# Data-Filtered Prediction With Decomposition and Amplitude-Aware Permutation Entropy for Workload and Resource Utilization in Cloud Data Centers

Haitao Yuan, *Senior Member, IEEE*, Qinglong Hu, *Student Member, IEEE*, Meijia Wang, *Student Member, IEEE*, Shen Wang, *Student Member, IEEE*, Jing Bi, *Senior Member, IEEE*, Rajkumar Buyya, *Fellow, IEEE*, Shuyuan Shi, Jinhong Yang, Jia Zhang, *Senior Member, IEEE*, and Mengchu Zhou, *Fellow, IEEE*

***Abstract*—In recent years, cloud computing has witnessed widespread applications across numerous organizations. Predicting workload and computing resource data can facilitate proactive service operation management, leading to substantial improvements in Quality of Service and cost efficiency. However, these data often exhibit nonlinearity, high volatility, and interdependencies across different categories, presenting challenges for accurate forecasting. Consequently, there is a critical need to develop a method that thoroughly and comprehensively analyzes all available data to forecast future trends effectively. This work proposes a novel integrated data-enhanced prediction model named SVAPI for achieving high-accuracy workload prediction in cloud computing systems. SVAPI employs the Savitzky–Golay filter, Variational mode decomposition, and the mode selection based on Amplitude-aware Permutation entropy for feature processing, whose features are subsequently utilized by Informer for multivariate joint analysis of the enhanced data, achieving high-precision prediction. Ablation and comparative experiments with advanced prediction models are conducted on the Google cluster trace and other typical datasets. Realistic data-driven results indicate that SVAPI improves the prediction accuracy by 37.7% compared to the original Informer, with each module contributing to the performance enhancement. Furthermore, compared with Autoformer, SVAPI enhances the prediction accuracy of workload, CPU, and memory by 65.6%, 66.9%, and 70.8%, respectively, demonstrating that SVAPI owns strong abilities in noise filtering, feature processing, and multivariate joint analysis for achieving higher prediction accuracy.**

***Index Terms*—Amplitude-aware permutation entropy (AAPE), cloud computing, deep learning, informer, Savitzky–Golay (SG) filter, variational mode decomposition (VMD).**

Received 20 June 2024; revised 21 September 2024 and 9 December 2024; accepted 30 December 2024. Date of publication 6 January 2025; date of current version 9 June 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62173013 and Grant 62473014; in part by the Beijing Natural Science Foundation under Grant L233005 and Grant 4232049; and in part by the Beihang World TOP University Cooperation Program. *(Corresponding author: Haitao Yuan.)*

Haitao Yuan, Meijia Wang, and Shen Wang are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn).

Qinglong Hu is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: qinglhu2-c@my.cityu.edu.hk).

Jing Bi is with the School of Software Engineering in the Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems Lab, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Shuyuan Shi is with the School of Integrated Circuit Science and Engineering, Beihang University, Beijing 100191, China (e-mail: smeshis@buaa.edu.cn).

Jinhong Yang is with the CSSC Systems Engineering Research Institute, Beijing 100036, China (e-mail: yangjinhong.66@163.com).

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

Mengchu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Digital Object Identifier 10.1109/JIOT.2024.3525301

## I. Introduction

DISTRIBUTED computing paradigms such as cloud computing [1], [2] and mobile edge computing [3] are introduced into the Internet of Things to provide computational resources, addressing the increasing requirement for computing power. Several typical applications, such as artificial intelligence, automatic target recognition, online games, social networks, and protein analysis, have emerged in recent years. However, due to the highly dynamic nature of computational service demands, these services often suffer from both under-provisioning and over-provisioning of resources, leading to substandard Quality of Service (QoS) and cost inefficiencies [4]. Therefore, proactive service operation management techniques are proposed to achieve desired runtime decision and allocation through workload prediction methods, ensuring QoS and cost efficiency [8].

