Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Host load prediction in cloud computing with Discrete Wavelet Transformation (DWT) and Bidirectional Gated Recurrent Unit (BiGRU) network

Javad Dogani, Farshad Khunjush [*], Mehdi Seydali

Department of Computer Science and Engineering and IT, School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran

A R T I C L E I N F O

Keywords:
Cloud computing
Host load prediction
Deep learning
Discrete Wavelet Transformation (DWT)
Bidirectional Gated-Recurrent Unit (BiGRU)

A B S T R A C T

Providing pay-as-you-go storage and computing services have contributed to the widespread adoption of cloud computing. Using virtualization technology, cloud service providers can execute several instances on a single physical server, maximizing resource utilization. A challenging issue in cloud data centers is that available resources are rarely fully utilized. The server utilization rate is poor and often below 30%. An accurate host workload prediction enhances resource allocation resulting in more efficient resource utilization. Recently, numerous methods based on deep learning for predicting cloud computing workload have been developed. An efficient strategy must predict long-term dependencies on nonstationary host workload data and be quick enough to respond to incoming requests. This study employs a Bidirectional Gated-Recurrent Unit (BiGRU), Discrete Wavelet Transformation (DWT), and an attention mechanism to improve the host load prediction accuracy. DWT is used to decompose input data into sub-bands with different frequencies and to extract patterns from nonlinear and nonstationary data in order to improve prediction accuracy. The extracted features are fed into BiGRu to predict future workload. The attention mechanism is used in order to extract the temporal correlation features. This hybrid model was evaluated with cluster data sets from Google and Alibaba. Experimental results reveal that our method improves prediction accuracy by 3% to 56% compared to a variety of state-of-the-art methods.

1. Introduction

Cloud computing indicates the on-demand accessibility of computer system resources, particularly data storage and computing resources, enabling the users to manage without direct intervention [1,2]. Organizations can rent cloud computing services as an alternative to investing in their own computing infrastructure or data centers. [3]. The various pay-as-you-go cloud services not only enable clients to purchase resources on-demand [4] but also enables the provision of an infinite amount of resource capacity (e.g., CPU, memory, network, and disk) at a reasonable price without investing in infrastructure or incurring additional expenditures for maintenance [5–7]. The average capacity utilization rate for regular deployments is less than 40%, although businesses require a relatively large number of servers and other resources to ensure the quality of service (QoS) during peak periods [8,9]. Fig. 1 depicts boxplots of CPU consumption over two working days for 50 Google cluster machines. Each record of this data represents the cumulative consumption in 5 min. As seen in Fig. 1, however, the average CPU utilization rarely exceeds 50%, and in most cases, it is less than 30%.

Virtualization technology enables cloud service providers to operate numerous virtual machine (VM) instances on a single physical server, resulting in resource utilization and increased return on investment [10]. The amount of resources required to service cloud applications is rarely constant and varies according to the volume of incoming requests. Predicting the number of required resources via analyzing the workload time series of each host prior to the arrival of requests enables more efficient scheduling and optimization of virtual machine migration. The provisioning of resources based on an accurate prediction prevents service level agreement (SLA) violations and enhances the QoS. In addition, resource provisioning and VM orchestration based on customer demand prediction results in optimal resource utilization, data center energy consumption reduction, and increased user satisfaction [11–16]. In addition, host workload prediction is essential to meet future user demands, improving the efficiency of cloud computing systems. Workload prediction plays a significant role in cloud resource provisioning because it lays the foundation for minimizing underutilization and resource waste, load balancing, energy consumption reduction, and preventing Service Level Agreement

[*] Correspondence to: Department of Computer Science and Engineering and IT, School of Electrical and Computer Engineering, Shiraz University, Mollasadara St., 71348-51154, Shiraz, Iran.

E-mail addresses: j.dogani@shirazu.ac.ir (J. Dogani), khunjush@shirazu.ac.ir (F. Khunjush), m.seydali@cse.shirazu.ac.ir (M. Seydali).

https://doi.org/10.1016/j.comcom.2022.11.018
Received 31 July 2022; Received in revised form 9 November 2022; Accepted 26 November 2022
Available online 30 November 2022

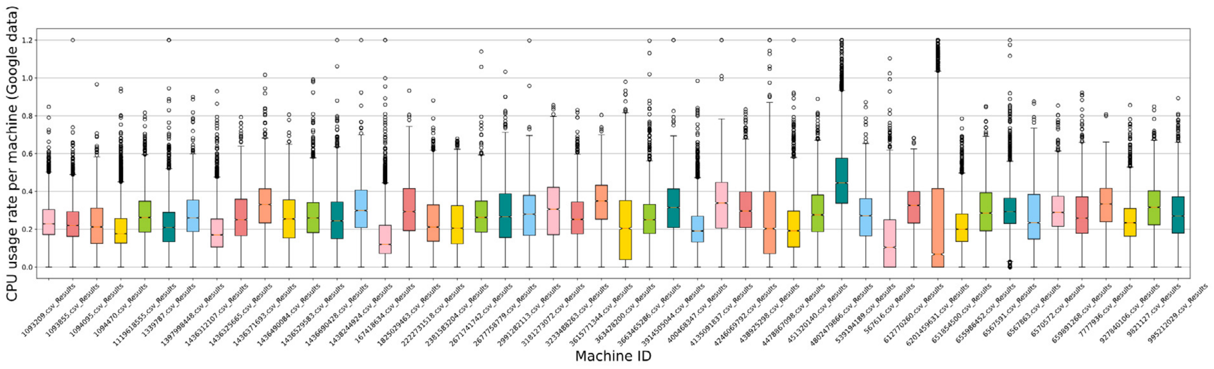

Fig. 1. The CPU usage for two full working days for the five machines from the Google cluster that have performed the most tasks.

(SLA) violations [3,17,18]. Today, data centers make extensive use of prediction models. As an example, PredictKube [19] is an open-source program created by Disney that uses artificial intelligence technologies to estimate workloads and then scale resources accordingly. Another example is Moneyball [20], which focuses on reducing this delay in resource availability by predicting the pause/resume patterns and proactively resuming resources for each Microsoft Azure SQL database.

Host workload traces in cloud data centers is represented as $S = \{x_1, x_2, \ldots\}$, which contains an infinite sequence of elements x_t, where x_t is a d-dimensional vector indicating the usage rate of a particular machine resource at time *t*. A time series is called stationary if its statistical features are constant over time. Stationary is widely known as a fixed correlation structure over time and periodic fluctuations. If these conditions do not exist in time-series data, time series show non-stationary characteristics, which is a significant challenge for prediction applications. The real-time CPU usage traces for the Google cluster machines in Fig. 1 demonstrate that host load data is nonstationary, with each machine exhibiting extremely rapid CPU utilization fluctuations as well as a time-dependent mean. The non-stationarity of cloud application workloads leads to fluctuating resource requirements and unpredictable overall demands generated by virtual machines (VMs) deployed on servers, impeding optimal resource allocation.

Workload prediction can be considered a time series prediction problem, where each workload at a specific time interval is related to workloads at the previous intervals [21]. Although statistic-based prediction methods achieve acceptable accuracy for predicting sequence-based problems, these methods assume the stationary nature of the collected data, which may contradict the dynamic nature of cloud environments. Recently, machine-learning prediction methods have been used for workload prediction, where their accuracy outperformed the statistic-based methods [22]. Unfortunately, those learning-based methods have considered workloads an independent collection of values. In order to predict nonstationary data accurately, the nonstationary data must first be decomposed into multiple stationary data. So, several decomposition methods have been developed to analyze nonstationary time series [23]. Among signal analysis techniques, experimental state analysis (EMD) and discrete wavelet transformation (DWT) have been widely used because of their effectiveness in processing nonlinear and nonstationary time series compared to other decomposition methods [24]. The primary benefit of DWT over other methods is its ability to decompose a signal into its frequency components directly and represent data in frequency domain distribution mode over time [25,26]. Before applying the prediction model, DWT is used to obtain higher prediction accuracy via extracting information about the past and future host workloads.

Large-scale workloads prediction models should be capable of making accurate and usable predictions. These models should be able to extract long-term dependencies in host load traces, deal with multidimensional data, have reasonable computational complexity, and extract patterns from nonlinear and nonstationary data. Unfortunately, conventional approaches for cloud computing host load prediction do not extract patterns from nonlinear and nonstationary data. The prediction might be conducted for other metrics such as memory, disk, and network. Our study, along with most prediction algorithms, has focused on CPU utilization since it is regarded as the principal cause of the QoS drop. Because the CPU serves as the most dynamic and fluctuating resource, it is one of the prominent sources of resource scarcity on clouds [27]. This study aims to improve resource utilization by predicting the CPU usage of cloud hosts using a deep learning approach. A recurrent neural network (RNN) is a deep neural network that uses historical data to predict future values [28]. One of the most popular RNN models is the Long Short-Term Memory Model (LSTM), used in previous studies [29,30] to predict host load in cloud computing data centers. Although LSTM-based methods can accurately model linear and nonlinear dependencies, they involve many computations that take a long time. The significant computational complexity of LSTM limits its deployment in resource allocation applications that need quick provisioning [31]. Previous research on LSTM-based predictions has not taken model training time into account. In order to address this issue, the Gated-Recurrent Unit (GRU), is used in this study, which is a simplified version of the LSTM network. GRU structure is more concise than LSTM, avoiding the problem of long updating time in the LSTM [32–34]. Since requests arrive quickly in real cloud environments, we need algorithms that can make the prediction quickly, so GRU is a better choice than other deep learning methods.

This article uses an extension of the GRU, the bidirectional GRU (BiGRU) model, to introduce host load prediction. The GRU calculates input only in the forward direction, whereas the BiGRU processes information in both the forward and backward directions [35]. This method combines a forward and a backward unit and provides additional gains over the one-directional GRU by enabling the extraction of dependencies between previous and future steps. In addition, We develop an attention mechanism for the BiGRU module. Due to the different effects of previous steps on the future host load, the attention-based layer can automatically exploit the various importance levels associated with a workload sequence at various points in time.

The proposed technique is divided into three stages. First, DWT [25] decomposes nonlinear and nonstationary host load traces into low and high-frequency sub-bands [26]. As a result, one approximate and multiple detailed signals are obtained to reduce the interaction within each signal. Second, each sub-band data is trained by the deep learning model, and the test values of each signal are predicted. Finally, an attention mechanism for the BiGRU module is deployed to automatically exploit different levels of dependencies in a workload sequence at different moments.

Fig. 2 shows the workload prediction structure of machines in cloud computing data centers. The user submits his resource usage request to the data center, and the prediction module predicts the future workload. The resource manager determines how to allocate resources based on the future workload prediction. Given the significance of

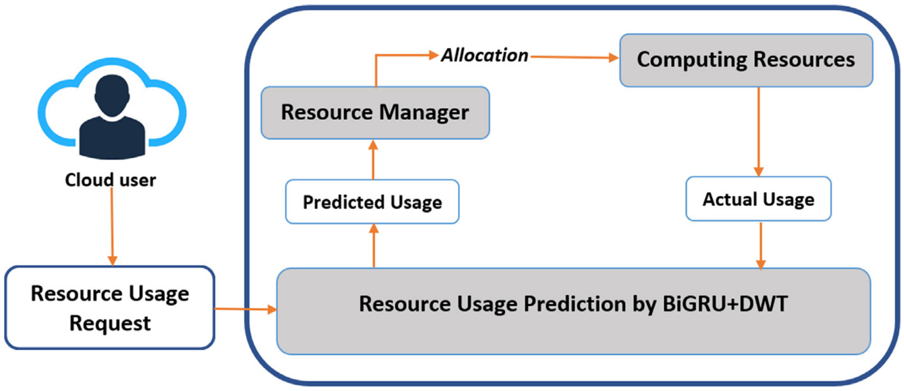

Fig. 2. Structure of host load prediction model in cloud data centers.

prediction accuracy for data center performance, this study focuses solely on predicting CPU usage in data centers and ignores scheduling and resource provisioning issues. The contribution of this article will be summarized as follows:

(1) A combination of DWT, BiGRU, and the attention mechanism is proposed to predict host workload data by learning linear and nonlinear dependencies. To the best of our knowledge, this is the first study to combine the DWT, the BiGRU model, and the attention mechanism to predict the host load in cloud computing.

(2) DWT is applied to extract patterns from nonlinear and nonstationary data so that BiGRU may successfully learn periodic and random fluctuations in the host load trace data.

(3) The proposed method aims to distinguish itself from existing deep learning algorithms that ignore the nonstationary features of cloud host load traces and the correlations between different sequences of host load trace data.

(4) The hybrid model is evaluated with real Google and Alibaba Cluster datasets. The experimental results indicate that the hybrid model has a lower prediction error.

2. Related work

In the past decade, machine learning techniques for predicting and transforming time series have attracted attention in scientific and industrial circles. These methods yield valuable observations regarding the behavior of various time series. One of the most important time series applications is the host workload in cloud computing data centers. We categorize the existing host workload prediction approaches as parametric, nonparametric models and deep learning models. The existing host workload prediction approaches are parametric, nonparametric, neural network-based, and deep learning models. Parametric approaches make broad assumptions regarding mapping input variables to output variables; hence, they are easier to train and require less data but may not be as effective. Nonparametric techniques involve few or no assumptions about the target function and, as a result, take a great deal more data, are slower to learn, and have a greater model complexity, but can produce more effective models.

