahead CPU usage prediction time series curve along with the actual CPU usage workload curve is shown in Fig. 12. Here, interval length is of 5 minutes.

Six hours in front conducted prediction result shown in Fig. 12a clearly exhibits the ability of our model to accurately identify the sudden workload peak. Here. the prediction load curve almost coincides with the actual CPU usage curve with very little deviation and slightly damped peak amplitude value. 1 day, 3 days, and 5 days ahead prediction shown in Fig. 12b, Fig. 12c, and Fig. 12d, respectively, also certify

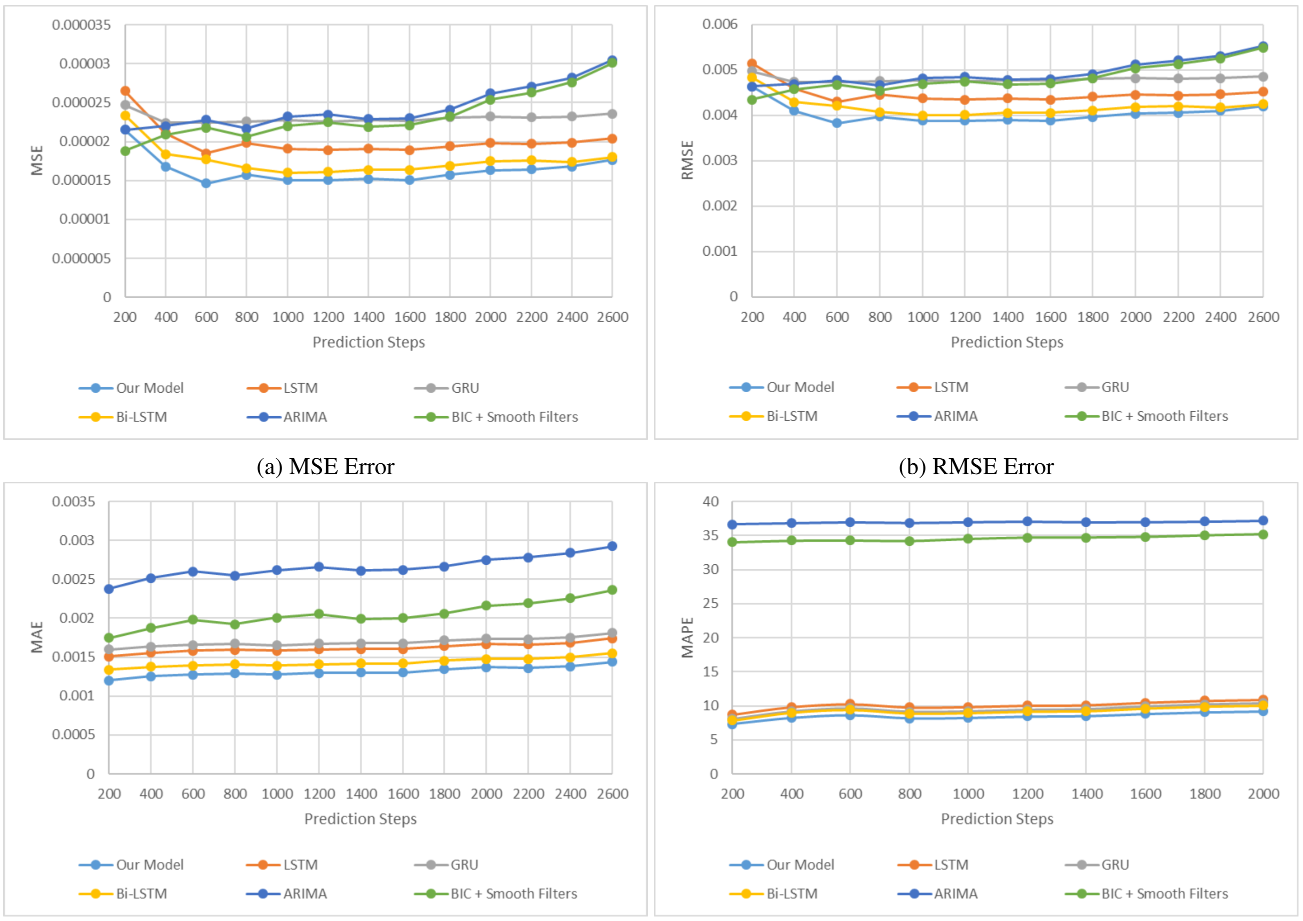


FIGURE 11. **Prediction length effect comparison of different forecasting models.**