Monitoring, analysis, planning, and execution are essential to managing service operations. The monitoring phase collects historical tasks, CPU utilization, memory utilization, and QoS metrics in cloud computing services. The analysis phase includes predicting future workload and assessing whether services meet the required criteria. The planning phase entails generating task offloading strategies with evolutionary or reinforcement learning algorithms to meet QoS requirements while minimizing the system cost. The execution phase involves offloading tasks to distributed devices for faster computation.

Therefore, workload prediction plays a crucial role in guaranteeing system performance and resource scheduling and ensuring that offloading strategies have adequate time to be determined by evolutionary algorithms [5] or reinforcement learning algorithms [7]. The phase of workload prediction collects historical tasks, CPU utilization, memory utilization, and QoS metrics in cloud computing services. It predicts future workload, enabling proactive service operations to deliver desired QoS and cost efficiency [8]. Current studies on workload prediction mainly rely on classical models or deep-learning-based ones. However, few of them jointly consider the workload and resource utilization prediction, which have significantly different characteristics. They fail to explore interconnections among multiple features and cannot reasonably handle long-term dependencies in long-time series sequences.

Long-term dependencies are complex to capture because dependency coupling between output and input over long ranges is complex [6]. Unlike above-mentioned studies, this work proposes a novel prediction model named SVAPI, which integrates the Savitzky–Golay (SG) filter [9], variational mode decomposition (VMD) [10], Amplitude-aware Permutation Entropy (AAPE) [11], and Informer [12] to enhance prediction performance. SVAPI addresses these challenges by adding a ProbSparse self-attention mechanism, a self-attention distilling operation, and a generative style decoder. Thus, SVAPI has low time complexity and memory usage, the ability to handle extremely long input time series, and fast inference speed for long sequences, simultaneously avoiding spreading cumulative errors during the inference. Realistic datasets, such as arriving tasks in the Google cluster trace [13], are adopted to evaluate the performance of SVAPI. Experimental results illustrate that SVAPI outperforms Informer and Autoformer [14], achieving more accurate prediction.

The remainder of this article is organized as follows. Section II discusses the differences between our proposed SVAPI and the related work. Section III defines the prediction problem and introduces the main components of the proposed SVAPI. Section IV gives the performance evaluation results of SVAPI with large-scale Google cluster traces. Section V concludes this work and provides the future work.

## II. Related Work

Current workload and resource utilization prediction methods in cloud data centers (CDCs) are mainly designed using classical or deep learning models. This section discusses the differences between SVAPI and related works in the literature. Table I shows the comparison among this work and its state-of-the-art studies.

### A. Prediction Methods With Classical Models

There are many studies on prediction methods with classical models in cloud computing systems. Saxena et al. [15] efficiently predicted resource utilization of servers and realize the load balance, thus minimizing the number of powered servers and maximizing resource utilization. An online system for resource prediction is designed for each virtual machine for the fewest violations of service level agreements. However, it uses a simple three-layer neural network to predict resource utilization, including input, hidden, and output layers. Kumara et al. [16] adopted a multilayer perception neural network to predict the performance of configurable cloud applications. Yet, their network is simple and fails to effectively predict the time series of workload and resource utilization. Xie et al. [17] combined triple exponential smoothing methods and an auto-regressive moving average model to predict the sequence of the dynamic resource utilization of container workload by capturing its nonlinear and linear features. It is fast but fails to capture long-term time dependency. Singh et al. [18] proposed an evolutionary quantum neural network for workload estimation in CDCs. It adopts the efficiency of quantum computing by transforming workload into qubits, which are propagated through the network to predict the workload proactively. A self-adaptive differential evolution algorithm is used to optimize network weights of qubits. However, it also fails to investigate the long-term time dependency. Moradi et al. [19] presented an online learning method for the performance prediction of applications running repetitively in multitenant clouds by using regression and neural network models. However, it ignores the noise data in the time series. Kim et al. [4] created an ensemble prediction method of real cloud application workloads using multiclass regression. It combines the power of multiple workload prediction models and utilizes their short and long-term fluctuations, thus providing predictive cloud resource management. However, this method fails to capture the long-term time dependency of the workload time series. Ilager et al. [20] designed a gradient-boosting model for predicting the host temperature by using thermal variations for hyper-scale CDCs. Then, a dynamic scheduling algorithm is proposed to minimize host peak temperature, thereby reducing cooling costs and increasing reliability. However, it is unsuitable for predicting long-term workload and resource utilization time series.