The parametric models include learning methods such as the Kalman filter [36], the Autoregressive Integrated Moving Average (ARIMA) model [37], and ARIMA variants. The ARIMA-based model was presented in [38]. It uses ARIMA to predict cloud workload and illustrates a prediction module for SaaS providers based on real traces of web server requests. A hybrid method, EEMD-RT-ARIMA, was proposed in [39], in which a nonstationary host utilization sequence is decomposed into relatively intrinsic mode function (IMF) components. Three new parts are reconstructed by selecting appropriate IMF components. Finally, the ARIMA method predicts each new component and superimposes the results to make the final prediction. ARIMA-based models cannot capture the nonlinear patterns of cloud host load traces, although ARIMA accounts for the nonstationary nature of time series. Nonetheless, due to their constant variance, these models are insufficiently accurate for processing the stochastic nature of the host load, which is an essential characteristic of the host load.

Nonparametric approaches include the popular support vector regression (SVR) method and the neural network (NN)-based methods. SVR is widely used to solve linear time series problems [40,41]. The SVR does not depend on the size of the input vector space and may map the input vectors nonlinearly into a high-dimension feature space to create a linear decision level, enabling it to apply in high-dimensional space. A multi-step-ahead method was proposed in [42] based on SVR for CPU load prediction. The prediction error is reduced by Kalman smoothing technology. A multi-step-ahead host load prediction method called KSwSVR was proposed in [43]. KSwSVR is based on statistical learning theory and integrates an improved SVR algorithm and Kalman smoother. It can reduce resource consumption rates and meet SLA. A combination of three-level wavelet transform (WT) and SVR was proposed [44] to address both dynamic resource allocation and auto-scaling. First, it applies WT and decomposes the load traces into some signals using different frequency scales. Second, it uses SVR to predict each component. Finally, a novel chaotic particle swarm optimization (PSO) algorithm is used for tuning SVR parameters. In [45], a weighted wavelet support vector machine (WWSVM) method was proposed that combines the WT and SVR in order to combine their advantages. The importance of multiple sample points is considered by weighting the data, hence reducing prediction error. The study in [46] proposes a PSO-based algorithm to find the optimal parameters for SVR training. SVR is trained using historical resource utilization data and a hybrid kernel function that combines the radial basis function (RBF) and the polynomial kernel function. Nonparametric methods such as SVR are not able to use nonlinear features to predict host load traces. The SVR uses the mapping of large amounts of traffic flow data into a high-dimensional space, and the inability to extract temporal and periodic features leads to poor predictive performance.

Artificial neural networks (ANNs) have performed well in classification and prediction issues in cloud computing [47]. A method named RVLBPNN (Rand Variable Learning Rate Backpropagation Neural Network) was proposed in [48]. Further, a load prediction model is proposed to reduce cloud computing energy consumption based on the Backpropagation Neural Network (BPNN) algorithm. A hybrid wavelet neural network method was proposed in [49], which models changes in load traces. It uses two heuristic algorithms, the Artificial Immune System and the Water Cycle Algorithm, to enhance the training process and tune optimized wavelet neural network parameters. The ANN models have the drawback of the possibility of getting trapped in a local minimum, and the choice of model architecture is a challenging problem.

Early research on neural networks often used shallow or single-layer networks, which failed to represent the uncertainty and nonlinearity of host load traces. In recent years, deep learning methods have been used in various time-series prediction methods. Deep learning methods can extract inherent spatial and temporal features from data and do not need data preprocessing. *Sibyl* is a deep learning-based method for one-step-ahead prediction proposed in [50]. *Sibyl* has two modules. The first module selects metrics by filtering out irrelevant metrics, and the other trains a neural network. A powerful neural network model was applied, built with bidirectional LSTM to predict actual load. A method based on LSTM Encoder–Decoder (LSTM-ED) was proposed in [51]. It constructs an internal representation of host load trace data in order to use less memory than standard LSTM and make multi-step-ahead predictions over intervals and actual load. A novel method was proposed based on a combination of GRU and LSTM, named LSRU [52]. LSRU results are better than LSTM and GRU separately. LSRU is a short-term and long-term prediction, and it can face some sudden bursts of workload. The study [30] incorporates both bi-directional and grid LSTM (BG-LSTM). The results show that an integrated model combining multiple entirely different deep learning models can reduce the error of predictions result. In [53], an actual load multi-step-ahead prediction method was proposed to apply LSTM to predict the mean workload over future time intervals. Bidirectional LSTM (BiLSTM) was used for host load prediction in [17]. This method used the memory capability of LSTM and the modeling ability of LSTM Encoder–Decoder (LSTM-ED). In [54], the host load prediction issue was addressed by proposing a hybrid algorithm using the empirical mode decomposition (EMD) method. EMD extracts some components in different frequency bands from original cloud load traces. Also, a new ensemble architecture combining GAN and LSTM is deployed to predict each sub-band individually. Although LSTM is suitable for processing and predicting time series data among deep learning methods, the complex structure of the LSTM network prolongs training time. In previous studies that used LSTM, the training time of the model was not considered, making it impossible to use these studies in real applications.

In [55], a novel host load prediction method was proposed. An autoencoder was employed as the pre-recurrent feature layer. This method predicts the future host load based on the Google cluster usage dataset. In [56], a GRU-based Encoder–Decoder network (GRUED) was applied to achieve accurate prediction. This network contains two gated GRU and two data sets, including Google resources usage and the Unix system load traces used to evaluate the proposed method. To provide more accurate prediction results, the study in [57] presented the DP-CUPA algorithm, a hybrid model by PSO and deep belief network (DBN), to predict the CPU usage algorithm. An ensemble CPU model load prediction was proposed in [58], employing a Bayesian information criterion. The cloud resource usage history is evaluated, and the best constituent model is chosen for each time slot. A couple of smooth filters are employed to omit the negative impacts of outliers in the data points. In [30], a logarithmic procedure is used to lower the standard deviation before smoothing resource workload traces. The approach then uses a strong filter to reduce noise interference and extreme points. In addition, a Min-Max scaler is used to normalize the data. A deep learning integrated technique for time series prediction is developed. It integrates network models such as BiLSTM and grid LSTM networks to accomplish high-quality workload and resource time series prediction. In [59], an ensemble learning approach is given that employs extreme learning machines (ELM) and a voting mechanism to weight the prediction outcomes. It selects the appropriate weights using a metaheuristic method based on blackhole theory. As base experts, k multilayer neural networks are used in this method. To determine the weights of synaptic connections among hidden and output neurons, an ELM solves a general linear system.

The combination of the interactive temporal recurrent convolution network (ITRCN) and GRU model was applied in [35] to predict traffic for a single-service host. The convolution neural network (CNN) captures the correlations between services in a network by learning network traffic as images, and GRU discovers temporal features. Although GRU-based methods are faster than LSTM-based methods, using the GRU-based methods directly to learn and predict non-static machine workload data does not increase forecast accuracy. If the time series of cloud computing load data is broken down into several other predictable components with a lower degree of nonstationary, the prediction accuracy can be significantly increased. We employ a two-step method that combines DWT and BiGRU. DWT is used to split the host load traces into several sub-bands. Then, a prediction model based on BiGRU is developed to predict linear and nonlinear relationships in the host data load more efficiently than existing deep learning techniques.

3. Background

This section describes the tools that have been used in this study. These tools include discrete wavelet transform (DWT), gated recurrent unit (GRU), Bidirectional GRU (BiGRU), and attention mechanism.

3.1. Discrete wavelet transform (DWT)

DWT is a mathematical method that is appropriate for time scale analysis in time series [26]. DWT is an effective method for splitting nonstationary signals into sub-bands of different scales [60]. The DWT is a signal processing technique for nonstationary and nonlinear signals. At each time–frequency scale, the output of wavelets might resemble the primary signal. The output of the wavelets captures the characteristics of the original signal and is used to identify the trend of sequence changes in the analysis of nonstationary time series, such as workload traces data in cloud data centers. The DWT decomposes the original host load traces into a series with various frequencies. The decomposition process is done by Mallat's algorithm [61], which uses high-pass and low-pass filters and can efficiently decompose nonstationary host load traces into multiple sequences of different frequencies. As a result, both the low-pass and high-pass filter outputs contain sub-bands. These sub-bands are called approximate coefficients and detail coefficients that are defined by *dA* and *dD* in Eqs. (1) and (2) [26]:

$$dA = \sum_{k=-\infty}^{+\infty} X[k]\,\varphi_l[2n-k], \tag{1}$$

$$dD = \sum_{k=-\infty}^{+\infty} X[k]\,\varphi_h[2n-k], \tag{2}$$

X represents the original host load traces signal, φ is the filter, and l represents the low-pass, and h represents the high-pass filter. The Mallat algorithm process for three-level decomposition has been shown in Fig. 3. The original host load traces X are decomposed through low-pass and high-pass filters in the first layer. The decomposition process results in $d1$ and $a1$ that are detailed and approximate sub-bands. In the second layer, the obtained $d1$ sub-band is passed through two low-pass and high-pass filters again to obtain two coefficients $a2$ and $d2$. Similarly, in the third layer, the obtained $d2$ sub-band is passed through two low-pass and high-pass filters to obtain two coefficients, $a3$ and $d3$. After the decomposition process, some components with different frequencies can be obtained. After decomposition, the obtained parts are of different lengths. To have sequences of the same size, the data is reconstructed using approximate and detailed coefficients by applying the inverse discrete wavelet transform (IDWT). IDWT obtains sequences with the same length as the original host load sequence but with different frequencies.

3.2. GRU model

RNN is a type of neural network that uses previous outputs as inputs when a hidden state is available. These models use cyclic connections in their hidden layers to reduce memory requirements and extract information from time series and sequence data. RNN networks can record

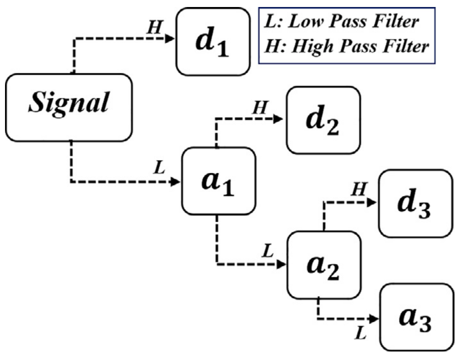

Fig. 3. A schematic diagram of DWT.

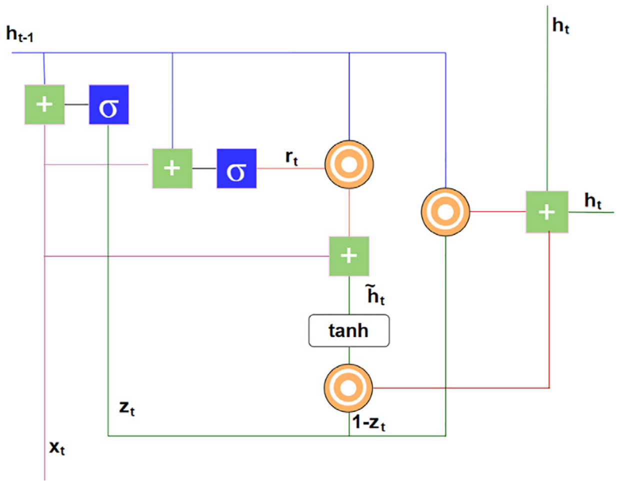

Fig. 4. GRU gate structure.

the information they currently see by employing memory variables. RNN is only able to store information that has been extracted a few steps prior. Since the output of each layer depends on the calculations of the previous layers, RNNs are known as recursive networks. RNNs are stacks of temporary copies of general neural networks, and each copy communicates information to each other. The RNN suffers from the well-known gradient vanishing problem, which prevents the model from learning the long-term dependencies, making it challenging to learn long-term dependencies. LSTM overcomes this problem by storing helpful information in the memory unit and removing unnecessary information. The primary proceeding is to replace the middle layer of RNN with an LSTM block. LSTM is well-known for learning long-term dependencies, which RNNs are not able to learn [62].

The GRU model as a simplified version of the LSTM, aims to provide a simpler and faster version of the recurrent network by combining the input and forget gates into one update gate. It includes only two reset gates and an update gate [63,64]. The update gate controls the effect of the output of the previously hidden layer at the previous time on the current hidden layer. The more significant value for the update gate leads to a more significant impact of the previously hidden layer's output on the current hidden layer. The reset gate mainly controls the combination of previous and current input information. The reset gate determines the neglection of the hidden layer information at the previous time, and the smaller value for the reset gate leads to ignorance of more details. The structure of the GRU gate is shown in Fig. 4.

Calculating the update gate z_t for time step t is the initial step. When x_t is inserted into the network unit, its weight W_z is multiplied by it. The same is done for h_{t-1}, which stores data for the past $t-1$ units and is multiplied by its own weight U_z. Both results are summed, and an activation function is employed to squash the result between 0 and 1 (Eq. (3)). The update gate aids the model in deciding how much historical data from earlier time steps must be transmitted to the future. This is incredibly effective since the model can decide to duplicate all historical data and remove the possibility of the vanishing gradient problem. The model uses the reset gate r t to determine how much previous information to forget. h_{t-1} and x_t is multiplied with their corresponding weights, sum the results and apply the activation function (Eq. (4)). $\tilde{h}_t$ is introduced as a new memory content that utilizes the reset gate to store historical information. Initially, x_t is multiplied by the weight W and h_{t-1} is multiplied by the weight U. Then, the element-wise product of the reset gate r_t and Uh_{t-1} is calculate. It will determine which time steps should be removed. The results of each step are then added together, and the nonlinear activation function *tanh* is applied (Eq. (5)). As the final step, the network calculates the h_t vector, which contains information about the current unit and transmits it to the network. In order to accomplish this, the update gate is required. It decides what to collect from the current memory content $\tilde{h}_t$ and what to acquire from the previous stages h_{t-1} based on z_t (Eq. (6)). The calculations of the z_t, r_t, $\tilde{h}_t$, *and* h_t are shown in Eqs. (3) to (6).

$$z_t = \sigma\left(W_z x_t + U_z h_{t-1}\right) \tag{3}$$

$$r_t = \sigma\left(W_r x_t + U_r h_{t-1}\right) \tag{4}$$

$$\tilde{h}_t = tanh\left(W x_t + r_t \otimes U h_{t-1}\right) \tag{5}$$

$$h_t = \left(1 - z_t\right) \otimes h_{t-1} + z_t \otimes \tilde{h}_t \tag{6}$$

3.3. BiGRU model

In a time series, both past and future records could have an effect on the current record [65]. The fundamental unit of BiGRU is constructed of two GRU units: one for forwarding propagation and one for backward propagation. BiGRU employs GRU units and separates conventional GRU neurons into forward and backward states for learning future data (positive time direction) and past data (negative time direction) [63]. As a result, BiGRU may learn from both previous and future data points and gain a better knowledge of current data points during the training phase, resulting in a lower error and higher accuracy during the prediction phase. Two hidden layers are connected in the opposite direction and share the same output layer. The output layer receives information about previous and future states. The general structure of BiGRU is represented in Fig. 5.

