Accurate Prediction of Workloads and Resources With Multi-Head Attention and Hybrid LSTM for Cloud Data Centers

Jing Bi, *Senior Member, IEEE*, Haisen Ma, *Student Member, IEEE*, Haitao Yuan, *Senior Member, IEEE*, and Jia Zhang, *Senior Member, IEEE*

***Abstract*—Currently, cloud computing service providers face big challenges in predicting large-scale workload and resource usage time series. Due to the difficulty in capturing nonlinear features, traditional forecasting methods usually fail to achieve high prediction performance for resource usage and workload sequences. Besides, there is much noise in original time series of resources and workloads. If these time series are not de-noised by smoothing algorithms, the prediction results can fail to meet the providers' requirements. To do so, this work proposes a hybrid prediction model named VAMBiG that integrates Variational mode decomposition, an Adaptive Savitzky-Golay (SG) filter, a Multi-head attention mechanism, Bidirectional and Grid versions of Long and Short Term Memory (LSTM) networks. VAMBiG adopts a signal decomposition method named variational mode decomposition to decompose complex and non-linear original time series into low-frequency intrinsic mode functions. Then, it adopts an adaptive SG filter as a data pre-processing tool to eliminate noise and extreme points in such functions. Afterwards, it adopts bidirectional and grid LSTM networks to capture bidirectional features and dimension ones, respectively. Finally, it adopts a multi-head attention mechanism to explore importance of different data dimensions. VAMBiG aims to predict resource usage and workloads in highly variable traces in clouds. Extensive experimental results demonstrate that it achieves higher-accuracy prediction than several advanced prediction approaches with datasets from Google and Alibaba cluster traces.**

***Index Terms*—Cloud data centers, deep learning, LSTM, adaptive Savitzky-Golay filter, attention mechanisms.**

Manuscript received 14 August 2022; revised 6 December 2022; accepted 2 March 2023. Date of publication 20 March 2023; date of current version 8 September 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62173013 and 62073005, in part by the Beijing Natural Science Foundation under Grant 4232049, and in part by the Fundamental Research Funds for the Central Universities under Grant YWF-22-L-1203. This paper was presented at the IEEE SMC 2022, Prague, Czech Republic [DOI: 10.1109/SMC53654.2022.9945419]. Recommended for acceptance by S. Song. *(Corresponding author: Haitao Yuan.)*

Jing Bi and Haisen Ma are with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn; mahaisen@emails.bjut.edu.cn).

Haitao Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: haitao.yuan@njit.edu).

Jia Zhang is with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX 75205 USA (e-mail: jiazhang@smu.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSUSC.2023.3259522, provided by the authors.

Digital Object Identifier 10.1109/TSUSC.2023.3259522

I. Introduction

OVER the past few decades, cloud computing is gradually being widely adopted by many large-scale companies and organizations. The world's leading Internet companies such as Google, Amazon, and Facebook have built their own cloud data centers (CDCs) [1], [2]. These CDCs provide users with computing, storage and network resources on demand through a shared resource pool. Users of CDCs may be individual users or companies. Therefore, resource requests range from short-term resource-intensive tasks to long-running user-oriented services. Dramatic changes in workloads cause over or under-provisioning of resources. Cloud service providers (CSPs) need to quickly determine resource allocation strategies to avoid violation of service level agreements [3]. Therefore, adaptive and accurate prediction of workloads and resource usage is highly pressing and essential for CSPs. However, workloads in CDCs are highly variable and fluctuate frequently. Resource usage also changes continuously during the execution of tasks. According to [4], the average CPU usage of entire clusters in Alibaba CDCs ranges from 10% to 80% with high fluctuations. When users' requests arrive at the same time, workloads increase dramatically, thereby leading to a shortage of available resources. On the contrary, if the workloads stay at a low level, idle clusters result in huge waste of resources.

The problem of workload prediction has attracted considerable studies in cloud computing. Several forecasting methods have been used for resource usage or workloads time series. Traditional prediction methods include linear regression [5], neural networks [6], Autoregressive Integrated Moving Average (ARIMA) [7] and Artificial Neural Network [8]. However, most of them require workloads with obvious regularity. Besides, these methods cannot efficiently capture non-linear characteristics of time series. Due to excellent sequence processing capability, recurrent neural network (RNN) models and its variants [9], [10], [11] have been widely used for highly variable workload prediction in recent years. However, they usually neglect the denoising process of the original time series. In addition, they only adopt a single deep learning network model, and cannot efficiently extract long-term dependencies or different dimension information in workload or resource usage data.

To address it, this work proposes a hybrid prediction model named VAMBiG that integrates the Variational mode decomposition (VMD) [12], the Adaptive Savitzky-Golay (SG) filter, the Multi-head attention mechanism, BiLSTM [13] and GridLSTM [14]. Main contributions are three-fold:

1) VAMBiG innovatively adopts two signal processing methods in the data prepricessing stage. It applies VMD to extract low-frequency and high-frequency features from the original sequences. It also applies an adaptive noise reduction method by dynamically changing critical parameters of an SG filter.
2) After data preprocessing, a hybrid LSTM-based deep learning model combining bidirectional LSTM (BiLSTM), grid LSTM (GridLSTM) and attention mechanisms is proposed to achieve highly accurate prediction of workloads and resource usage. In this way, long-term dependencies and intricate characteristics in time series are learnt.
3) Extensive experiments are conducted with real-world workloads and resource usage to verify VAMBiG. The results demonstrate that VAMBiG outperforms typical benchmark methods and several RNN-based models in terms of prediction accuracy.

For clarity, we note major differences between the current one and our prior work [15] as follows.

1) Different from [15], this work further extracts non-linear features of time series data, and adopts VMD to decompose complex and non-linear original workload and resource usage time series into low-frequency and intrinsic mode functions with different characteristics.
2) Different from [15], this work investigates the importance of different output nodes of hidden layers by integrating the attention mechanism after the layers of GridLSTM and BiLSTM.
3) The work in [15] only adopts a single type of a dataset from the Google cluster. Different from it, this work adopts two types of real-world datasets from Google and Alibaba clusters to demonstrate both robustness and prediction performance of VAMBiG. For each dataset, we consider three different types of the time series data including workload, CPU usage, and RAM usage.

The rest of this work is organized as follows. Section II introduces the related work on time series prediction. Section III describes the model framework of VAMBiG. Section IV evaluates VAMBiG via experiments with real-world workload traces data. Finally, Section V concludes this work.

II. Related Work

This section mainly introduces prediction methods for time series, which are mainly classified into traditional prediction methods and deep learning-based prediction ones.