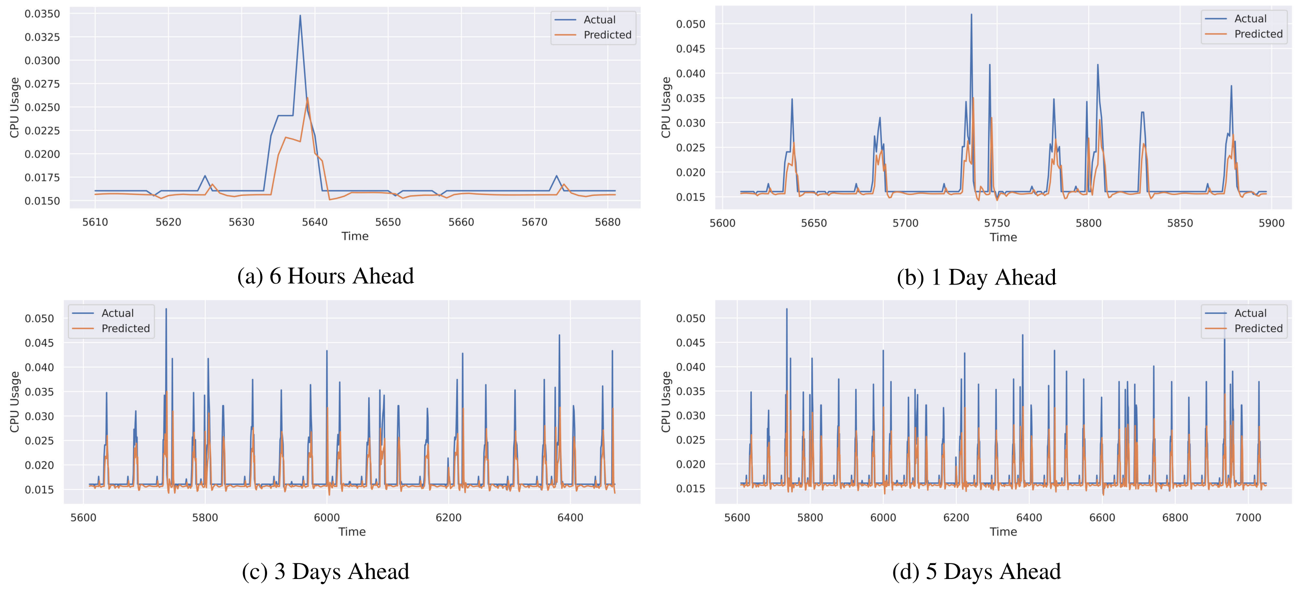


FIGURE 12. **Multi-step ahead CPU usage prediction using proposed hybrid model.**

the aptness of our model in multi-step ahead prediction. In each of the scenario, both the curve coincides with each other despite having long prediction windows. It showcases that our model is capable to possess precise forecasting and is very slightly affected by the range of the prediction window. In each of these scenarios, we can see that our model successfully identifies all the sudden CPU usage peaks in the future. As a result, our model turns out to be most suitable for a long-step-ahead future workload prediction.

I. STATISTICAL TEST

To precisely identify the performance of the proposed method we conduct a statistical test. The main purpose of this test is to validate the performance analysis of our proposed model. It exhibits the statistical significance of the obtained errors. For this purpose, we perform a two-tailed T-test as shown in [53]. The equation for mean (μ), variance (σ^2), and critical value (τ_t) is as follows:

$$\mu = \frac{1}{m}\sum_{k=1}^{m}\epsilon_k \tag{32}$$

$$\sigma^2 = \frac{1}{m-1}\sum_{k=1}^{m}(\epsilon_k - \mu)^2 \tag{33}$$

$$\tau_t = \frac{\sqrt{m}\,|\mu - \epsilon_0|}{\sigma} \tag{34}$$

Here, ϵ_k represents the k-th error of the test value, m is the number of test steps taken into consideration for a statistical test, ϵ_0 is the maximum assumed error value. Now, for the statistical test of MAE, with m = 1001, the mean test value of the MAE for our proposed method obtained from (32) is 0.001273. From (33), we calculate the variance of our proposed model as 0.000011. By taking $\epsilon_0 = 0.0015$, the critical value of our proposed model obtained from (34) is 2.1991, which is larger than the given 1.96 (for confidence level 95%) by the two-tailed T-test table. With a confidence level of 95, this result demonstrates that the suggested model's test MAE mean (0.001273) is smaller than the anticipated test MAE (0.0015). The results for the other compared state-of-the-art models are calculated in a similar process and are shown in Table 7.