BiGRU has comprised of two primary components: (1) a forward computational component that updates the hidden status based on historical host load data, and (2) a backward computational component that updates the hidden status based on future host load data. BiGRU is used to extract features representing the interaction between input instances to predict the workload over time. In particular, the BiGRU network extracts features highly dependent on the input time series data and employs both forward and backward computational methods. The extracted features are sent to the fully connected layer in the subsequent phase. The output layer is then utilized to predict the future host load.

3.4. BiGRU encoder–decoder by attention mechanism

The BiGRU encoder–decoder architecture is based on a learning model and now has been applied as the state-of-the-art sequence prediction architecture. Two learning networks, encoder and decoder, read and generate variant-length sequences. An attention mechanism [66] is composed of an encoder–decoder structure. The encoder generates an attention vector from the input and then passes it to the decoder. The decoder takes the output of the encoder as an input and generates a hidden state.in our work, The encoder and decoder components are BiGRU. The encoder component predicts multi-step future data based on

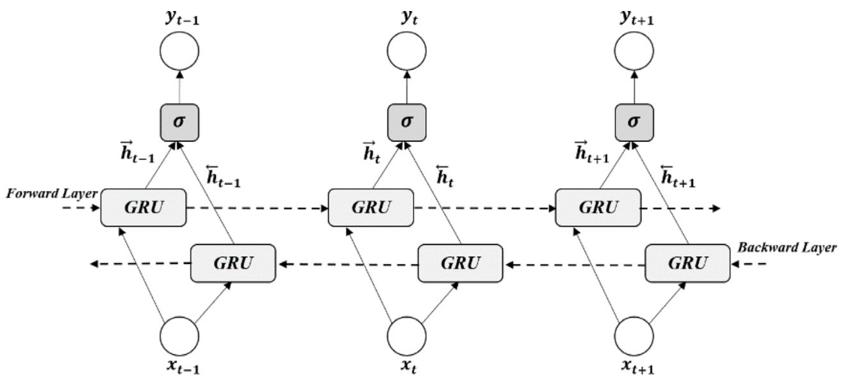

Fig. 5. General structure of BiGRU.

the preceding temporal context and historical data. The responsibility of the BiGRU encoder is to encode the input data in order to generate the context vector (CT). A CT is a fixed-length vector that serves as a temporal representation for the input data. The BiGRU then decodes the CT and generates a prediction. The following formula is used to determine the probability of the prediction sequence:

$$p(w_1, w_2, \dots, w_\Delta | X_1, X_2, \dots, X_T) = \prod_{t=1}^{\Delta} p(f_t | C_T, w_1, w_2, \dots, w_{t-1}) \tag{7}$$

where $(X_1, X_2, \dots, X_T)$ are the input features. We employ a classical encoder–decoder [60] that can be used to model web application workload traces and perform a multi-step prediction. In the encoder–decoder architecture, the encoder part compresses all the hidden representations of historical information into a CT vector. The temporal attention layer acts as an interface between the encoding and the decoding phases. BiGRU is used as an encoder that takes a time series $x = (x_1, x_2, \dots, x_T)$ as input and maintains the hidden internal state of h. In each step t, the GRU reads x_t and updates the hidden mode h_t as follows:

$$h_t = BiGRU\left(x_t, h_{t-1}\right) \tag{8}$$

Then, based on the BiGRU output, the temporal context vector C_i is generated in the ith decoding step as the total weight of the hidden states of the encoder network, which are considered the weighted sum of the hidden states of the encoder network. These vectors are used to select the best encoder hidden representation frames that map decoder attention to these frames. The process of calculating the temporal context attention vector C_i is:

$$C_i = \sum_{t=1}^{T} \alpha_{it} h_t \tag{9}$$

The weight α_{it} of each hidden state h_t is calculated as follows:

$$\alpha_{it} = \frac{\exp\left(\theta_{it}\right)}{\sum_{k=1}^{T} \exp\left(\theta_{ik}\right)} \tag{10}$$

$$\theta_{it} = a(S_{i-1}, h_j) \tag{11}$$

θ_{it} denotes the correlation between the output value at position i and the input value at position t, indicating the scoring function used to compute the correlation value. In our approach, general global attention is used as the scoring function, which is determined using Eq. (12) [66]:

$$\theta_{it} = S_{i-1} W M_a h_t \tag{12}$$

Where WM_a is the weight matrix of the scoring function. S_{i-1} and h_t indicate the hidden state of the decoder layer and the hidden state of the encryption layer, respectively. As shown in the above formulas, Eq. (11) represents the global general attention calculation between S_{i-1} and h_t. Attention weights are calculated using Eq. (9), which are related to the significance of the time series at time t and are used for predicting the output at time i. The vector e_{it} has The length of T and is used as the attention mask over the input workload time series. C_i is the last state of the attention layer. The softmax function is used for e_{it} vector normalization. The encoder and the decoder are trained together to maximize the log-likelihood output sequence, which are:

$$\underset{\theta}{argmin}\, C = -\sum_{i=1}^{n} \log p\left(w_i | X_i; \theta\right) + \lambda \|\theta^2\| \tag{13}$$

where n denotes the size of the training dataset, λ determines the significance of the penalty or the regularity of the loss function, and θ is the model parameter, including W and b of each layer. The model uses the mean square error (MSE) as a loss function of the training process. The early stop is utilized to terminate the training process when the validation loss no longer decreases to avoid overfitting.

4. Proposed method

This study describes a hybrid strategy for host workload prediction in cloud computing using a combination of DW, BiGRU, and attention mechanisms. Its novel contribution combines the data filtering method DWT with the attention-based BiGRU network to provide a strategy for host workload prediction that is both accurate and efficient. This section discusses the proposed method, DWT-BiGRU-attention. BiGRU is superior to other learning methods such as BPNN and SVR for learning nonstationary data and capturing the dynamic and nonlinear nature of time series data. However, when applied to nonstationary data such as cloud computing workload data, the prediction accuracy of BIGRU is low.

Due to the dynamic properties of cloud computing, the input signal is comprised of some components with different frequencies. These components are low-frequency and high-frequency components, and their distributions are different. Each component with different frequencies is independently trained by a BiGRU, which improves the training model performance. Therefore, an appropriate decomposition algorithm is required to decompose the host load traces data into low-frequency and high-frequency signals. DWT algorithms have multi-scale resolution and time-shifting characteristics and are used to parse the original input data and use the time scale function for analysis. By splitting the time series of cloud computing load data into several other predictable components with a reduced degree of non-stationarity, DWT significantly improves the accuracy of predictions. The pseudocode of the proposed method is illustrated in Fig. 6. The DWT method decomposes the host data traces into several predictable sub-bands. This decomposition effectively separates the high-frequency part of the primary host load traced signal due to random load changes and oscillations. It solves irregular fluctuations in the host workload sequence in data centers. Considering an input train set containing the host load sequence time series, it has m data points. We use The Mallat algorithm process for three-level decomposition. The original host load

```
The pseudocode of the proposed method
Require: $D$, a dataset containing host workload data input features $X$ and output feature $y$
Require: $P_{sets}$, set of hyperparameters with different values
Require: $M$, a single prediction model by $DWT\text{-}BiGRU\text{-}attention$
1  Split $D$ into three parts, $D^{train}$, $D^{validation}$, $D^{test}$
2  for $i = 1$ to $|D^{train}|$ do
3      read each training record
4      Apply $DWT$ to extract three detailed ($d1$, $d2$, and $d3$) and one approximate ($a1$) sub-band
5      A matrix $X[R * |D^{train}|]$ is employed to store all coefficients
6  end
7  for each $p$ in $P_{sets}$ do:
8      Train $M$ on $D^{train}$ with hyperparameters set $p$ to predict output feature
9      The result of $M$ is denormalized for each sub-bands.
10     A linear combination is used to fuse the $M$ results for each sub-bands, and the final prediction
is generated by averaging the results.
11     Validate learning model performance by applying it on $D^{validation}$
12     $Accuracy$ =Calculate accuracy of $M$
13     While Improved:
14             $Improved$=FALSE
15             Train M on $D^{train}$ with hyperparameters set $p$ to predict output feature
16             Validate learning model performance by applying it on $D^{validation}$
17             $Accuracy$ =Calculate accuracy of $M$
18             if $Accuracy < bestAccuracy$:
19                     $bestAccuracy = Accuracy$
20                     $Improved$=True
21             end if
22     end
23 end
24 Select optimal hyperparameter set $p^*$ from $P_{sets}$ where $MAPE$ is best
25 Train $M$ on $D^{train}$ with hyperparameters set $p^*$ to predict output feature
26 Compute $MAPE$ for $M$ with $D^{test}$
```

Fig. 6. Pseudo Code of the proposed method.

traces X are decomposed through low-pass and high-pass filters. As shown in Fig. 7, The decomposition process results in three detailed sub-bands ($d1$, $d2$, and $d3$) and one approximate sub-band ($a1$). Since the approximate and detailed sub-bands values are different, using it directly is not suitable for training BiGRU. Therefore, we use the Z-score normalization method [67] to normalize host load traces.

Fig. 7 illustrates the process of the proposed method. As depicted in Fig. 7, DWT is utilized to extract patterns from nonlinear and nonstationary data, after which BiGRU encoder–decoder by Attention Mechanism can effectively learn periodic and random fluctuations-related features in the host workload data. The encoder recursively inputs the cloud workload sequence and updates the cell memory state vector and hidden state vector at each time step. The encoder summarizes the whole input sequence into the final vectors. The encoder treats the final cell memory state vector and hidden state vector as the summarization and output of the input sequence and passes these two vectors to the decoder as the initial input of the decoder. This strategy may lead to information loss of input sequence as all information is summarized to the final cell memory state and hidden state. To resolve this issue, we employ an attention-based BiGRU encoder–decoder network with the temporal attention mechanism to select parts of hidden states that are highly related to the target location across all time steps of the encoder, instead of the final states. Two elements comprise the prediction model: a BiGRU encoder–decoder network with an attention mechanism and an output layer. The encoder starts by receiving the workload sequence data and encoding it into a context vector. The decoder produces intermediate predicted results for the output layer at each iteration. As the final output of the model, the output layer returns the workload prediction values. Temporal dependencies are related to intrinsic time patterns, which means that a value is dependent on its previous values in a time series. Variations in incoming workload are influenced by a periodic temporal pattern corresponding to the human workday. It indicates that the demand for resources at one data point may be highly comparable to another data point at a different period, depending on the day of the week, the season, and the time of day. Bi-GRU is used to learn the hidden representation of input workload data. BiGRU explores dependencies and extracts temporal features from resource usage traces.

In a large cloud computing environment, computation-intensive tasks are usually divided into multiple sub-components known as batch workloads. In batch workloads, execution of succeeding subparts must await the completion of preceding subparts. In such a scenario, each preceding workload step has a distinct effect on the current task. For example, peak workloads and initial workloads of the latter subparts may have significant effects, whereas bottom workloads may have only a minor effect. The historical workload sequence must be included when modeling the relationships between the present time step and its context. Integrated with the temporal attention mechanism, the BiGRU encoder–decoder model captures typical workload patterns more effectively. Different subsequences may be related to specific

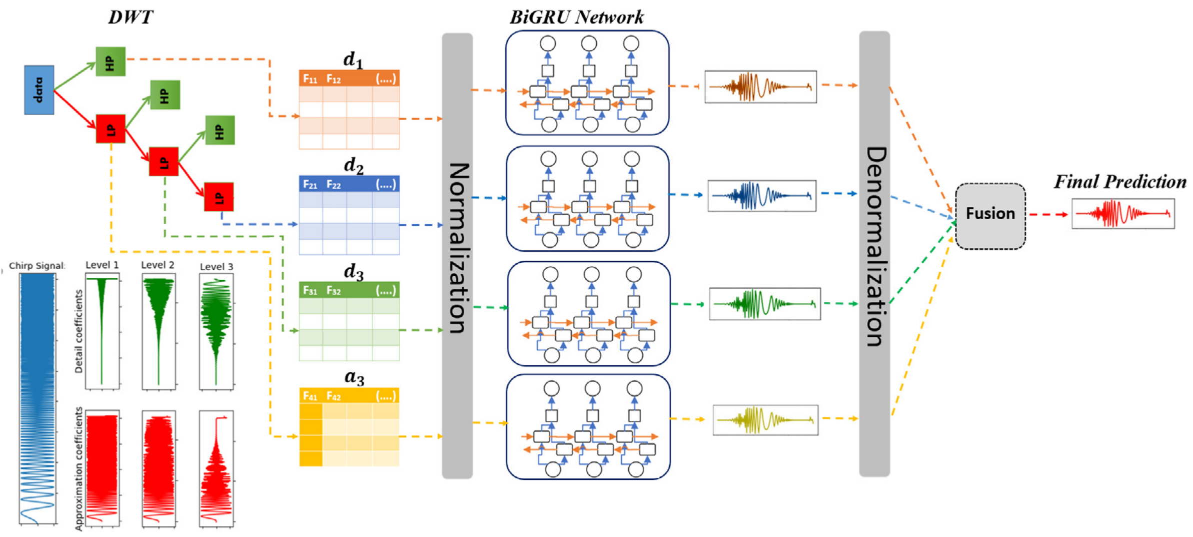

Fig. 7. The process of our method.