A. Traditional Time Series Prediction Methods

Several classical forecasting methods are widely used in time series forecasting and workload prediction in cloud computing. In previous studies, an autoregressive integrated moving average (ARIMA) model is used to predict future workload according to historical real traces of requests to web servers in [16]. However, ARIMA has poor performance in capturing nonlinear characteristics of time series data in a complex CDC environment. Besides, they only adopt a single data type of web servers. Yunus et al. [17] propose a modified ARIMA model to simulate wind speed time series data. It adopts a decomposition approach to split data into high-frequency and low-frequency parts and models these two parts, respectively. Yet, it assumes that the transformed wind-speed data is stationary, and it has less data fluctuation than the workload data in our work. A support vector regression (SVR)-based method is presented in [18] to implement the long-term prediction of energy consumption. However, it partially ignores potential features derived from the data, especially when considering highly non-stationary and non-linear sequences. To improve the performance of SVR, an empirical mode decomposition (EMD) method is exploited to extract local trends characterizing the data in the preprocessing. Lu et al. [6] combine a K-means clustering algorithm and a backpropagation neural network (BPNN) to predict the future workload trend of CDCs. However, the initial value of K is difficult to determine and BPNN suffers from a vanishing gradient problem.

In summary, most of traditional time series prediction methods are based on heuristic algorithms, traditional neural networks or regression methods. Yet, they require regularity or obvious trends in the original sequence data. Different from these methods, we concatenate two deep learning prediction models (BiLSTM and GridLSTM) based on recurrent neural networks to predict highly variable and non-linear workload data in CDCs. Besides, this work considers three different types of time series data including CPU, RAM usage and the number of arriving tasks.

B. Deep Learning-Based Prediction Methods

Due to limitations caused by traditional forecasting methods on data characteristics, many studies have adopted deep learning methods to realize time series forecasting. Among them, RNNs-based approaches have shown great success in the time series prediction. Over the past few years, LSTMs have been adopted to solve challenges of workload forecasting in CDCs. Huang et al. [19] propose an RNN-LSTM model to forecast the workload and performance of web servers. However, combining LSTMs into RNN structures does not guarantee improved performance in the long-term prediction. A hybrid model integrating convolutional neural network (CNN) and LSTM is adopted in [20] for the multivariate workload data prediction including the CPU, memory, and network usage. The CNN layer extracts spatial features of data and the LSTM one is suitable for modeling temporal information. However, it does not adopt filtering algorithms in the preprocessing stage to reduce the noise in the raw data, thereby reducing the prediction accuracy. Chen et al. [21] propose a gate recurrent unit (GRU)-based method to achieve the workload prediction. Due to its different gate structures, GRU converges more easily with fewer parameters. However, regardless of LSTM or GRU, the prediction model

TABLE I
DIFFERENCE BETWEEN THIS WORK AND RELATED WORKS

Predication technique	Non-linearity	Large dataset	Decomposition method	Data filtering	Attention mechanism	References
RNN-LSTM	×	×	×	×	×	[19]
CNN-LSTM	×	×	×	×	×	[20]
ARIMA	✓	×	×	×	×	[16]
SVR	×	×	✓	×	×	[18]
K-RVLBPNN	✓	✓	×	×	×	[6]
L-PAW	✓	✓	×	×	×	[21]
SABG	✓	✓	×	✓	×	[15]
Our work	✓	✓	✓	✓	✓	

only extracts the correlation information in the forward direction between the time series data. To solve this issue, bidirectional RNNs are used to train their parameters in both forward and backward directions to understand the sequence context. Ko et al. [22] adopt BiLSTM for the wind power forecasting. However, the wind power data adopted in their work tends to be less volatile than the workload data from CDCs. Li et al. [23] propose a framework based on a sparrow search algorithm (SSA) and bidirectional GRU (BiGRU) for the oil rate forecasting. The SSA is chosen to find appropriate hyperparameters and the BiGRU is adopted to extract the bidirectional sequence information from the oil rate time series.

Overall, most of deep learning-based prediction methods adopt only one variant of RNN or LSTM. In addition to this, these methods fail to adopt a suitable algorithm for the noise reduction of the original data during the data preprocessing stage. Different from above-mentioned studies, our proposed VAMBiG is a novel model that integrates BiLSTM, GridLSTM and the attention mechanism to perform more accurate prediction for large-scale and non-linear resource usage and workload data in CDCs. BiLSTM and GridLSTM capture bidirectional features and different dimension information, respectively. Besides, in the data preprocessing stage, our model adopts the VMD to extract non-linear characteristics and the adaptive SG filter to achieve better denoising performance under low-distortion conditions. The experimental results demonstrate that our proposed VAMBiG outperforms several typical deep learning prediction models, e.g., LSTM, BiLSTM, and GridLSTM. Table I presents the difference between our proposed work and related works in a holistic set of metrics: non-linearity, handling large data set, decomposition method, data filtering, and attention mechanism.

III. MODEL FRAMEWORK

The details of VAMBiG are described, and they inlcude data preprocessing and data prediction.

A. Problem Definition

This work addresses a problem of predicting the amount of resource usage and the number of arriving tasks in a CDC. The previous m time slots are the input sequence I ($I = I_1, I_2, \ldots, I_{m-1}, I_m$). I is used to forecast the number of arriving tasks or the amount of resource usage $\hat{y}_{m+1}$ at time slot $m+1$. The relation between I and $\hat{y}_{m+1}$ is given as:

$$\hat{y}_{m+1} = \mathrm{f}\left(I_1, I_2, \ldots, I_{m-1}, I_m\right). \tag{1}$$

The training objective is to minimize the error between the predicted value $\hat{y}_{m+1}$ and the ground-truth one y_{m+1}.

B. Long Short Term Memory (LSTM)

Compared with traditional RNNs, LSTM [24] improves the memory capacity of network models by designing memory cells. The memory cell is formulated as:

$$\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}{+}x_t] + b_f\right)\\
i_t &= \sigma\left(W_i \cdot [h_{t-1}{+}x_t] + b_i\right)\\
\widetilde{c}_t &= \tanh\left(W_{\widetilde{c}} \cdot [h_{t-1} + x_t] + b_{\widetilde{c}}\right)\\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t\\
o_t &= \sigma\left(W_o \cdot [h_{t-1} + x_t] + b_o\right)\\
h_t &= o_t \cdot \tanh\left(c_t\right).
\end{aligned} \tag{2}$$

where i, f and o denote the input, the forget, and the output gate units. c denotes a state of the LSTM cell. x denotes the input of the LSTM cell. h_{t-1} denotes the output result of the LSTM memory cell at time $t{-}1$. W denotes the weight matrix and b denotes the bias. $\sigma(\cdot)$ and $\tanh(\cdot)$ denote the sigmoid and hyperbolic tangent functions, respectively.

C. Signal Decomposition

Ordinary forecasting methods may lead to the loss of local transient information of the series due to non-linear and non-stationary characteristics of workload time series in CDCs. To analyze local characteristics of non-linear sequences, the time series can be regarded as signals, which can be decomposed at multiple scales with signal decomposition methods to obtain a simpler set of sub-signals. The features can be extracted more easily with these sub-signals are more easily for improving final prediction results.