TABLE 7. Statistical MAE test of the compared models with respect to the proposed BHyPreC model (ϵ_0 = 0.0015) for 95% confidence level.

Model	μ	σ^2	ϵ_0	τ_t	T-Table Value
ARIMA	0.0026	0.000016	0.0029 > **0.0015**	2.1183	1.96
BIC [52]	0.0020	0.000018	0.0023 > **0.0015**	2.0762	1.96
LSTM	0.0015	0.000011	0.0017 > **0.0015**	2.1835	1.96
GRU	0.0015	0.000011	0.0018 > **0.0015**	2.1209	1.96
Bi-LSTM	0.0014	0.000011	0.0017 > **0.0015**	2.1719	1.96

Analyzing Table 7, we can see that for the statistical test of MAE, the maximum assumed MAE value ϵ_0, for each of the compared models is greater than the maximum assumed MAE value of the proposed model (ϵ_0 = 0.0015). It succinctly explains the statistical significance of our proposed model over other state-of-the-art baseline models. It also proves that the observed performance supremacy of our model is not due to experimental errors.

V. CONCLUSION AND FUTURE WORK

Accurate forecasting of a long time in front, future CPU usage workload is necessary for conducting efficient VM migration and unification tasks, resource allocation, and job scheduling tasks without breaching SLA protocols. In this paper, we proposed an RNN based novel hybrid model named BHyPreC to address these issues. Our model combines Bi-LSTM, LSTM, and GRU units to implement a deep learning-based approach to tackle the non-linearity of time series data effectively. Four of the most relevant prediction models in this field are also trained with the same dataset to compare and evaluate the performance of our model. Result analysis visibly indicates that our proposed model equipped with the best possible combination outperforms the best case scenario of all other compared models. Error percentage increase, CDF, and box plot analysis for each metric evaluation also certify the superiority of our model. An increase in the prediction step size also proves a minimal effect on our prediction model in comparison to the other three RNN based models and one traditional time series prediction model. As a result, in multi-step 5 days ahead prediction, our model can accurately reconstruct the workload pattern with all the sudden peaks and falls. Therefore, our model can be used as an efficient tool in resource allocation, scheduling, load balancing, and VM migration task design by having the proper insight into upcoming CPU usage data.

In the future, we will try to build a more robust prediction model which will ensemble various features from different datasets. It will provide large-scale flexibility in the prediction model. Again, we will try to design an energy-efficient resource allocation algorithm, based on the upcoming CPU usage knowledge obtained from the proposed hybrid model.

ACKNOWLEDGMENT

The CPU workload traces used in this research is taken from the GRID Workloads Archive. It is provided by Bitbrains IT Services Inc.

REFERENCES

[1] P. M. Mell and T. Grance, "The NIST definition of cloud computing," *Nat. Inst. Sci. Technol.*, vol. 800, no. 7, p. 145, Sep. 2011.

[2] R. Beri and V. Behal, "Cloud computing: A survey on cloud computing," *Int. J. Comput. Appl.*, vol. 111, no. 16, pp. 19–22, Jan. 2015.

[3] O. Ali, A. Shrestha, J. Soar, and S. F. Wamba, "Cloud computing-enabled healthcare opportunities, issues, and applications: A systematic review," *Int. J. Inf. Manage.*, vol. 43, pp. 146–158, Dec. 2018.

[4] A. Mahalle, J. Yong, X. Tao, and J. Shen, "Data privacy and system security for banking and financial services industry based on cloud computing infrastructure," in *Proc. IEEE 22nd Int. Conf. Comput. Supported Cooperat. Work Design ((CSCWD))*, May 2018, pp. 407–413.