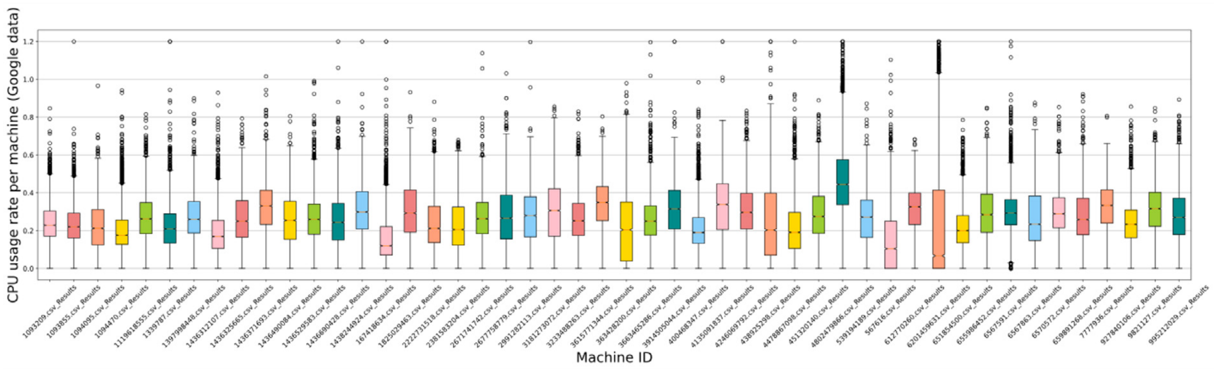

Fig. 8. The boxplot of 5-min host load data for 50 randomly selected devices from the Google Cluster dataset.

timesteps. During each prediction, the temporal attention mechanism adaptively gives greater weight to more strongly related subsequences. After training the BiGRU encoder–decoder by attention Mechanism, the test data set is utilized to predict each sub-band. The predicted values of a sub-band are denormalized. Each BiGRU's output is merged to determine the final prediction. The linear combination and the average of prediction results are considered for final prediction outcomes. Finally, the host load prediction results are obtained by comparing the original data to the denormalized predicted values.

5. Expriments and discussion

5.1. Data set

To evaluate the performance of the proposed method, two actual data center host load traces were used. The first is the host load traces in the Google data center [68], and the second is the Alibaba cluster dataset [69].

Google Cluster Dataset: This is a record of the workloads executing on eight Google Borg compute clusters in May of 2019. The trace details every task submission, scheduling decision, and resource utilization data for jobs that ran within those clusters. Google Cluster monitored the resource usage of more than 12,500 physical computers over 29 days, recording 67,2074 jobs and over 26 million records. These measurements were taken every five minutes. The task_usage table indicates the number of available resources consumed by each task. Each table entry contains twenty fields, such as the average and maximum CPU usage, the amount of memory allocated for each task, task ID, device ID, unmapped and total page cache usage, maximum input and output time, disk usage, etc.

Our model is trained using the data from the first 18 days. 19 to 25 days are used for the verification process. Verification is essential for selecting appropriate parameters and avoiding overfitting. The remainder consists of the test suite used to model evaluation. As a dataset, we randomly chose 50 machines. Fig. 8 shows a boxplot of 5-minute host load data for 50 randomly selected machines from the Google Cluster dataset. To demonstrate the precision of our method, we selected four unique computers and manipulated their CPU utilization and variances. Four machines, G1, G2, G3, and G4, are selected. G1 with ID 1438244924 was chosen due to its low average CPU utilization and high volatility. G2 with id 4246069792 was chosen due to its high average CPU utilization and moderate volatility. G3 with ID 1436371693 was selected because of its high average CPU consumption and moderate volatility, whereas G4 with ID 451320140 was selected because of its high average CPU consumption and high volatility.

Alibaba Cluster Dataset: The second database is the publicly available Alibaba Cluster Log, which is published by the Cluster Trace Program of the Alibaba Group. color-6The dataset records the performance of over 1300 machines over a 12-hour period. These traces record CPU, memory, and disk usage parameters at 5-minute intervals for all machines. For the purpose of simplicity, we predict CPU usage. The average CPU utilization of machines within the Alibaba dataset is approximately 26%, with a standard deviation of 10%. Fig. 9 is a boxplot of the Cpu utilization of 100 randomly selected machines from a number of 1312 available machines.

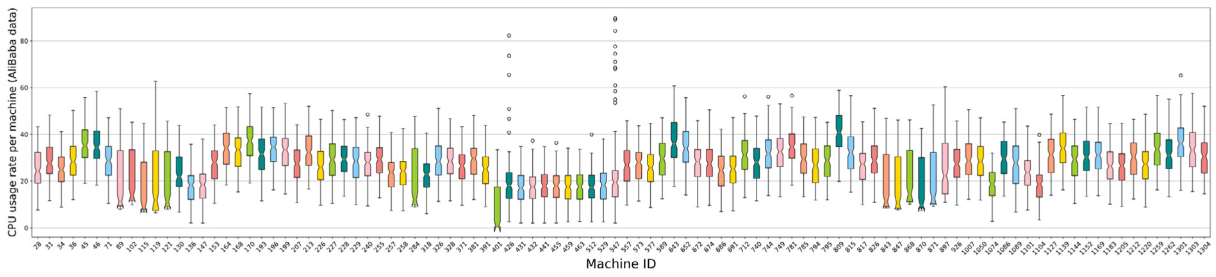

Fig. 9. The boxplot of 100 machines that were randomly selected from 1312 available machines of the Alibaba dataset.

Table 1
Parameters of our BiGRU-based model.

Parameter	Value
Optimizer	Adam
Loss function	MSE
Learning rate	0.01
$\|w\|$	28
Input layer size	24/64
Hidden layer size	128
Batch size	64
Epoch	60

Given that applications in data centers may have different usage patterns, as described in [26], we selected four machines, A1, A2, A3, and A4, from a subset of 100 randomly selected machines with varying CPU usage requirements in order to evaluate diverse CPU demands. The A1 (Machine ID 809) serves requests requiring considerable CPU resources and changing requirements. The A2 (Machine ID 431) satisfies CPU-intensive but slightly varying requirements. A3 (Machine ID 119) meets the need for considerable changes in CPU usage at different times, whereas A4 (Machine ID 170) meets the requirement for small fluctuations in CPU usage at different times.

5.2. Configuration

In this experiment, we evaluate our prediction model using BiGRU, and compare it with other time series prediction methods, including the SVR, BPNN, and LSTM. Table 1 lists the parameters chosen by the validation set. The BiGRU is trained based on the Adam optimizer [70], which has been proven to work well. The learning rate is set to 0.01, the number of hidden units is set to 128, and the loss function is the Mean Squared Error (MSE). The dropout layer is used to prevent overfitting. Numerous experiments are required to determine the optimal model parameters. To obtain |w|, we performed a grid search on |w| = {12, 20, 28, 36, 44, 52, 60}, which yields a performance of 28 on the validation set. The window length |w| and batch size parameters have been chosen using the grid search approach. We also perform a grid search for batch sizes = {16, 32, 64, 128, and 256}, with 64 demonstrating the highest performance. The parameter setting of the LSTM and BPNN is exactly the same as the BiGRU, whereas details of SVR are described as follows. For the SVR model, the kernel function is the Radial Basis Function (RBF), the optimal cost parameter c and the width parameter g are determined in numerous experiments. Before training, z-normalization is used to preprocess input data.

5.3. Evaluation metrics

Three widely perceived criteria were employed to measure prediction error in this study, and a statistical test was used to evaluate the prediction results and the scientific performance. Each of these criteria is discussed in more detail below.

Typical Evaluation Metrics

We use Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) to evaluate performance. RMSE squared is the square root of the deviation from the predicted and actual values relative to the number of predictions.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{14}$$

$$RMSE = \left[\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2\right]^{\frac{1}{2}} \tag{15}$$

$$MAPE = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{|y_i|} \tag{16}$$

where n indicates the test size. The predicted value is denoted by y_i, whereas the real value is denoted by $\hat{y}_i$. The MAE (Eq. (14)) determines the mean absolute error between predicted and actual data. Because the MAE is particularly sensitive to noises, the root mean square error (RMSE) (Eq. (15)) provides a more accurate representation of the result. Additionally, because the deviation is determined using the absolute value and therefore avoids compensating for both positive and negative values, it may accurately depict the status of the prediction error. (Eq. (16)) is the average percentage used to determine the absolute rate of variance between the predicted and actual values used to evaluate the performance of different models.

Statistical Analysis Metric

The Diebolde–Mariano test (DM) [71] is used to demonstrate the effectiveness of our model. DM was used to determine how the proposed model differed significantly from the baseline model in terms of prediction results. Diebold and Mariano [71] provide comprehensive descriptions of the DM test. The null hypothesis is that the two predictions are equivalent in terms of accuracy. The alternative hypothesis can be specified as either both predictions having different accuracy or one prediction being more accurate than the other prediction methods.

5.4. Evaluation

We implemented the proposed model using the Keras library in python programming language and the Tensorflow framework. The tests were conducted on a machine powered by an Intel Core i7-6500 CPU and an Nvidia GeForce 920M graphics card. On Google cluster data, the proposed model requires 21.2 s to train, whereas the training time for the Alibaba cluster is 12.6 s. As the Google and Alibaba clusters utilize considerably more powerful machines than the one employed in this work, the execution time in real applications will be significantly less than the obtained values, enabling the implementation of our method in real-world applications. To assess the effectiveness of our model, we compared it to other baseline and state-of-the-art models. Table 2 gives a detailed description of the various baseline models. First, we evaluate the performance of our technique, called DWT-BiGRU-attention, using nine algorithms: DWT-LSTM-attention, DWT-BiGRU, DWT-LSTM, DWT-BPNN, DWT-SVR, BiGRU, LSTM, BPNN, and

Table 2
Description of the compared models.

Category	Name	Description
Baseline methods	*SVR*	SVR is a method for mapping nonlinear input vectors into high-dimensional feature space in order to generate a linear decision plan.
	BPNN	BPNN is a neural network-based learning method based on a reduction gradient. The gradient of the error function is determined in this method concerning the weight of its neural network.
	LSTM	A recurrent neural network (RNN) architecture used in deep learning is an expert at processing time-series data.
	BiGRU	The model employs BiGRU to predict CPU usage using host workload time series.
DWT based methods	*DWT-SVR*	The model employs DWT for data decomposition and SVR to predict CPU usage using host workload time series.
	DWT-BPNN	The model employs DWT for data decomposition and BPNN to predict CPU usage using host workload time series.
	DWT-LSTM	The model employs DWT for data decomposition and LSTM to predict CPU usage using host workload time series.
	DWT-BiGRU	The model employs DWT for data decomposition and BiGRU to predict CPU usage using host workload time series.
Attention-based methods	*DWT-LSTM-attention*	The model employs DWT for data decomposition and LSTM and attention mechanism to predict CPU usage using host workload time series.
	DWT-BiGRU-attention	The model employs DWT for data decomposition and BiGRU and attention mechanism to predict CPU usage using host workload time series.

Table 3
The results for Google cluster host load.

	Machine G1			Machine G2			Machine G3			Machine G4		
	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE
SVR	34.2	10.2	8.4	22.3	6.4	8.0	21.6	13.0	10.9	24.1	14.3	9.9
BPNN	29.1	6.9	7.8	18.0	6.0	5.5	19.1	9.8	7.8	21.3	13.5	9.6
LSTM	23.1	6.5	4.8	15.8	3.3	5.0	15.6	7.4	5.7	17.7	8.4	5.8
BiGRU	20.5	5.7	4.1	14.7	3.1	4.6	15.2	6.9	5.3	16.4	8.2	5.5
DWT-SVR	28.2	9.2	7.3	20.1	4.6	6.6	19.2	11.1	8.8	21.5	9.6	6.8
DWT-BPNN	24.0	6.8	5.1	17.4	4.4	5.3	17.3	7.9	6.3	20.6	9.7	7.0
DWT-LSTM	18.1	5.0	3.7	13.6	1.8	4.4	14.2	6.5	5.3	16.4	7.0	4.4
DWT-BiGRU	17.6	4.1	3.4	12.8	1.5	3.8	13.3	5.9	5.0	15.2	6.9	4.3
DWT-LSTM-attention	16.4	3.9	3.2	11.3	1.4	3.5	11.8	5.3	4.8	14.7	6.3	3.9
DWT-BiGRU-attention	**15.4**	**3.1**	**2.8**	**10.5**	**1.1**	**3.0**	**10.9**	**4.8**	**3.9**	**13.3**	**5.6**	**3.2**

SVR. The comparative algorithms are divided into three groups. Group one includes baseline algorithms, including SVR, BPNN, LSTM and BiGRU. To assess the efficiency of the DWT method's decomposition of host load data traces, comparative techniques incorporating the DWT, such as DWT-LSTM, DWT-SVR, and DWT-BPNN, are included in the second group. The second set of prediction algorithms uses DWT operations, whereas the first group uses baseline techniques. Group 1 was utilized to determine the optimal LSTM, SVR, and BPNN neural networks for the time series prediction task, whereas Group 2 illustrates the influence of a DWT modification on prediction accuracy. Group 3 illustrates the influence of an attention mechanism on prediction accuracy. In these experiments, workload time series from four Google machines (G1, G2, G3, and G4) and four Alibaba machines (A1, A2, A3, and A4) are used as model input, whereas CPU consumption predictions for the next step ahead are used as output.