VMD is an adaptive and non-recursive signal decomposition algorithm [12]. It can be used to decompose the input time series data into R discrete mode functions [25]. For each mode, central

pulsation and bandwidth are determined in the VMD decomposition process. To obtain the bandwidth, there are three steps to be completed. In step one, a unilateral frequency spectrum is obtained by the Hilbert transform for each mode. In step two, the central frequency of each mode is evaluated by exponential tuned to transform it into the baseband. In step three, the bandwidth of each mode is estimated by adopting the Gaussian smoothing. The variational problem with constraints is generalized as:

$$\min_{u_r,w_r}\left\{\sum_{r=1}^{R}\left\|\partial_t\left[\left(\delta(t)+\frac{j}{\pi t}\right)*u_r(t)\right]e^{-jw_rt}\right\|_2^2\right\}$$
$$\text{s.t.}\sum_{r=1}^{R}u_r(t)=s(t). \tag{3}$$

where $s(t)$ is the original input, u_r denotes the r-th mode, w_r denotes the r-th mode center frequency. R denotes the number of modes. $\|\|_2^2$ denotes the squared L^2-norm. t denotes the time step, and $\delta(t)$ denotes the Dirac distribution.

To make the problem unconstrained, the augmented Lagrangian item combined with quadratic penalty and Lagrangian multiplier λ is given as:

$$L(u_r,w_r,\lambda)=\alpha\sum_{r=1}^{R}\left\|\partial_t\left[\left(\delta(t)+\frac{j}{\pi t}\right)*u_r(t)\right]e^{-jw_rt}\right\|_2^2$$
$$+\left\|s(t)-\sum_{r=1}^{R}u_r(t)\right\|_2^2+\left\langle\lambda(t),s(t)-\sum_{r=1}^{R}u_r(t)\right\rangle. \tag{4}$$

The original minimization problem is solved by an alternate direction method of multipliers (ADMM). Then, the modes $u_r(t)$ and w_r are updated by ADMM, which is given as:

$$u_r^{n+1}(w)=\frac{s(w)-\sum_{i\neq R}u_i(w)+\frac{\lambda(w)}{2}}{1+2\alpha\left(w-w_r\right)^2}$$
$$w_r^{n+1}=\frac{\int_0^\infty w\left|u_r(w)\right|^2dw}{\int_0^\infty\left|u_r(w)\right|^2dw}. \tag{5}$$

where $s(w)$, $u_(w)$, and $\lambda(w)$ are the Fourier transform of $s(t)$, $u_(t)$ and $\lambda(t)$, respectively.

The modes divided from the original data by utilizing VMD are also called intrinsic mode functions (IMFs). Compared with the original input data, the non-stationarity of IMFs is reduced. Each IMF contains the information of a separate part of the original time series.

D. Adaptive SG Filter

The SG filter is a widely used data smoothing and denoising algorithm [26]. It estimates the sequence data locally within a moving window by polynomial fitting of fixed orders. By choosing a symmetric window of $2m$+1 (m denotes the half length of window) samples centered at $n=0$ in a time series $X[n]$, the fitting polynomial is denoted by $p(n)$, which is obtained as:

$$p(n)=\sum_{k=0}^{N}a_qn^q\quad n\in[-m,m]. \tag{6}$$

where N denotes a given polynomial order, and a_q denotes the qth coefficient of the polynomial.

Mean square error (MSE), ϵ, is minimized to obtain the optimal coefficients (a_k) by the least square method, i.e.,

$$\epsilon=\sum_{n=-m}^{m}(p(n)-X[n])^2=\sum_{n=-m}^{m}\left(\sum_{k=0}^{N}a_qn^q-X[n]\right)^2. \tag{7}$$

The smoothed data is obtained by $p(n)$ at the central time step $n=0$ as $p(0)=a_0$. The above procedure is repeated for each data point. However, the fixed coefficients of the typical SG filter lack adaptability when time series data changes rapidly. To improve denoise performance under low distortion conditions, the adaptive SG filter [27] is used to dynamically change the order of fitting polynomial according to time series. For one-dimensional time series data, discrete curvature is introduced to represent the fluctuation of the number of arriving tasks or resource usage. The time series data can be considered as planar curves. The polynomial order is adaptively selected according to the estimation of the discrete curvature of each data point. An estimation method of one dimensional discrete curvature based on standard estimation methods [28] is used for the time series data.

First, we estimate discrete curvature by the variation of tangent, which is approximated by forward and backward longest digital straight segment (DSS) of each data point. For both sides of each data point, two DSSs and their tangents are estimated. We compute curvature by variation between two tangents. Considering each point $p_i=(t_i,x_i)$ of the time series, we compute the centered slope angle variation of p_i as:

$$\theta_{i,k}=\tan^{-1}\left(\left|\frac{x_i-x_{i-k}}{t_i-t_{i-k}}\right|\right)$$
$$\delta_{i,k}=\theta_{i+1,k}-\theta_{i-1,k}. \tag{8}$$

where i denotes the ith time step, k denotes the searching length of DSS, and $\theta_{i,k}$ is slope of the tangent line at p_i.

Besides, the forward ($f_{i,k}$) and backward ($b_{i,k}$) searching vectors are defined as:

$$f_{i,k}=p_i-p_{i+k}$$
$$b_{i,k}=p_i-p_{i-k}. \tag{9}$$

To obtain the maximum length of these two searching vectors for a DSS, we initialize a small critical angle Δ and a maximum length of searching vector k_m as the range indicator. Δ represents the reasonable deviation of time series data points due to noise. When one of central Angle variations of each data point in the sequence is equal or less than Δ, the sequence is regarded as a DSS. The maximum length of searching vector is defined as:

$$k_f=\max\{k:\forall s(-\Delta\leq\delta_{i+s,2}\leq\Delta\leftarrow1\leq s\leq k)\}$$
$$k_b=\max\{k:\forall s(-\Delta\leq\delta_{i-s,2}\leq\Delta\leftarrow1\leq s\leq k)\} \tag{10}$$

where k_f and k_b denote the maximum length of forward and backward searching vectors, and $\delta_{i+s,2}$ denotes the slope angle of the tangent line of each time series data point is calculated by the second neighbor.

Then, L_f (L_b) denotes the length of the forward (backward) DSS, which are given as:

$$L_f = \sqrt{(x_i - x_{i+k_f})^2 + (t_i - t_{i+k_f})^2}$$
$$L_b = \sqrt{(x_i - x_{i-k_b})^2 + (t_i - t_{i-k_b})^2}$$
$$\theta_f = \tan^{-1}\left(\left|\frac{x_i - x_{i+k_f}}{t_i - t_{i+k_f}}\right|\right)$$
$$\theta_b = \tan^{-1}\left(\left|\frac{x_i - x_{i-k_b}}{t_i - t_{i-k_b}}\right|\right). \quad (11)$$

where θ_f (θ_b) denotes angle of forward (backward) DSS.