[5] H. M. Nguyen, S. Woo, J. Im, T. Jun, and D. Kim, "A workload prediction approach using models stacking based on recurrent neural network and autoencoder," in *Proc. IEEE 18th Int. Conf. High Perform. Comput. Commun., IEEE 14th Int. Conf. Smart City, IEEE 2nd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2016, pp. 929–936.

[6] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *J. Netw. Comput. Appl.*, vol. 52, pp. 11–25, Jun. 2015.

[7] M. D. De Assuncao, J.-P. Gelas, L. Lefevre, and A.-C. Orgerie, "The Green Grid'5000: Instrumenting and using a Grid with energy sensors," in *Remote Instrumentation for eScience and Related Aspects*. New York, NY, USA: Springer, 2012, pp. 25–42.

[8] I. Alzamil and K. Djemame, "Energy prediction for cloud workload patterns," in *Proc. Int. Conf. Econ. Grids, Clouds, Syst., Services*. Cham, Switzerland: Springer, 2016, pp. 160–174.

[9] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implement.*, vol. 2, 2005, pp. 273–286.

[10] G. Zhang, X. Zhu, W. Bao, H. Yan, and D. Tan, "Local storage-based consolidation with resource demand prediction and live migration in clouds," *IEEE Access*, vol. 6, pp. 26854–26865, 2018.

[11] X. Tang, X. Liao, J. Zheng, and X. Yang, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Comput.*, vol. 21, no. 3, pp. 1581–1593, Sep. 2018.

[12] L. Li, J. Dong, D. Zuo, and J. Wu, "SLA-aware and energy-efficient VM consolidation in cloud data centers using robust linear regression prediction model," *IEEE Access*, vol. 7, pp. 9490–9500, 2019.

[13] W. Hussain, F. K. Hussain, M. Saberi, O. K. Hussain, and E. Chang, "Comparing time series with machine learning-based prediction approaches for violation management in cloud SLAs," *Future Gener. Comput. Syst.*, vol. 89, pp. 464–477, Dec. 2018.

[14] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *Proc. 6th Int. Conf. Syst. Syst. Eng.*, Jun. 2011, pp. 276–281.

[15] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2012, pp. 1287–1294.

[16] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, "PRACTISE: Robust prediction of data center time series," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2015, pp. 126–134.

[17] M. E. Karim, M. Foysal, and S. Das, "Stock price prediction using bi-lstm and gru based hybrid deep learning approach," in *Proc. Int. Conf. Comput. Technol. Solutions Artif. Intell.*, 2021.

[18] M. E. Karim and S. Ahmed, "A deep learning-based approach for stock price prediction using bidirectional gated recurrent unit and bidirectional long short term memory model," in *Proc. IEEE Global Conf. Advancement Technol. (GCAT)*, Oct. 2021.

[19] M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi, "Deep learning-based stock price prediction using LSTM and bi-directional LSTM model," in *Proc. 2nd Novel Intell. Lead. Emerg. Sci. Conf. (NILES)*, Oct. 2020, pp. 87–92.

[20] A. G. Salman, Y. Heryadi, E. Abdurahman, and W. Suparta, "Single layer & multi-layer long short-term memory (LSTM) model with intermediate variables for weather forecasting," *Proc. Comput. Sci.*, vol. 135, pp. 89–98, Jan. 2018.

[21] C. Dabrowski and F. Hunt, "Using Markov chain analysis to study dynamic behaviour in large-scale grid systems," in *Proc. 7th Australas. Symp. Grid Comput. e-Res.*, 2009, pp. 29–40.

[22] M. A. S. Monfared, R. Ghandali, and M. Esmaeili, "A new adaptive exponential smoothing method for non-stationary time series with level shifts," *J. Ind. Eng. Int.*, vol. 10, no. 4, pp. 209–216, Dec. 2014.

[23] M. S. Aslanpour, S. E. Dashti, M. Ghobaei-Arani, and A. A. Rahmanian, "Resource provisioning for cloud applications: A 3-D, provident and flexible approach," *J. Supercomput.*, vol. 74, no. 12, pp. 6470–6501, Dec. 2018.

[24] A. Shahidinejad, M. Ghobaei-Arani, and L. Esmaeili, "An elastic controller using Colored Petri Nets in cloud computing environment," *Cluster Comput.*, vol. 23, no. 2, pp. 1–27, 2019.

[25] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct. 2014.

[26] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "CloudInsight: Utilizing a council of experts to predict future cloud application workloads," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 41–48.