5.4.1. Google cluster dataset

Four Google Cluster machines were used for prediction operations, and the MAPE, RMSE, and MAE results are given in Table 3. The bold font in Table 3 shows the optimal values for each metric.

As seen in Table 3, DWT-BiGRU-attention has a higher prediction accuracy than the other approaches studied. Comparing DWT-BiGRU-attention to various prediction algorithms reveals that DWT-BiGRU-attention has significantly reduced MAPE, RMSE, and MAE, as a result of the use of DWT, Bidirectional learning of BiGRU, and attention mechanism. DWT enables prediction models to learn nonstationary features and capture the dynamic and nonlinear nature of time series data. The bidirectional nature of BiGRU enables the prediction model to learn information from two past and future directions and to learn the relationship between the different data points in historical data during the training phase. The attention mechanism enables the prediction model to analyze the historical data relationships and extract temporal dependencies to modify the weights accordingly. Fig. 10 compares actual and predicted CPU use for baseline and DWT-based approaches using Google Cluster data for four selected machines. Fig. 11 compares actual and predicted CPU usage for DWT-based approaches with DWT-attention-based methods using Google Cluster data for four selected machines.

We standardized the evaluation metrics criteria values to compare our technique error to other methods. Fig. 12 depicts the normalized values for RMSE, MAE, and MAPE. The results indicate that estimation of CPU consumption in the Google Cluster Database test set using RMSE and MAE criteria outperforms basic approaches. Fig. 13 illustrates a boxplot representation of the absolute deviation of CPU usage for each prediction algorithm using the Google cluster dataset. We observed that the DWT-BiGRU technique is superior in absolute error reduction.

Table 4 demonstrates the results of the DM test of each baseline method using the Google cluster dataset. As seen in Table 4, all of the *DM* test results for all prediction horizons exceed the upper limit of the 1% significance threshold. The prediction accuracy of our model varies significantly from other models.

Following is a comparison of the proposed method with numerous previous studies using the Google cluster dataset. The methods provided in [30,46], and [54] are utilized for these purposes. Since these studies did not specify which Google cluster machine data they were tested on, we implemented these approaches and evaluated them on the same machines as the proposed method.

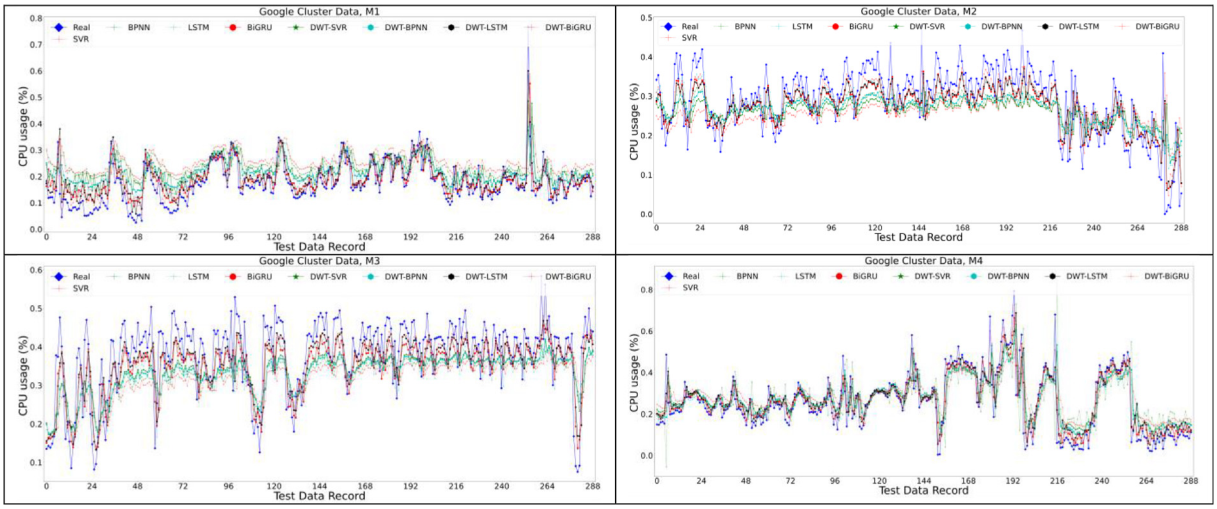

Fig. 10. Comparison of actual and predicted CPU usage for baseline and DWT based methods with Google Cluster data for four selected machines.

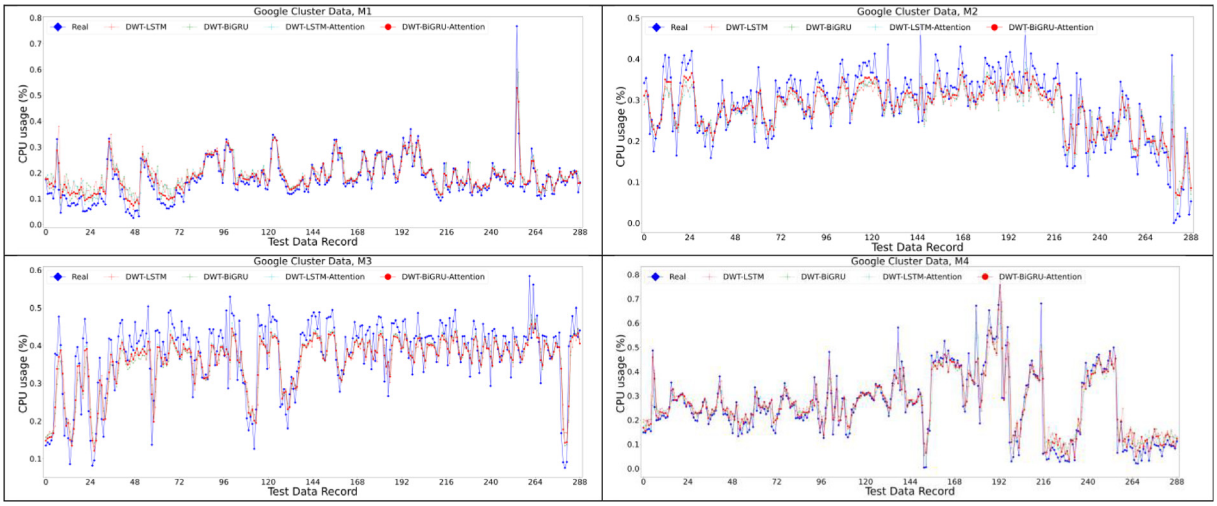

Fig. 11. Comparison of actual and predicted CPU usage for baseline and DWT-based methods with Google Cluster data for four selected machines.

Table 4
DM test results for Google cluster machines.

Model	DM value (significance level)			
	M1	M2	M3	M4
SVR	8.367 (1%)	7.254 (1%)	6.665 (1%)	7.254 (1%)
BPNN	7.824 (1%)	5.520 (1%)	7.447 (1%)	5.520 (1%)
LSTM	6.881 (1%)	5.798 (1%)	5.559 (1%)	5.798 (1%)
BiGRU	7.796 (1%)	7.413 (1%)	4.212 (1%)	7.413 (1%)
DWT-SVR	4.058 (1%)	4.872 (1%)	3.259 (1%)	4.872 (1%)
DWT-BPNN	3.997 (1%)	3.075 (1%)	2.733 (1%)	3.075 (1%)
DWT-LSTM	3.808 (1%)	2.769 (1%)	2.381 (1%)	2.769 (1%)
DWT-BiGRU	4.058 (1%)	4.872 (1%)	3.259 (1%)	4.872 (1%)
DWT-LSTM-attention	3.997 (1%)	3.075 (1%)	2.733 (1%)	3.075 (1%)
DWT-BiGRU-attention	3.808 (1%)	2.769 (1%)	2.381 (1%)	2.769 (1%)

According to Table 5, DWT-BiGRU-attention has a higher accuracy than the other studied. Comparing DWT-BiGRU-attention to other workload prediction studies reveals that the combination of DWT, BiGRU, and attention mechanism considerably reduces MAPE, RMSE, and MAE. Table 5 demonstrates the error metric of different studies using the Google cluster dataset.

5.4.2. *Alibaba cluster data*

This section describes the outcomes of applying the proposed method to four hosts from the Alibaba cluster dataset. Table 6 displays the prediction error for several criteria. Our method has the lowest error rate for predicting CPU utilization for various host workload patterns. As demonstrated in Table 5, DWT-BiGRU-attention has a higher prediction accuracy than the other techniques in the Alibaba cluster dataset. The experimental results indicate that the MAE, MAPE, and RMSE of the DWT-BiGRU-attention approach are lower than other methods.

As demonstrated by Table 6, DWT-BiGRU-attention has a higher prediction accuracy than the other methods examined. The comparison of DWT-BiGRU-attention and other prediction techniques indicates that DWT-BiGRU has a considerably lower MAPE, RMSE and MAE. Fig. 14 compares actual and predicted CPU use for baseline and DWT-based approaches using Alibab Cluster data for four selected machines. Fig. 15 compares actual and predicted CPU usage for DWT-based

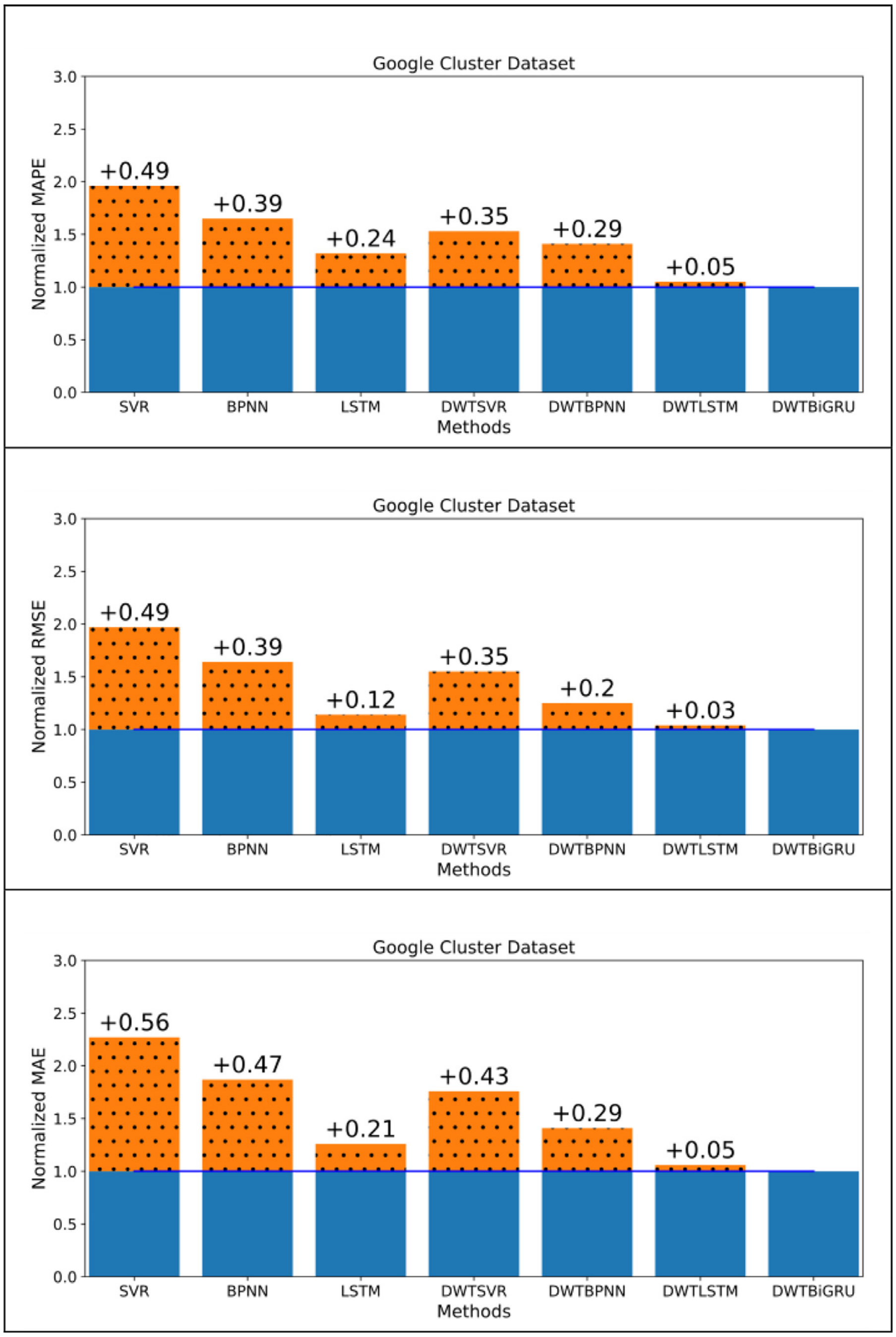

Fig. 12. Normalized Metrics Comparison for different methods using Google Cluster dataset.

Table 5
The comparison of errors metrics in different studies using Google cluster host load.