Finally, the discrete curvature at time step i, denoted by C_i, is calculated as:

$$C_i = \frac{(L_b + L_f)(\theta_b + \theta_f)}{4 L_b L_f} \quad (12)$$

The adaptive SG filter is characterized by the adaptive selection of the fitting polynomial order according to discrete curvature values of time series data. The discrete curvature of each data point is mapped to a sequence of orders, denoted by Order(i). Order(i) is given as:

$$\text{Order(i)} = \max\left(J, \text{floor}\left(\frac{NC_i}{C_{\max} - C_{\min}} + \frac{1}{2}\right)\right) \quad (13)$$

where floor(x) generates the largest integer less than or equal to x, and $C_{\min}$ and $C_{\max}$ denote the maximum and minimum values of discrete curvature, respectively.

Consequently, we obtain a dynamically changing sequence of orders of each data point. For each data point, the data is smoothed by fitting polynomials of different orders.

E. Multi-Head Attention Mechanism

Instead of applying a single and common attention function [29], this work adopts the multi-head attention [30] to explore the importance of different output dimensions of the GridLSTM layer in VAMBiG. The Multi-head attention follows the ideas of CNN models with multiple convolutional kernels in each convolutional layer. Each head performs computation of attentions simultaneously with no sharing of parameters among each other. Eventually, computation results of each head are integrated together. It allows models to learn relevant information about the time series in different subspaces and can extract richer feature information. The input of the multi-head attention layer is the output of the GridLSTM layer. The attention module has three matrices named query Q, key K and value V. The output of the GridLSTM layer is denoted as a global feature matrix M. M becomes the initial value of the query matrix Q, the key matrix K, and the value matrix V. It is beneficial to linearly project these three matrices to Q', K', and V'. Then, multiple heads that can extract related information from different dimension features are divided by Q', K', and V'. The computation of the attention function is the dot-product attention, which is given as:

$$\text{Attention}\left(Q', K', V'\right) = \text{softmax}\left(\frac{Q' K'^T}{\sqrt{d_k}}\right) V'. \quad (14)$$

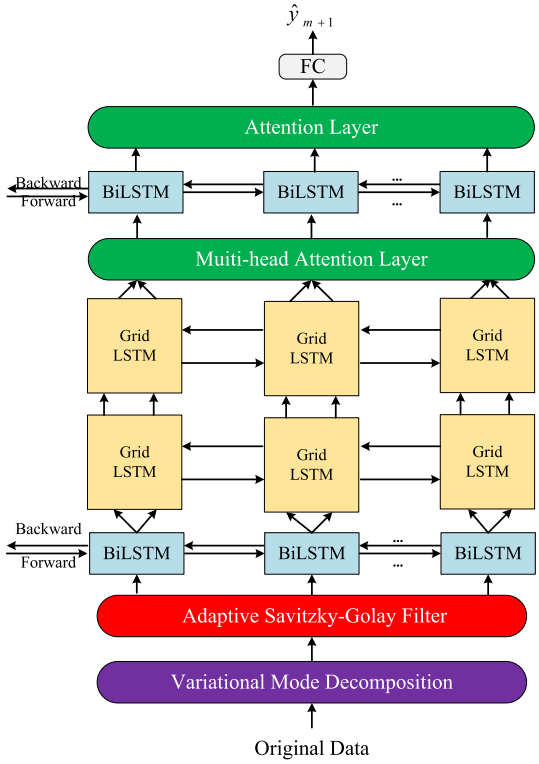

Fig. 1. Structure of VAMBiG.

where Q' is compared to K' by computing their similarity. The $softmax(\cdot)$ denotes the softmax function.

The multi-head attention adopts the scaled dot product attention in parallel for h times and h denotes the number of heads. Furthermore, we concatenate these head vectors. Finally, the concatenated attention enters a linear layer to generate new representations.

$$\text{MultiHead}\left(Q', K', V'\right) = \text{Concat}\left(head_1, \ldots, head_h\right) W.$$
$$head_i = \text{Attention}\left(Q', K', V'\right) \quad (15)$$

F. VAMBiG

As shown in Fig. 1, this work combines BiLSTM, GridLSTM and the attention mechanism layer to achieve better prediction performance. BiLSTM is used to train their parameters in both forward and backward directions to understand the sequence context. Two-dimensional GridLSTM has recurrent connections in the dimension of depth in addition to the temporal dimension. VAMBiG stacks a GridLSTM layer in the middle of two BiLSTM layers. First, the VMD and the adaptive SG filter are used in the data preprocessing to improve the feature extraction and the denoising performance, respectively. Then, we adopt the Min-Max scaler to lessen the scale of the raw data. Then, the processed sequence data is inputted into the first BiLSTM layer. Furthermore, we adopt the attention mechanism after the GridLSTM Layer and the second BiLSTM one. Finally, the output passes through a full connection layer to yield the prediction value.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT FILTERS

Methods	R^2			**RMSLE**			**RMSE**		
	Workloads	CPU	RAM	Workloads	CPU	RAM	Workloads	CPU	RAM
Median filter	0.920	0.876	0.434	0.546	0.498	0.797	127.4	10.1	10.5
Average filter	0.898	0.948	0.682	1.254	0.604	0.635	143.9	6.5	7.9
ASG filter	**0.979**	**0.964**	**0.919**	**0.508**	**0.429**	**0.496**	**62.5**	**5.5**	**3.9**

IV. EXPERIMENTAL RESULTS

We present extensive experiments to verify the efficiency of VAMBiG with two real-world datasets. We compare VAMBiG with its typical peers in terms of prediction accuracy.

A. Dataset and Experimental Setup

The experiments are performed on a server with 8 GB memory, and Intel® Core™ i7-7500 U CPUs with 4-core 2.70 GHz processor and another server with 32 GB memory, and an Nvidia RTX2080Ti GPU. We choose the cluster-usage traces from Google's and Alibaba's compute clusters to obtain characteristics of actual task or resource usage data. The former dataset includes 672,003 jobs and 25,462,157 tasks in 29 days. In this cluster data, work arrives as the form of jobs. Each job is comprised of multiple tasks. A task represents a Linux program that can be run on a machine. Our work adopts VAMBiG to predict time series sequences of workloads and resource usage. We divide 29 days into 20,880 time slots and the length of each time slot is two minutes. The time series data of workloads and resource usage, e.g., CPU and Random Access Memory (RAM) usage, are counted by analyzing the timestamp in each task. Then, three time series are obtained, as shown in Fig. 1 in the supplementary file, available online.