[27] D. Chen, X. Zhang, L. L. Wang, and Z. Han, "Prediction of cloud resources demand based on hierarchical Pythagorean fuzzy deep neural network," *IEEE Trans. Services Comput.*, early access, Mar. 25, 2019, doi: 10.1109/TSC.2019.2906901.

[28] Z. Chen, Y. Zhu, Y. Di, and S. Feng, "Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network," *Comput. Intell. Neurosci.*, vol. 2015, pp. 1–14, Jan. 2015.

[29] R. Khorsand, M. Ghobaei-Arani, and M. Ramezanpour, "A self-learning fuzzy approach for proactive resource provisioning in cloud environment," *Softw., Pract. Exper.*, vol. 49, no. 11, pp. 1618–1642, Nov. 2019.

[30] R. Khorsand, M. Ghobaei-Arani, and M. Ramezanpour, "FAHP approach for autonomic resource provisioning of multitier applications in cloud computing environments," *Softw., Pract. Exper.*, vol. 48, no. 12, pp. 2147–2173, Dec. 2018.

[31] A. Shahidinejad, M. Ghobaei-Arani, and M. Masdari, "Resource provisioning using workload clustering in cloud computing environment: A hybrid approach," *Cluster Comput.*, vol. 24, no. 1, pp. 319–342, Mar. 2021.

[32] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen, "An adaptive prediction approach based on workload pattern discrimination in the cloud," *J. Netw. Comput. Appl.*, vol. 80, pp. 35–44, Feb. 2017.

[33] D. F. Kirchoff, M. Xavier, J. Mastella, and C. A. F. D. Rose, "A preliminary study of machine learning workload prediction techniques for cloud applications," in *Proc. 27th Euromicro Int. Conf. Parallel, Distrib. Netw. Process. (PDP)*, Feb. 2019, pp. 222–227.

[34] K. S. Qaddoum, N. N. E. Emam, and M. A. Abualhaj, "Elastic neural network method for load prediction in cloud computing grid," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 2, p. 1201, Apr. 2019.

[35] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host CPU utilization in cloud computing using recurrent neural networks," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2017, pp. 67–72.

[36] A. Shahidinejad and M. Ghobaei-Arani, "Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach," *Softw., Pract. Exper.*, vol. 50, no. 12, pp. 2212–2230, Dec. 2020.

[37] D. Janardhanan and E. Barrett, "CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2017, pp. 55–60.

[38] Y. Zhu, W. Zhang, Y. Chen, and H. Gao, "A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, pp. 1–18, Dec. 2019.

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

[40] Y. Guo and W. Yao, "Applying gated recurrent units pproaches for workload prediction," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–6.

[41] C. Peng, Y. Li, Y. Yu, Y. Zhou, and S. Du, "Multi-step-ahead host load prediction with GRU based encoder-decoder in cloud computing," in *Proc. 10th Int. Conf. Knowl. Smart Technol. (KST)*, Jan. 2018, pp. 186–191.

[42] Y. Cheng, C. Wang, H. Yu, Y. Hu, and X. Zhou, "GRU-ES: Resource usage prediction of cloud workloads using a novel hybrid method," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City, IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 1249–1256.

[43] M. N. Hasan Shuvo, M. N. H. Shuvo, M. M. S. Maswood, M. M. S. Maswood, A. G. Alharbi, and A. G. Alharbi, "LSRU: A novel deep learning based hybrid method to predict the workload of virtual machines in cloud data center," in *Proc. IEEE Region Symp. (TENSYMP)*, Jun.'2020, pp. 1604–1607.

[44] H. M. Nguyen, G. Kalra, T. J. Jun, S. Woo, and D. Kim, "ESNemble: An echo state network-based ensemble for workload prediction and resource allocation of web applications in the cloud," *J. Supercomput.*, vol. 75, no. 10, pp. 6303–6323, Oct. 2019.

[45] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang, "Adversarial sparse transformer for time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–11.

[46] G. T. Hicham, E. A. Chaker, and E. Lotfi, "Comparative study of neural networks algorithms for cloud computing CPU scheduling," *Int. J. Electr. Comput. Eng.*, vol. 7, no. 6, p. 3570, Dec. 2017.