	Machine G1			Machine G2			Machine G3			Machine G4		
	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE
Kernel based SVR [46]	21.3	7.9	5.9	15.3	3.7	5.2	14.8	7.9	6.1	13.2	8.1	6.1
BG-LSTM [30]	17.2	4.6	3.5	13.1	1.7	4.0	13.8	6.3	4.8	15.7	6.9	4.2
E2LG [54]	16.7	3.7	3.1	12.2	1.4	3.3	12.7	5.4	4.5	14.4	6.2	3.8
DWT-BiGRU-attention	**15.4**	**3.1**	**2.8**	**10.5**	**1.1**	**3.0**	**10.9**	**4.8**	**3.9**	**13.3**	**5.6**	**3.2**

and DWT-attention-based methods with Alibab Cluster data for four selected machines. Figs. 14 and 15 prove the efficiency of the DWT decomposition algorithm and attention mechanism, respectively.

Figs. 14 and 15 show that DWT-BiGRU-attention generally produces more accurate predictions than other models for all hosts. Combining the DWT and BiGRU techniques enables the model to learn temporal dependency by moving forward and backwards in time from past host load data, extracting feature information about future host load changes, and achieving a more strong nonlinear generalizability than other methods.

Our method is more accurate than previous methods at predicting actual CPU usage. Moreover, it is essential to emphasize that our

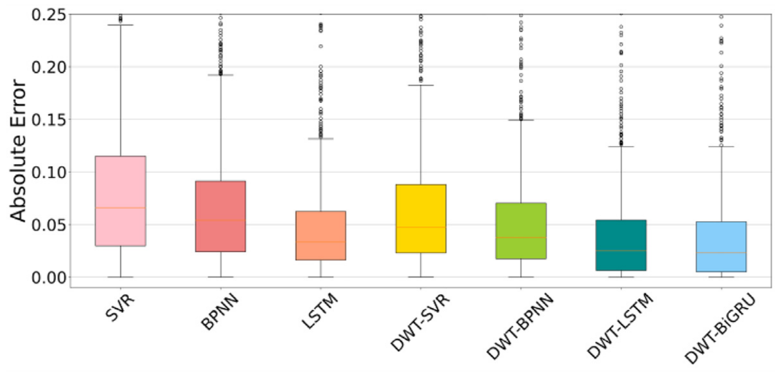

Fig. 13. The boxplot diagram for the absolute error of processor usage for each prediction algorithm using the Google cluster dataset.

Table 6
The results for Alibaba dataset host load.

	Machine A1			Machine A2			Machine A3			Machine A4		
	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE
SVR	16.6	7.41	5.51	7.19	0.049	0.041	24.18	2.5	2.0	24.7	9.0	7.20
BPNN	15.51	5.52	4.41	6.31	0.043	0.036	17.82	1.8	1.53	18.13	7.12	5.51
LSTM	12.0	4.81	3.76	4.43	0.033	0.025	14.95	1.69	1.38	14.6	5.56	4.41
BiGRU	11.72	4.70	3.52	4.30	0.030	0.25	14.73	1.62	1.37	14.4	5.41	4.19
DWT-SVR	15.45	5.86	4.68	6.72	0.045	0.038	19.74	2.1	1.6	23.7	8.30	6.93
DWT-BPNN	14.73	5.42	4.47	5.78	0.040	0.033	16.72	1.7	1.4	16.09	5.82	4.72
DWT-LSTM	11.70	4.62	3.59	4.25	0.029	0.024	14.53	1.6	1.35	14.21	5.47	4.25
DWT-BiGRU	10.66	4.21	3.32	4.03	0.028	0.024	14.20	1.5	1.23	13.45	5.23	4.14
DWT-LSTM-attention	11.23	4.35	3.42	4.12	0.029	0.024	14.42	1.5	1.24	13.92	5.31	4.24
DWT-BiGRU-attention	**10.4**	**3.9**	**3.15**	**3.96**	**0.027**	**0.022**	**14.12**	**1.48**	**1.22**	**13.2**	**5.21**	**4.11**

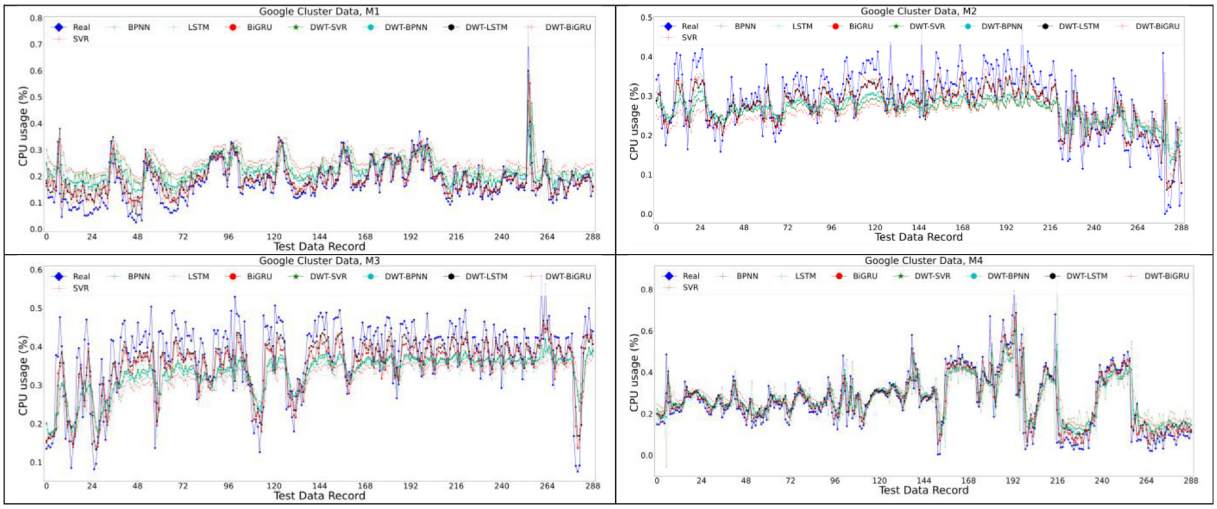

Fig. 14. Comparison of actual and predicted CPU usage for baseline and DWT based methods with Alibaba Cluster data for four selected machines.

prediction is more accurate when load fluctuations occur. For example, machine A3 demonstrates a high degree of fluctuation, with each record in the test data set is significantly different from the previous record. Our methodology employs DWT to distinguish between high and low frequencies, which improves the accuracy of predictions. Two hidden layers are connected in the opposite direction to the same output layer in BiGRU, enabling the output layer to collect data from both the past and the future, allowing accurate CPU load prediction.

Fig. 16 depicts the results of our comparison of the normalized MAPE, RMSE, and MAE of our method to those of other techniques. In terms of MAPE, RMSE, and MAE, the results indicate that DWT-BiGRU-attention predicts CPU usage in the Alibaba Cluster Dataset more accurately than conventional methods. Fig. 17 depicts a boxplot of the absolute CPU error for each prediction algorithm using the Alibaba cluster dataset. We found that, in terms of absolute error reduction, the DWT-BiGRU-attention strategy outperforms others.

The performance differences between the proposed model and various baseline models can be determined further by analyzing the results of the DM test. Table 7 demonstrates the results of the DM test of each baseline method using the Alibaba cluster dataset. As seen in Table 7, all but two of the *DM* test results for all machines exceed the upper limit of the 1% significance threshold. The prediction accuracy of our model varies significantly from other models. The DM test values of the DWT-BiGRU-attention model for predicting three steps exceed the upper limit of the 5% significance level. It indicates that the predicted performance of our model is significantly different from the DWT-BiGRU-attention model, with a probability of 95%.

Following is a comparison between the proposed method and a number of previous studies that utilized the Alibaba cluster dataset.

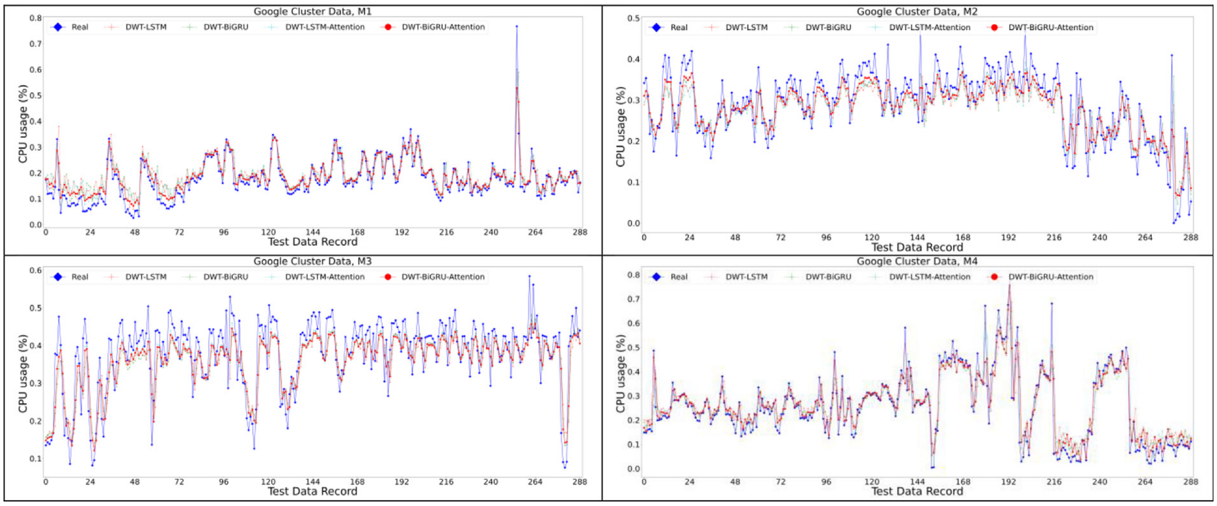

Fig. 15. Comparing actual and predicted CPU usage for baseline and DWT-based methods with Alibaba Cluster data for four selected machines.

Table 7
DM test results for Alibaba cluster machines.

Model	DM value (significance level)			
	M1	M2	M3	M4
SVR	8.367 (1%)	7.254 (1%)	6.665 (1%)	7.254 (1%)
BPNN	7.824 (1%)	5.520 (1%)	7.447 (1%)	5.520 (1%)
LSTM	6.881 (1%)	5.798 (1%)	5.559 (1%)	5.798 (1%)
BiGRU	7.796 (1%)	7.413 (1%)	4.212 (1%)	7.413 (1%)
DWT-SVR	4.058 (1%)	4.872 (1%)	3.259 (1%)	4.872 (1%)
DWT-BPNN	3.997 (1%)	3.075 (1%)	2.733 (1%)	3.075 (1%)
DWT-LSTM	3.808 (1%)	2.769 (1%)	2.381 (1%)	2.769 (1%)
DWT-BiGRU	4.058 (1%)	4.872 (1%)	3.259 (1%)	4.872 (1%)
DWT-LSTM-attention	3.997 (1%)	3.075 (1%)	2.733 (1%)	3.075 (1%)
DWT-BiGRU-attention	3.808 (1%)	2.769 (1%)	2.381 (1%)	2.769 (1%)

For these goals, the approaches described in [30,46], and [54] are employed. Table 8 demonstrates the error metric of different studies using the Alibaba cluster dataset.

According to Table 8, DWT-BiGRU-attention has a higher accuracy than the other studied. Comparing DWT-BiGRU-attention to other workload prediction studies reveals that the combination of DWT, BiGRU, and attention mechanism considerably reduces MAPE, RMSE, and MAE.

5.5. Discussion

Based on Tables 3–6 and Figs. 10–17, the complete comparison analysis results are summarized as follows:

– According to the experiment results, the SVR and BPNN approaches have the most significant prediction errors among the three criteria. These results demonstrated that SVR and BPNN cannot extract the linear and nonlinear features of host load traces. SVR employs vast quantities of data to map them into a high-dimensional space; therefore, the absence of temporal and periodic features leads in poor prediction accuracy. LSTM outperforms SVR and BPNN with regard to performance prediction. Comparing BiGRU to other baseline methods indicates that BiGRU significantly reduces MAPE, MAE, and RMSE prediction errors. BiGRU combines a forward and a backward unit and promises added benefits over unidirectional GRU and other baseline methods by allowing the extraction of dependencies between previous and future steps.

– A comparison of methods employing DWT compared to methods not using DWT demonstrates that employing DWT for prediction increases the prediction accuracy. For instance, according to the result for the M1 machine in Tables e and 5, the MAPE, RMSE, and MAE values for *BiGRU* are 22.5, 5.7, and 4.1, respectively, whereas the values for DWT-BiGRU are 21.6, 5.1, and 3.4, respectively, which is significantly less than the BiGRU error values. Similar comparisons indicate that the MAPE, MMRE, and MAE criteria of the DWT-SVR, DWT-BPNN, and DWT-LSTM methods have decreased compared to the SVR, BPNN, and LSTM. For all M1, M2, M3, and M4 machines, the employing DWT method reduced the MAPE, MMRE, and MAE values for all methods. Since DWT extracts temporal correlation between the records of the input host load data, employing DWT reduces significant fluctuations and makes more accurate predictions via learning short-term and long-term dependencies. As a result, it was concluded that learning nonstationary data and capturing the dynamic and nonlinear nature of workload time series data by DWT improved the accuracy of host resource usage prediction in cloud computing.

– Although both BiGRU and LSTM models may be used to learn long-term dependencies from historical host load traces, the MAPE, MAE, and RMSE criteria for the BiGRU model outperform those for LSTM. Compared to DWT-LSTM, the DWT-BiGRU significantly decreases prediction error rates for MAPE, MAE, and RMSE. It is because BiGRU predicts workload through bidirectional learning. The results indicate that BiGRU may give superior accuracy than LSTM in nonstationary host load data.