Alibaba cluster traces include static and runtime information from 4,000 machines, 9,000 online services, and 4,000,000 batch jobs in eight days. Users submit batch jobs, e.g., MapReduce and machine learning ones to cluster servers. We divide eight days into 10,097 time slots, each of which lasts for two minutes. Finally, the workload and resource usage time series from the Alibaba cluster are obtained and shown in Fig. 2 in the supplementary file, available online.

In the data preprocessing, we adopt VMD to break down the primary workload time series and resource usage into multiple modes. In our experiments, the primary time series data is divided into three modes.

Additionally, several filters are further compared to verify the performance of the adaptive SG (ASG) filter. This section selects four different filters (Median filter, Average filter, ASG filter) and three evaluation metrics. These filters are adopted for IMFs divided from workloads and resource usage. The prediction results with different filters are listed in Table II. It is observed that the performance of the ASG filter is much better than Median and Average filter.

To determine the optimal combination of VAMBiG parameters, we conduct multiple trials systemically. We choose Google's CPU usage (Data 1) and Alibaba's workload (Data 2) time series as the experimental data in the following

TABLE III
TRAINING RESULTS OF DIFFERENT EPOCH VALUES (α) AND BATCH SIZES (β) WITH GOOGLE'S CPU USAGE AND ALIBABA'S WORKLOAD TIME SERIES

β \ α	Data 1				Data 2			
	100	250	1000	2000	100	250	1000	2000
50	1.054	0.771	0.448	0.851	0.253	0.262	0.290	0.324
100	1.403	0.603	0.501	0.633	0.260	0.241	0.306	0.288
200	0.766	0.504	**0.429**	0.444	0.305	0.246	0.305	0.313
500	1.042	0.832	0.574	0.544	0.316	0.286	**0.227**	0.291
1000	1.633	1.012	0.518	0.571	0.322	0.311	0.245	0.243
2000	0.922	0.523	0.601	0.954	0.372	0.307	0.271	0.297

TABLE IV
PERFORMANCE COMPARISON OF DIFFERENT OPTIMIZERS

Optimizer	Loss Value	
	Data 1	Data 2
SGD	0.0014	0.0082
Adagrad	0.0026	0.0081
Adadelta	0.0068	0.0078
RMSprop	9.0734e-06	9.3516e-04
Adam	**4.6706e-06**	8.5593e-04
Nadam	6.1573e-06	**8.3851e-04**

TABLE V
PERFORMANCE COMPARISON OF DIFFERENT TIME STEP LENGTH

Time Step Length	Loss Value	
	Data 1	Data 2
10	1.1202e-05	0.0025
20	4.5286e-06	0.0011
30	6.2259e-06	8.7444e-04
40	4.3330e-06	7.0712e-04
50	5.3996e-06	8.1668e-04
60	**4.0318e-06**	**6.7139e-04**
70	5.0274e-06	6.7139e-04

TABLE VI
PERFORMANCE COMPARISON OF DIFFERENT ACTIVATION FUNCTIONS

Activation Function	Loss Value	
	Data 1	Data 2
Sigmoid	6.1814e-06	9.5426e-04
Tanh	6.9580e-06	8.7588e-04
Relu	**4.9472e-06**	8.5664e-04
Selu	6.0155e-06	**8.3274e-04**
Elu	8.1068e-06	9.7584e-04

TABLE VII
FINAL SETTING OF VAMBIG PARAMETERS FOR ALIBABA'S WORKLOAD

Parameters	Values	Description
X	60	Network input
Y	1	Network output
Structure	[60, 30, 25, 10, 1]	Hidden state number
Optimizer	Nadam	Optimization function
Batch size	500	Number of samples
Epoch value	1000	Iteration number
R	3	Number of VMD modes

TABLE VIII
FINAL SETTING OF VAMBIG PARAMETERS FOR GOOGLE'S CPU USAGE

Parameters	Values	Description
X	60	Network input
Y	1	Network output
Structure	[60, 35, 45, 30, 1]	Hidden state number
Optimizer	Adam	Optimization function
Batch size	200	Number of samples
Epoch value	1000	Iteration number
R	3	Number of VMD modes

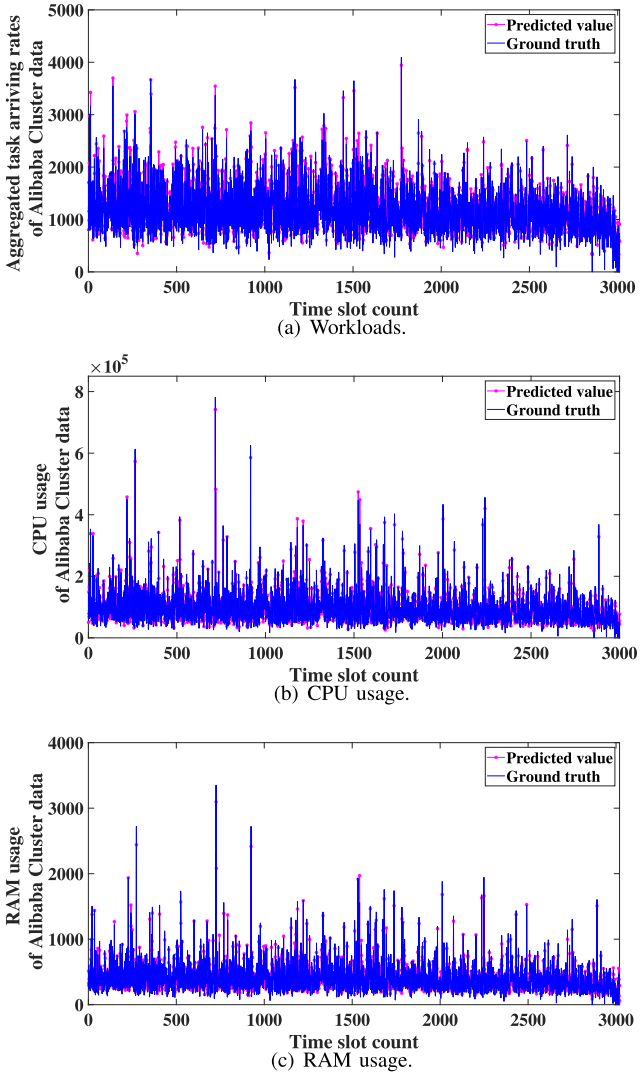

Fig. 2. Prediction results for workload and resource usage time series with VAMBiG.