[47] H. Shen and X. Hong, "Host load prediction with bi-directional long short-term memory in cloud computing," 2020, *arXiv:2007.15582*. [Online]. Available: http://arxiv.org/abs/2007.15582

[48] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values," *Transp. Res. C, Emerg. Technol.*, vol. 118, Sep. 2020, Art. no. 102674.

[49] Y. S. Patel, R. Jaiswal, S. Pandey, and R. Misra, "*k* stacked bidirectional LSTM for resource usage prediction in cloud data centers," in *Proc. Int. Conf. Internet Things Connected Technol.* Cham, Switzerland: Springer, 2020, pp. 147–157.

[50] S. Li, J. Bi, H. Yuan, M. Zhou, and J. Zhang, "Improved LSTM-based prediction method for highly variable workload and resources in clouds," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2020, pp. 1206–1211.

[51] S. Shen, V. Van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2015, pp. 465–474.

[52] S. Tofighy, A. A. Rahmanian, and M. Ghobaei-Arani, "An ensemble CPU load prediction algorithm using a Bayesian information criterion and smooth filters in a cloud computing environment," *Softw., Pract. Exper.*, vol. 48, no. 12, pp. 2257–2277, Dec. 2018.

[53] J. Hu and W. Zheng, "A deep learning model to effectively capture mutation information in multivariate time series prediction," *Knowl.-Based Syst.*, vol. 203, Sep. 2020, Art. no. 106139.

MD. EBTIDAUL KARIM was born in Chattogram, Bangladesh, in 1994. He received the B.Sc. degree in electronics and communication engineering from Khulna University of Engineering & Technology (KUET), Bangladesh, in 2018, where he is currently pursuing the M.Sc. degree with the Department of Electronics and Communication. He is currently working as a Lecturer with the Department of Electronics and Communication Engineering, KUET. He has authored or coauthored two conference papers. His research interests include cloud computing, antenna and microwave, deep learning, and computer vision.

MIRZA MOHD SHAHRIAR MASWOOD (Member, IEEE) received the B.Sc. degree in electronics and communication engineering from Khulna University of Engineering & Technology (KUET), Bangladesh, in 2010, and the M.Sc. degree in electrical engineering and the Ph.D. degree in telecommunications and computer networking from the University of Missouri-Kansas City (UMKC), USA, in 2015 and 2018, respectively. In 2011, he joined as a Lecturer with the Department of Electronics and Communication Engineering (ECE), KUET. During his Ph.D. degree at UMKC, he worked as a Graduate Research and Teaching Assistant. He is currently an Assistant Professor with the ECE Department, KUET. He reviewed more than 60 peer-reviewed conference papers and journal articles. He has authored or coauthored over 25 journals and conference papers. His research interests include data center optimization, traffic engineering, cloud computing, fog computing, and machine learning. He serves as a Reviewer for the *Journal of Network and Systems Management* and IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.

SUNANDA DAS received the B.Sc. Engineering degree in computer science and engineering (CSE) from Khulna University of Engineering & Technology (KUET), Khulna, Bangladesh, in 2018. He is currently working as a full-time Lecturer with the CSE Department, KUET. His research interests include machine learning, deep learning, artificial intelligence, natural language processing, and computer vision. He has published a couple of papers on these topics and actively working on various research projects with his students, colleagues, and collaborators.

ABDULLAH G. ALHARBI (Member, IEEE) received the B.Sc. degree in electronics and communications engineering from Qassim University, Saudi Arabia, in 2010, and the master's and Ph.D. degrees in electrical engineering from the University of Missouri-Kansas City, USA, in 2014 and 2017, respectively. From 2010 to 2012, he was an Electrical Engineer with Saudi Aramco Company. He is currently an Assistant Professor with the Electrical Engineering Department, Al-Jouf University, Saudi Arabia. He has authored or coauthored over 45 journals and conference papers and one Springer book chapter. His research interests include digital IC design, memristor-based circuits, traffic engineering, cloud computing, and network routing. He is a member of Gulf Engineering Union, the IEEE Circuits and Systems Society, the IEEE Young Professionals, the IEEE Signal Processing Society, the IEEE Instrumentation and Measurement Society Membership, and the IEEE Communications Society Membership; and a Professional Member of ACM. He was a recipient of several awards from Saudi Arabian Cultural Mission, USA, and the University of Missouri-Kansas City.