– Figs. 8 and 10 illustrate how the attention process contributes to reducing prediction error. Comparing the prediction performance of DWT-BiGRU-attention and DWT-LSTM-attention with DWT-BiGRU and DWT-LSTM, it is clear that the MAPE, MMRE, and MAE of attention-based prediction models are significantly lower than comparable approaches without an attention mechanism across all Machines. The attention mechanism provides different weights to resource usage at different times and considers the varied effect of historical data on future workload. The results of the comparative analysis indicate that the attention mechanism improves the host workload prediction performance.

– As shown in Fig. 14, the DWT-BiGRU-attention technique has a lower absolute error rate than the DWT-LSTM-attention method. It is because of the superior performance of our method with respect to long-term dependencies. The number of outlier points in the boxplot for DWT-LSTM-attention is more than that of DWT-BiGRU-attention.

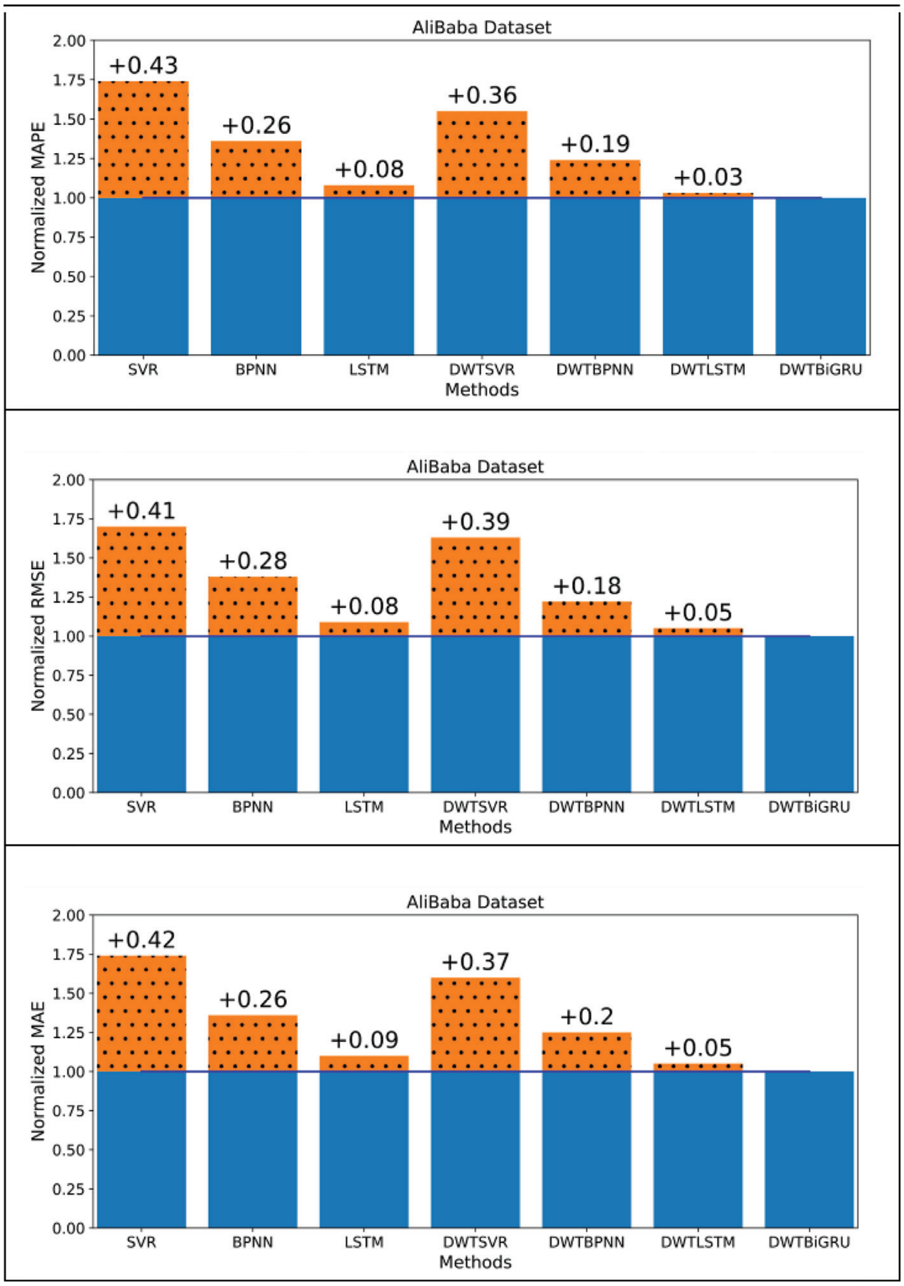

Fig. 16. Comparing normalized Metrics for different methods using Alibaba data set.

Table 8
The comparison of errors metrics in different studies using Alibab cluster host load.

	Machine G1			Machine G2			Machine G3			Machine G4		
	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE
Kernel based SVR [46]	14.76	5.81	4.63	6.30	0.042	0.035	18.75	2.07	1.71	22.79	8.33	6.42
BG-LSTM [30]	11.93	5.03	4.15	5.31	0.032	0.029	15.61	1.89	1.53	15.82	7.14	5.31
E2LG [54]	10.52	4.04	3.17	4.02	0.027	0.023	14.17	1.59	1.23	13.35	5.37	4.16
DWT-BiGRU-attention	**10.4**	**3.9**	**3.1**	**3.96**	**0.027**	**0.022**	**14.12**	**1.48**	**1.22**	**13.2**	**5.21**	**4.11**

In terms of RMSE, MAPE, and MAE, it causes the DWT-LSTM-attention approach is less accurate than the DWT-BiGRU-attention method.

– Even though some deep learning-based techniques, such as the LSTM network, are highly effective and accurate at predicting host load, they are often time-consuming and inappropriate for efficient resource allocation and scheduling based on prediction results. Due to the necessity of generating quick predictions on cloud computing data center load data, our technique is superior to LSTM-based methods in real-world applications.

– As demonstrated by Tables 5 and 8, the proposed method is more precise than other previous techniques. The method described in [46] is a workload prediction method that employs hybrid kernel-based SVR and then a voting mechanism to weigh the results. The reason why the proposed method is superior to [46] is that, unlike SVR-based methods, the DWT-BiGRU-attention effectively extracts hidden nonlinear patterns from workload data. It demonstrates that BiGRU combines forward and backward learning and reduces prediction errors significantly. The approach employed in [30] combines the BiLSTM

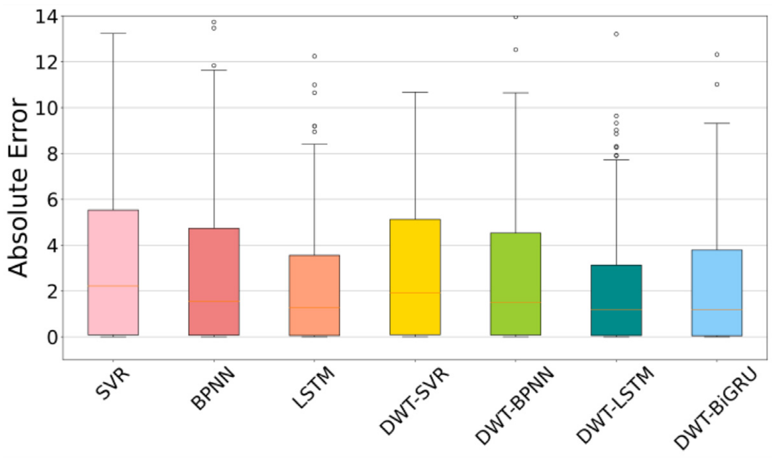

Fig. 17. The boxplot diagram for the absolute error of processor usage for each prediction algorithm using the Alibaba data set.

and GridLSTM models for accurate workload time series prediction. The approach employed in [54] decomposes the workload time series into its components in different frequency bands using EMD, and then a GAN/LSTM learning model predicts each sub-band workload time series. Comparing the results of the proposed method with [30,54] reveals that the accuracy of the proposed method has been enhanced by employing the technique of decomposing the input signal into many sub-bands and then employing the attention-based deep learning model.

6. Conclusion

This article proposes a host workload prediction method in cloud computing by combining the DWT, BiGRU model and attention mechanism. In addition to learning long-term dependencies in BiGRU, DWT can decompose nonlinear and nonstationary data into predictable sub-bands in order to predict future host workload in cloud computing. The proposed approach was evaluated using two real-time host load trace datasets, the Google Cluster Database and the Alibaba Cluster. According to the experimental results, basic techniques cannot learn nonlinear data, mainly when random fluctuations occur in the data. However, the model presented in the proposed method shows good compatibility and achieves better results than DWT-LSTM, DWT-BPNN, DWT-SVR, LSTM, BPNN, and SVR in both datasets. Since distributed computing is increasingly oriented toward lightweight virtualization technologies such as containers, we will predict container workloads in the Docker and Kubernetes environments in future research. We run various applications in these environments, and a series of simulations will run to generate tasks and measure the container's CPU load. Then, the workload will be predicted using machine learning models, and resource provisioning will be performed.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

CRediT authorship contribution statement

Javad Dogani: Methodology, Software, Simulation, Writing – original draft. **Farshad Khunjush:** Conceptualization, Validation, Methodology, Writing – review & editing. **Mehdi Seydali:** Conceptualization, Software, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] T. Khan, T., S. Tian, R. Ilager, W. Buyya, Workload forecasting and energy state estimation in cloud data centres: ML-centric approach, Future Gener. Comput. Syst. 128 (2022) 320–332, http://dx.doi.org/10.1016/j.future.2021.10.019.
[2] A.A. Khan, M. Zakarya, R. Khan, I.U. Rahman, M. Khan, A. Khan, An energy performance efficient resource consolidation scheme for heterogeneous cloud datacenters, J. Netw. Comput. Appl. 150 (2020) 102497, http://dx.doi.org/10.1016/j.jnca.2019.102497.
[3] A. Nelli, R. Jogdand, SLA-based workload scheduling technique in multi-cloud platform, J. Ambient Intell. Hum. Comput. (2022) http://dx.doi.org/10.1007/s12652-021-03666-z.
[4] T. Xie, C. Li, N. Hao, Y. Luo, Multi-objective optimization of data deployment and scheduling based on the minimum cost in geo-distributed cloud, Comput. Commun. 185 (2022) 142–158.
[5] K. Chakravarthi, L. Shyamala, TOPSIS inspired budget and deadline aware multi-workflow scheduling for cloud computing, J. Syst. Archit. 114 (2021) 101916, http://dx.doi.org/10.1016/j.sysarc.2020.101916.
[6] J. Zhu, Q. Li, S. Ying, SAAS parallel task scheduling based on cloud service flow load algorithm, Comput. Commun. 182 (2022) 170–183.
[7] R. Anantha Kumar, K. Kartheeban, Resource allocation using dynamic pricing auction mechanism for supporting emergency demands in cloud computing, J. Parallel Distrib. Comput. 158 (2021) 213–226, http://dx.doi.org/10.1016/j.jpdc.2021.07.016.
[8] J. Dogani, F. Khunjush, M.R. Mahmoudi, Seydali. M., Multivariate workload and resource prediction in cloud computing using CNN and GRU by attention mechanism, J. Supercomput. (2022) http://dx.doi.org/10.1007/s11227-022-04782-z.
[9] W. Shu, K. Cai, K, N. Xiong, Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing, Future Gener. Comput. Syst. 124 (2021) 12–20, http://dx.doi.org/10.1016/j.future.2021.05.012.
[10] M. Alotaibi, Hybrid metaheuristic technique for optimal container resource allocation in cloud, Comput. Commun. 191 (2022) 477–485.
[11] H. Kholidy, An intelligent swarm based prediction approach for predicting cloud computing user resource needs, Comput. Commun. 151 (2020) 133–144, http://dx.doi.org/10.1016/j.comcom.2019.12.028.
[12] R. Anantha Kumar, K. Kartheeban, Resource allocation using dynamic pricing auction mechanism for supporting emergency demands in cloud computing, J. Parallel Distrib. Comput. 158 (2021) 213–226, http://dx.doi.org/10.1016/j.jpdc.2021.07.016.
[13] H. Mezni, F. Hamoud, F. Charrada, Predictive service placement in cloud using deep learning and frequent subgraph mining, J. Ambient. Intell. Hum. Comput. (2022) http://dx.doi.org/10.1007/s12652-022-03720-4.
[14] S. Zaman, D. Grosu, Combinatorial auction-based dynamic VM provisioning and allocation in clouds, in: 2011 IEEE Third International Conference on Cloud Computing Technology and Science, 2011, http://dx.doi.org/10.1109/cloudcom.2011.24, [Preprint]. Available at.
[15] Rodero I. others, Towards energy-aware autonomic provisioning for virtualized environments, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10, 2010, http://dx.doi.org/10.1145/1851476.1851520, [Preprint]. Available at.

[16] A. Suresh, R. Varatharajan, Competent resource provisioning and distribution techniques for cloud computing environment, Cluster Comput. 22 (S5) (2017) 11039–11046, http://dx.doi.org/10.1007/s10586-017-1293-6, Available at.

[17] H. Shen, X. Hong, Host load prediction with Bi-directional long short-term memory in cloud computing, 2020, arXiv:2007.15582v1.

[18] G. Yang, et al., Prediction of the resource consumption of distributed deep learning systems, in: Abstract Proceedings of the 2022 ACM SIGMETRICS/IFIP PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, 2022, http://dx.doi.org/10.1145/3489048.3530962, [Preprint]. Available at.

[19] http://www https://predictkube.com/.