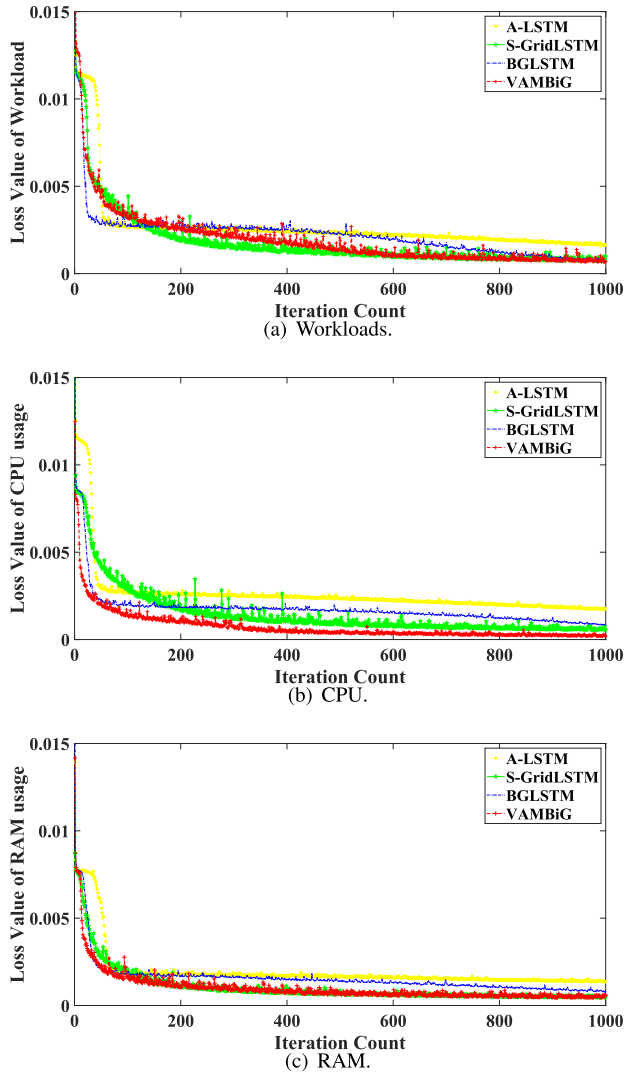

Fig. 3. Loss values of VAMBiG and other three benchmark methods for prediction of three Alibaba cluster datasets.

experiments. Since epoch value (α) and batch size (β) are critical and adjustable during model training, we choose the two parameters. The training results are shown in Table III. VAMBiG achieves the best prediction accuracy when α=1000 and $\beta = 200$ ($\alpha = 1000$ and $\beta = 500$) with Google's CPU usage (Alibaba's workload) time series. Furthermore, we select the optimizer of Adam and Nadam during the model training. The comparison of six optimizers is shown in Table IV. The time step length of the input data of VAMBiG is set to 60 when the loss value is lowest in Table V. The comparison of different activation functions is listed in Table VI.

Parameter settings of VAMBiG are given in Tables VII and VIII. X and Y denote input and output dimensions. Structure shows the hidden state number in each layer. Optimizer shows an optimization function to conduct gradient descent. Batch size shows the number of samples in each batch. Epoch value is the number of iterations. Besides, $R = 3$, i.e., the original sequence is decomposed into three IMFs.

B. Benchmarks

This work selects three baseline prediction methods for comparison to demonstrate the prediction performance our proposed VAMBiG.

1) *LSTM.* LSTM is characterized by its ability to capture long-term dependencies of time series. The parameter setting of LSTM is demonstrated in detail by (2).

TABLE IX
PERFORMANCE COMPARISON OF DIFFERENT METHODS WITH GOOGLE DATA

Methods	R^2			**RMSLE**			**RMSE**		
	Workloads	CPU	RAM	Workloads	CPU	RAM	Workloads	CPU	RAM
LSTM	0.912	0.887	0.648	0.851	0.819	0.517	133.3	9.7	8.3
BiLSTM	0.943	0.945	0.728	0.633	0.802	0.529	107.5	6.8	7.3
GridLSTM	0.920	0.801	0.600	0.803	0.763	0.550	173.9	12.9	8.9
A-LSTM	0.915	0.890	0.722	0.669	0.639	0.805	133.8	9.6	7.4
A-BiLSTM	0.948	0.920	0.833	0.630	0.502	0.612	102.5	8.2	5.7
A-GridLSTM	0.922	0.848	0.900	0.799	0.979	0.802	166.2	9.1	7.6
S-LSTM	0.949	0.922	0.885	0.580	0.661	0.516	101.3	8.0	4.7
S-BiLSTM	0.966	0.941	0.914	0.577	0.561	0.527	83.2	7.1	4.8
S-GridLSTM	0.945	0.915	0.885	0.559	0.701	0.732	105.6	8.4	4.5
VAMBiG	**0.979**	**0.964**	**0.919**	**0.508**	**0.429**	**0.496**	**62.5**	**5.4**	**3.9**

TABLE X
PERFORMANCE COMPARISON OF DIFFERENT METHODS WITH ALIBABA DATA

Methods	R^2			**RMSLE**			**RMSE**		
	Workloads	CPU	RAM	Workloads	CPU	RAM	Workloads	CPU	RAM
LSTM	0.630	0.741	0.729	0.407	0.352	0.339	235.8	27496.5	123.7
BiLSTM	0.661	0.764	0.748	0.447	0.368	0.359	220.3	26256.4	119.3
GridLSTM	0.734	0.742	0.752	0.377	0.393	0.342	195.2	27427.1	118.2
A-LSTM	0.736	0.762	0.748	0.360	0.359	0.337	194.4	26388.9	119.1
A-BiLSTM	0.720	0.767	0.780	0.424	0.367	0.348	199.9	26066.3	122.6
A-GridLSTM	0.732	0.749	0.759	0.314	0.353	0.346	195.7	27089.6	113.6
S-LSTM	0.883	0.862	0.891	0.330	0.301	0.298	129.0	20098.1	78.8
S-BiLSTM	0.866	0.852	0.899	0.327	0.314	0.288	138.6	16991.4	75.0
S-GridLSTM	0.862	0.864	0.887	0.355	0.319	0.294	140.5	15909.7	79.7
VAMBiG	**0.913**	**0.917**	**0.909**	**0.227**	**0.357**	**0.277**	**126.7**	**13992.1**	**71.6**

2) *BiLSTM.* As an improved variant LSTM, BiLSTM combines two LSTMs as forward and backward LSTM layer. The forward LSTM processes the time series data from $t = 1$ to T, the backward from $t = T$ to 1, respectively. Thus, the BiLSTM can capture features from the past and future information. Each LSTM cell structure of BiLSTM is the same as that of the ordinary LSTM. The forward layer with hidden state $\overrightarrow{h}_t$ is calculated based on its previous hidden states $\overrightarrow{h}_{t-1}$ and the input at the current time step x_t. The backward layer with hidden state $\overleftarrow{h}_t$ is calculated based on its future hidden states $\overleftarrow{h}_{t+1}$ and the input at the current time step x_t. Then, the $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are concatenated into the hidden layer state of BiLSTM at time step t, denotes as H_t. They are computed as follows:

$$\begin{aligned}
\overrightarrow{h}_t &= \text{LSTM}\left(\overrightarrow{h}_{t-1}, x_t, c_{t-1}\right) \quad t\in[1,T] \\
\overleftarrow{h}_t &= \text{LSTM}\left(\overleftarrow{h}_{t+1}, x_t, c_{t+1}\right) \quad t\in[T,1] \\
H_t &= [\overrightarrow{h}_t, \overleftarrow{h}_t].
\end{aligned} \tag{16}$$

3) *GridLSTM [14].* Unlike LSTM, GridLSTM contains recurrent neural network connections along different dimensions. In time series prediction, Grid LSTM has cells along two dimensions, the temporal one of the time series itself and the vertical one along the depth. This structure improves learning capacity of LSTM.

C. Evaluation Metrics

There are three performance evaluation metrics used to compare the prediction accuracy in our experiments, i.e., R^2 [31], Root Mean Squared Logarithmic Error (RMSLE) [32], and Root Mean Square Error (RMSE) [33]. Here, $\hat{y}_i$ denotes the predicted value and y_i denotes the true one. $\bar{y}$ denotes the average value of all the samples of true values. n denotes the number of samples. R^2 denotes the goodness of fit between the predicted value and the true one. When all the predicted values equal the true ones, the prediction performance is perfect. In this case, $R^2 = 1$. $R^2 \in (-\infty, 1]$ and it is given as:

$$R^2 = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (y_i - \bar{y})^2}. \tag{17}$$

RMSE averages the sum of squares of the difference between all predicted values and true ones. Then, RMSE takes the square root of the average value. When it is closer to 0, the prediction accuracy is higher. It ranges from 0 to $+\infty$, and it is given as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}. \tag{18}$$

RMSLE is suitable for assessing whether there are large outliers in the predicted data. Compared with RMSE, RMSLE is less likely to be dominated by large values in the data. It is in the range of $[0, +\infty]$ and it is given as:

$$\text{RMSLE} = \sqrt{\frac{1}{n}\left[\sum_{i=1}^{n} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2\right]}. \tag{19}$$

D. Prediction Results

We choose the first 70% of time series data as training data and the remaining 30% as test one. Fig. 2(a)–(c) illustrate the predicted results and actual ones of workloads, CPU and RAM usage with VAMBiG, respectively.

Tables IX and X show the performance comparison of VAMBiG and different methods including LSTM, BiLSTM, GridLSTM, A-LSTM, A-BiLSTM, A-GridLSTM, S-LSTM, S-BiLSTM, and S-GridLSTM. The A- means that the prediction model adopts the attention mechanism. The S- means that the adaptive SG filter is applied in the data preprocessing stage. It is observed from Table IX that the adaptive SG filter improves the prediction accuracy. Besides, the attention mechanism also enhances the performance of prediction with respect to different evaluation metrics. Among all these methods, VAMBiG achieves the best performance in terms of RMSLE. Therefore, VAMBiG outperforms other peers by combining advantages of the adaptive SG filter, BiLSTM, GridLSTM and the attention mechanism.

Fig. 3 compares loss values of A-LSTM, S-GridLSTM, BGLSTM [33] and SABG for workloads, CPU, and RAM time series, respectively. The loss values decrease as epoch values increase. It is observed that the loss values of VAMBiG are lower than other models after iteration 800. VAMBiG has better modeling ability than other variants of LSTM. Therefore, VAMBiG outperforms other variants of LSTM given the same structure and parameter settings.

E. Discussions With Real-Time-Based Strategies

In terms of reducing energy consumption in CDCs, existing studies has proposed a series of algorithms on task scheduling. The simulated annealing algorithm [34] assigns each pending request to the most suitable working node of underlying resources, which guarantees the minimum response time for each request. The study in [35] dynamically exploits proactive and reactive scheduling methods for scheduling real-time, aperiodic and independent tasks. In summary, these approaches develop efficient scheduling strategies for time-sensitive and high-priority tasks while optimizing their energy cost and quality of service. In contrast, workload and resource prediction in our work allows CDC providers to estimate of the number of incoming tasks and resource usage in the future. It allows CDC providers to flexibly and more accurately allocate resources in CDCs in advance. Thus, the two strategies can complement each other for more accurate and efficient usage of resources in CDCs.

V. Conclusion

Achieving accurate forecasting of highly variable workloads and resource usage in a cloud data center is a critical and challenging problem for service providers. This work first adopts variational mode decomposition and an adaptive Savitzky-Golay filter to extract non-linear features and achieve better denoising performance under low distortion conditions. Then, the processed data are input into a hybrid prediction model to obtain the prediction results. Specifically, the model named VAMBiG integrates Variational mode decomposition (VMD), an Adaptive Savitzky-Golay filter, the Multi-head attention mechanism, Bidirectional and Grid LSTM networks. It adopts the attention mechanism to improve the extraction ability of important sequence information. It captures bidirectional relations and encodes implicit information from forward data points to backward ones with bidirectional LSTM. In addition, it explores the dimension information of time and depth in a time series with GridLSTM. Experimental results with real-life datasets from Google and Alibaba cluster traces demonstrate that VAMBiG achieves more accurate prediction results than several widely used methods.

Our next work plans to extend our work in two aspects. Although VAMBiG performs well on Google and Alibaba traces, we should apply it to different real-world workload datasets. We also plan to further address a prediction problem of multivariate and high-dimensional time series.