[20] O. Poppe, et al., Moneyball, Proc. VLDB Endow. 15 (6) (2022) 1279–1287, http://dx.doi.org/10.14778/3514061.3514073.

[21] Baldan F.J. others, A forecasting methodology for workload forecasting in cloud systems, IEEE Trans. Cloud Comput. 6 (4) (2018) 929–941, http://dx.doi.org/10.1109/tcc.2016.2586064, Available at.

[22] M.M. Al-Sayed, Workload time series cumulative prediction mechanism for cloud resources using neural machine translation technique, J. Grid Comput. 20 (2) (2022) http://dx.doi.org/10.1007/s10723-022-09607-0.

[23] M. Rhif, A. Ben Abbes, I. Farah, B. Martínez, Y. Sang, Wavelet transform application for/in non-stationary time-series analysis: A review, Appl. Sci. 9 (7) (2019) 1345, http://dx.doi.org/10.3390/app9071345.

[24] H. Toumi, Z. Brahmi, M.M. Gammoudi, RTSLPS: Real time server load prediction system for the ever-changing cloud computing environment, J. King Saud Univ. Comput. Inform. Sci. (2019) http://dx.doi.org/10.1016/j.jksuci.2019.12.004.

[25] X. Fu, W. Luo, C. Xu, X. Zhao, Short-term traffic speed prediction method for urban road sections based on wavelet transform and gated recurrent unit, Math. Probl. Eng. 2020 (2020) 1–13, http://dx.doi.org/10.1155/2020/3697625.

[26] J. Xiang, Z. Qiu, Q. Hao, H. Cao, Multi-time scale wind speed prediction based on WT-bi-LSTM, MATEC Web Conf. 309 (2020) 05011, http://dx.doi.org/10.1051/matecconf/202030905011.

[27] N. Djennane, et al., CPU-based prediction with self organizing map in Dynamic Cloud Data Centers, Int. J. Sensors Wirel. Commun. Control 11 (7) (2021) 733–747, http://dx.doi.org/10.2174/2210327910666201216123246, Available at.

[28] C. Fan, J. Wang, W. Gang, S. Li, Assessment of deep recurrent neural network-based strategies for short-term building energy predictions, Appl. Energy 236 (2019) 700–710, http://dx.doi.org/10.1016/j.apenergy.2018.12.004.

[29] E. Golshani, M. Ashtiani, Proactive auto-scaling for cloud environments using temporal convolutional neural networks, J. Parallel Distrib. Comput. 154 (2021) 119–141, http://dx.doi.org/10.1016/j.jpdc.2021.04.006.

[30] J. Bi, S. Li, H. Yuan, M.C. Zhou, Integrated deep learning method for workload and resource prediction in cloud systems, Neurocomputing 424 (2021) 35–48, http://dx.doi.org/10.1016/j.neucom.2020.11.011.

[31] S. Baig, W. Iqbal, J.L. Berral, A. Erradi, D. Carrera, Adaptive prediction models for data center resources utilization estimation, IEEE Trans. Netw. Serv. Manag. 16 (4) (2019) 1681–1693, http://dx.doi.org/10.1109/TNSM.2019.2932840.

[32] D. Lien Minh, A. Sadeghi-Niaraki, H.D. Huy, K. Min, H. Moon, Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network, IEEE Access 6 (2018) 55392–55404, http://dx.doi.org/10.1109/access.2018.2868970.

[33] A.S. Saud, S. Shakya, Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE, Procedia Comput. Sci. 167 (2020) 788–798, http://dx.doi.org/10.1016/j.procs.2020.03.419.

[34] G. Weiss, Y. Goldberg, E. Yahav, On the practical computational power of finite precision RNNs for language recognition, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2018, http://dx.doi.org/10.18653/v1/p18-2117.

[35] X. Cao, Y. Zhong, Y. Zhou, J. Wang, C. Zhu, W. Zhang, Interactive temporal recurrent convolution network for traffic prediction in data centers, IEEE Access 6 (2018) 5276–5289, http://dx.doi.org/10.1109/access.2017.2787696.

[36] H. Huang, N. Cressie, Spatio-temporal prediction of snow water equivalent using the Kalman filter, Comput. Statist. Data Anal. 22 (2) (2016) 159–175, http://dx.doi.org/10.1016/0167-9473(95)00047-X.

[37] S.L. Ho, M. Xie, T.N. Goh, A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction, Comput. Ind. Eng. 42 (2–4) (2002) 371–375.

[38] R.N. Calheiros, E. Masoumi, R. Ranjan, R. Buyya, Workload prediction using ARIMA model and its impact on cloud applications' QoS, IEEE Trans. Cloud Comput. 3 (4) (2015) 449–458, http://dx.doi.org/10.1109/TCC.2014.2350475.

[39] J. Chen, Y. Wang, A hybrid method for short-term host utilization prediction in cloud computing, J. Electr. Comput. Eng. (2019) 1–14, http://dx.doi.org/10.1155/2019/2782349.

[40] K.W. Lau, Q.H. Wu, Local prediction of nonlinear time series using support vector regression, Pattern Recognit. 41 (5) (2008) 1539–1547, http://dx.doi.org/10.1016/j.patcog.2007.08.013.

[41] M.S. Raimundo, J. Okamoto, SVR-wavelet adaptive model for forecasting financial time series, in: 2018 International Conference on Information and Computer Technologies, ICICT, 2018, pp. 111–114, http://dx.doi.org/10.1109/INFOCT.2018.8356851.

[42] R. Hu, J. Jiang, G. Liu, L. Wang, CPU load prediction using support vector regression and Kalman smoother for cloud, in: 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, 2013, http://dx.doi.org/10.1109/icdcsw.2013.60.

[43] R. Hu, J. Jiang, G. Liu, L. Wang, Kswsvr: A new load forecasting method for efficient resources provisioning in cloud, in: 2013 IEEE International Conference on Services Computing, 2013, http://dx.doi.org/10.1109/scc.2013.67.

[44] S. Sharifian, M. Barati, An ensemble multi-scale wavelet-GARCH hybrid SVR algorithm for mobile cloud computing workload prediction, Int. J. Mach. Learn. Cybern. 10 (11) (2019) 3285–3300.

[45] W. Zhong, Y. Zhuang, J. J. Sun, J. Gu, A load prediction model for cloud computing using PSO-based weighted wavelet support vector machine, Appl. Intell. 48 (11) (2018) 4072–4083.

[46] P. Nehra, A. Nagaraju, Host utilization prediction using hybrid kernel based support vector regression in cloud data centers, J. King Saud Univ. Comput. Inform. Sci. (2021) http://dx.doi.org/10.1016/j.jksuci.2021.04.011.

[47] W. Jiang, Z. Song, J. Zhan, Z. He, X. X. Wen, K. K. Jiang, Optimized co-scheduling of mixed-precision neural network accelerator for real-time multitasking applications, J. Syst. Archit. 110 (2020) 101775, http://dx.doi.org/10.1016/j.sysarc.2020.101775.

[48] Y. Lu, J. Panneerselvam, L. Liu, Y. Wu, RVLBPNN: A workload forecasting model for smart cloud computing, Sci. Program. 2016 (2016) 1–9, http://dx.doi.org/10.1155/2016/5635673.

[49] S. Jeddi, S. Sharifian, A water cycle optimized wavelet neural network algorithm for demand prediction in cloud computing, Cluster Comput. 22 (4) (2019) 1397–1412.

[50] Z. Zhang, X. Tang, J. Han, P. Wang, Sibyl: Host load prediction with an efficient deep learning model in cloud computing, Algor. Archit. Parallel Process. (2018) 226–237, http://dx.doi.org/10.1007/978-3-030-05054-2_17.

[51] H.M. Nguyen, G. Kalra, D. Kim, Host load prediction in cloud computing using long short-term memory encoder–decoder, J. Supercomput. 75 (11) (2019) 7592–7605.

[52] H. Shuvo, M.N. Hasan Shuvo, M.N. Shahriar Maswood, M.M. Shahriar Maswood, M.M. Alharbi, A.G. Alharbi, LSRU: A novel deep learning based hybrid method to predict the workload of virtual machines in cloud data center, in: 2020 IEEE Region 10 Symposium, TENSYMP, 2020, http://dx.doi.org/10.1109/tensymp50017.2020.9230799.

[53] B. Song, Y. Yu, Y. Zhou, Z. Wang, S. Du, Host load prediction with long short-term memory in cloud computing, J. Supercomput. 74 (12) (2017) 6554–6568.

[54] P. Yazdanian, S. Sharifian, E2LG: a multi-scale ensemble of LSTM/GAN deep learning architecture for multistep-ahead cloud workload prediction, J. Supercomput. (2021) http://dx.doi.org/10.1007/s11227-021-03723-6.

[55] Q. Yang, Y. Zhou, Y. Yu, J. Yuan, X. Xing, S. Du, Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing, J. Supercomput. 71 (8) (2015) 3037–3053.

[56] C. Peng, Y. Li, Y. Yu, Y. Zhou, S. Du, Multi-step-ahead host load prediction with GRU based encoder-decoder in cloud computing, in: 2018 10th International Conference on Knowledge and Smart Technology, KST, 2018, http://dx.doi.org/10.1109/kst.2018.8426104.

[57] Y. Wen, Y. Wang, J. Liu, B. Cao, Q. Fu, CPU usage prediction for cloud resource provisioning based on deep belief network and particle swarm optimization, Concurr. Comput.: Pract. Exper. 32 (14) (2020) http://dx.doi.org/10.1002/cpe.5730.

[58] S. Tofighy, A.A. Rahmanian, M. Ghobaei-Arani, An ensemble CPU load prediction algorithm using a Bayesian information criterion and smooth filters in a cloud computing environment, Softw. - Pract. Exp. 48 (12) (2018) 2257–2277, http://dx.doi.org/10.1002/spe.2641.

[59] J. Kumar, A. Kumar Singh, R. Buyya, Ensemble learning based predictive framework for virtual machine resource request prediction, Neurocomputing 397 (2020) 20–30, http://dx.doi.org/10.1016/j.neucom.2020.02.014.

[60] A. Feltane, Time-frequency based methods for non-stationary signal analysis with application to EEG signals, 2016, Open Access Dissertations Paper 445 https://digitalcommons.uri.edu/oa_diss/445.

[61] S. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. Pattern Anal. Mach. Intell. 11 (7) (1989) 674–693, http://dx.doi.org/10.1109/34.192463.

[62] L. Song, J. Xue, X. Wang, J. Zhang, L. Wang, Z. Jiang, J. Cheng, Time-series well performance prediction based on long short-term memory (LSTM) neural network model, J. Pet. Sci. Eng. 186 (2020) http://dx.doi.org/10.1016/j.petrol.2019.106682.

[63] J. Zhao, D. Zeng, S. Liang, H. Kang, Q. Liu, Prediction model for stock price trend based on recurrent neural network, J. Ambient. Intell. Hum. Comput. 12 (2021) 745–753, http://dx.doi.org/10.1007/s12652-020-02057-0.

[64] D. She, M. Jia, A BiGRU method for remaining useful life prediction of machinery, Measurement 167 (2021) 108277, http://dx.doi.org/10.1016/j.measurement.2020.108277.

[65] H. Zou, H. Liu, T. Zhou, L. Jiashun, Y. Zhan, Short-term traffic flow prediction using DTW-bigru model, in: 2020 35th Youth Academic Annual Conference of Chinese Association of Automation, YAC, 2020, pp. 557–562, http://dx.doi.org/10.1109/YAC51587.2020.9337579.

[66] Z. Niu, G. Zhong, H. Yu, A review on the attention mechanism of deep learning, Neurocomputing 452 (2021) 48–62, http://dx.doi.org/10.1016/j.neucom.2021.03.091, Available at.
[67] Fei N. others, Z-score normalization, hubness, and few-shot learning, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). Available At, 2021, http://dx.doi.org/10.1109/iccv48922.2021.00021.
[68] [dataset] https://github.com/Google/cluster-data.
[69] [dataset] https://github.com/alibaba/clusterdata.
[70] H. Salem, A.E. Kabeel, E.M.S. El-Said, O.M. Elzeki, Predictive modelling for solar power-driven hybrid desalination system using artificial neural network regression with adam optimization, Desalination 522 (2022) 115411, http://dx.doi.org/10.1016/j.desal.2021.115411.
[71] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10) Israel, Haifa, 2010, pp. 807–814.

Javad Dogani received his B.Sc. degree in software engineering from technical and Vocational University, Shiraz, Iran, in 2009 and the M.Sc. degree in software engineering from, School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran in 2012. His Ph.D. is in the field of Cloud Computing from the School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran. His main research areas are Distributed systems, Cloud Computing, Machine Learning, Parallel Computing, and Big-Data Processing.

Farshad Khunjush received the B.Sc. and M.Sc. degrees in Computer Engineering from Shiraz University and a Ph.D. degree from the University of Victoria, Canada, 2008. He was a Post-Doctoral fellow at the Laboratory for Parallel and Intelligent Systems (LAPIS) (2008 to 2009). He joined the Department of Computer Science at Shiraz University in 2009. He served as the head of System/Software Engineering (2010–2014) and acted as the director of ICT Center (2014–2020). He was a visiting professor at EPFL in 2019 and the University of Toronto (2020–2021). His research interests include Multi-Core & Parallel Computer Architectures and Cloud Computing.

Mehdi Seydali received his B.Sc. degree in software engineering from Shahid Bahonar University, Kerman, Iran, in 2008, Iran and M.Sc. degree from Tarbiat Modares University, Tehran, Iran, in 2013. His Ph.D. is in the field of Cloud Computing from the School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran. His research interest is mainly in Big data, Cloud computing, parallel processing and distributed deep learning.