References

[1] H. Yuan, J. Bi, and M. Zhou, "Multiqueue scheduling of heterogeneous tasks with bounded response time in hybrid green IaaS clouds," *IEEE Trans. Ind. Inform.*, vol. 15, no. 10, pp. 5404–5412, Oct. 2019.
[2] H. Yuan, J. Bi, and M. Zhou, "Energy-efficient and QoS-Optimized adaptive task scheduling and management in clouds," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 2, pp. 1233–1244, Apr. 2022.
[3] A. Amokrane, R. Langar, M. F. Zhani, R. Boutaba, and G. Pujolle, "Greenslater: On satisfying green SLAs in distributed clouds," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 3, pp. 363–376, Sep. 2015.
[4] J. Guo et al., "Who limits the resource efficiency of my datacenter: An analysis of Alibaba datacenter traces," in *Proc. IEEE/ACM 27th Int. Symp. Qual. Serv.*, Phoenix, AZ, USA, 2019, pp. 1–10.
[5] X. Tang, X. Liao, J. Zheng, and X. Yang, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Comput.*, vol. 21, pp. 1581–1593, Feb. 2018.
[6] Y. Lu, L. Liu, J. Panneerselvam, X. Zhai, X. Sun, and N. Antonopoulos, "Latency-based analytic approach to forecast cloud workload trend for sustainable datacenters," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 3, pp. 308–318, Third Quarter 2020.
[7] P. Singh, P. Gupta, and K. Jyoti, "TASM: Technocrat ARIMA and SVR model for workload prediction of web applications in cloud," *Cluster Comput.*, vol. 22, no. 2, pp. 619–633, Jun. 2019.
[8] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Gener. Comput. Syst.*, vol. 81, pp. 41–52, Apr. 2018.
[9] L. Ruan, Y. Bai, S. Li, S. He, and L. Xiao, "Workload time series prediction in storage systems: A deep learning based approach," *Cluster Comput.*, vol. 26, pp. 1–11, Jan. 2021.
[10] H. Xue, D. Q. Huynh, and M. Reynolds, "PoPPL: Pedestrian trajectory prediction by LSTM with automatic route class clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 77–90, Jan. 2021.
[11] Y. Lu, L. Liu, J. Panneerselvam, B. Yuan, J. Gu, and N. Antonopoulos, "A GRU-Based prediction framework for intelligent resource management at cloud data centres in the age of 5G," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 486–498, Jun. 2020.
[12] K. Dragomiretskiy and D. Zosso, "Variational mode decomposition," *IEEE Trans. Signal Process.*, vol. 62, no. 3, pp. 531–544, Feb. 2014.
[13] A. Yuille and J. Wang, "Semantic part segmentation using compositional model combining shape and appearance," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 1788–1797.
[14] H. Fei and F. Tan, "Bidirectional grid long short-term memory (BiGridLSTM): A method to address context-sensitivity and vanishing gradient," *Algorithms*, vol. 11, no. 11, pp. 1–16, Oct. 2018.
[15] J. Bi, H. Ma, H. Yuan, K. Xu, and M. Zhou, "Adaptive prediction of resources and workloads for cloud computing systems with attention-based and hybrid LSTM," in *Proc. Int. Conf. Syst. Man Cybern.*, Prague, Czech Republic, 2022, pp. 1–6.

[16] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Fourth Quarter 2015.
[17] K. Yunus, T. Thiringer, and P. Chen, "ARIMA-Based frequency-decomposed modeling of wind speed time series," *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 2546–2556, Jul. 2016.
[18] L. Ghelardoni, A. Ghio, and D. Anguita, "Energy load forecasting using empirical mode decomposition and support vector regression," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 549–556, Mar. 2013.
[19] Z. Huang, J. Peng, H. Lian, J. Guo, and W. Qiu, "Deep recurrent model for server load and performance prediction in data center," *Complexity*, vol. 2017, no. 99, pp. 1–10, Nov. 2017.
[20] S. Ouhame, Y. Hadi, and A. Ullah, "An efficient forecasting approach for resource utilization in cloud data center using CNN-LSTM model," *Neural Comput. Appl.*, vol. 33, pp. 10043–10055, Mar. 2021.
[21] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, Apr. 2020.
[22] M. -S. Ko, K. Lee, J. -K. Kim, C. W. Hong, Z. Y. Dong, and K. Hur, "Deep concatenated residual network with bidirectional LSTM for one-hour-ahead wind power forecasting," *IEEE Trans. Sustain. Energy*, vol. 12, no. 2, pp. 1321–1335, Apr. 2021.
[23] X. Li, X. Ma, F. Xiao, C. Xiao, F. Wang, and S. Zhang, "Time-series production forecasting method based on the integration of bidirectional gated recurrent unit (Bi-GRU) network and Sparrow Search Algorithm (SSA)," *J. Petroleum Sci. Eng.*, vol. 208, pp. 1–13, Jan. 2022.
[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[25] L. Lv, Z. Wu, J. Zhang, L. Zhang, Z. Tan, and Z. Tian, "A VMD and LSTM based hybrid model of load forecasting for power grid security," *IEEE Trans. Ind. Inform.*, vol. 18, no. 9, pp. 6474–6482, Sep. 2022.
[26] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Anal. Chem.*, vol. 36, pp. 1627–1639, Jul. 1964.
[27] H. Huang, S. Hu, and Y. Sun, "A discrete curvature estimation based low-distortion adaptive Savitzky–Golay filter for ECG denoising," *Sensors*, vol. 19, no. 7, pp. 1–18, Apr. 2019.
[28] Freeman and Davis, "A corner-finding algorithm for chain-coded curves," *IEEE Trans. Comput.*, vol. C-26, no. 3, pp. 297–303, Mar. 1977.
[29] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 45, pp. 48–62, Sep. 2021.
[30] Y. Li, L. Zhang, Z. Lv, and W. Wang, "Detecting anomalies in intelligent vehicle charging and station power supply systems with multi-head attention models," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 555–564, Jan. 2021.
[31] Y. Song, Y. D. Wang, X. Hu, and J. Liu, "An efficient and explainable ensemble learning model for Asphalt pavement condition prediction based on LTPP dataset," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 22084–22093, Nov. 2022.
[32] J. Bi, X. Zhang, H. Yuan, J. Zhang, and M. Zhou, "A hybrid prediction method for realistic network traffic with temporal convolutional network and LSTM," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 3, pp. 1869–1879, Jul. 2022.
[33] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, vol. 424, pp. 35–48, Feb. 2021.
[34] X. Xu, L. Cao, X. Wang, X. Xu, L. Cao, and X. Wang, "Resource pre-allocation algorithms for low-energy task scheduling of cloud computing," *J. Syst. Eng. Electron.*, vol. 27, no. 2, pp. 457–469, Apr. 2016.
[35] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *J. Syst. Softw.*, vol. 99, pp. 20–35, Jan. 2015.

Jing Bi (Senior Member, IEEE) is currently an associate professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. She has more than 80 publications including journal and conference papers. Her research interests include distributed computing, cloud computing, large-scale data analysis, machine learning, and performance optimization. She was the recipient of the IBM Fellowship Award and the recipient of the Best Paper Award-Finalist in the 16th IEEE International Conference on Networking, Sensing and Control. She is now an associate editor of the *IEEE Access*.

Haisen Ma (Student Member, IEEE) received the BS degree in software engineering from Zhengzhou University, in 2021. He is currently working toward the master's degree with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. His research interests include cloud computing, data center, Big Data, time series prediction, machine learning, and deep learning.

Haitao Yuan (Senior Member, IEEE) received the PhD degree in computer engineering from the New Jersey Institute of Technology (NJIT), Newark, NJ, USA, in 2020. He is currently an associate professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include cloud computing, edge computing, data centers, Big Data, machine learning, deep learning, and optimization algorithms. He received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC.

Jia Zhang (Senior Member, IEEE) received the PhD degree in computer science from the University of Illinois at Chicago. She is currently the Cruse C. and Marjorie F. Calahan Centennial chair in engineering, professor with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in earth science.