Instead of using classical or simpler machine learning approaches, we adopt a deep learning model called SVAPI for predicting different types of data, including workload, CPU, and memory, with high volatility, nonlinearity, and complicated interdependencies. SVAPI comprehensively analyzes trends of heterogeneous data in cloud computing systems.

### B. Deep-Learning-Based Prediction Methods

Recent years have witnessed the fast growth of prediction methods with deep learning in CDCs. Chen et al. [21] designed a hierarchical and fuzzy neural network to predict the number of requisite cloud services for consumers by coalescing fuzzy logic and neural representation of the original data. However, their network is relatively simple and cannot capture the long-term dependency of the data. Bi et al. [22] proposed a prediction method that combines frequency-enhanced channel attention and a decomposed Transformer for capturing frequency domain patterns in the resource utilization time series. It adopts a self-attention memory mechanism to handle long-term dependencies, yet its computation is quadratic and higher. Maroudis et al. [23] adopted a graph neural network with graph representations and associativity of clients for predicting violations of service level agreements, which

includes three components of preprocessing, input formulation, and composite prediction. However, it fails to predict the workload and resource utilization time series and cannot handle long-term dependencies. Ruan et al. [24] presented a trend prediction approach for turning points of cloud workload time series with a deep learning model enhanced by cloud features, thereby realizing high resource utilization and QoS. However, it only focuses on the turning points of the highly variable workload time series. Yuan et al. [25] proposed a prediction approach combining the SG filter, VMD, bi-directional and grid LSTM to predict resource utilization and workload in CDCs. It investigates temporal and depth features of fluctuating time series for predicting workload and resources. However, it cannot well handle the long-term dependencies in the time series data. Bi et al. [26] developed a hybrid method combining a temporal convolutional network, multihead self-attention, and a bi-directional gated recurrent unit (GRU) network for predicting attacks. It also designs a hybrid optimization algorithm that combines genetic operations, simulated annealing, and particle swarm optimization to determine the model hyperparameters. Yet, it focuses on the prediction of network attacks. Gao et al. [27] designed a failure prediction method with multilayer bidirectional LSTM to detect failures of tasks and jobs in clouds, thereby predicting whether tasks and jobs are completed or failed. However, it fails to predict future resource utilization and workload time series.

The above-mentioned deep learning methods adopt classical neural networks, Transformer-based, GRU-based, or LSTM-based variants. However, their computation of self-attention is high and they suffer from the memory bottleneck in stacking layers for long input sequences. SVAPI adopts the SG filter to eliminate noise and VMD to capture the temporal characteristics. It utilizes AAPE to select the decomposed modes with more informative features. Then, it adopts the ProbSparse self-attention mechanism to significantly decrease the time complexity and memory usage and efficiently process the extremely long input sequences. It adopts the self-attention distilling method to reduce the cascading layer input. It predicts the long time series at one forward operation, thereby increasing the inference speed of long-sequence predictions.

## III. Proposed Methodology

This section presents the definition of the problem and an overview of the proposed SVAPI. Four components, including the SG filter, VMD, AAPE, and Informer, are integrated into SVAPI to enhance the prediction accuracy for workloads and resource utilization in cloud computing systems.

### A. Problem Definition

This work defines the prediction problem for a dataset $\mathcal{D} = \{S^1, S^2, \ldots, S^m\}$. $m$ denotes the number of features in $\mathcal{D}$, and $S^i$ denotes the sequence of feature $i$, and $S^i = \{s_1^i, s_2^i, \ldots, s_L^i\}$. $s_l^i$ signifies the $l$th value in sequence $S^i$, and $L$ denotes the length of the sequence. The Input-$I$-predict-$O$ paradigm utilizes $m$ feature sequences of length $I$ to predict a single future sequence of length $O$, i.e.,

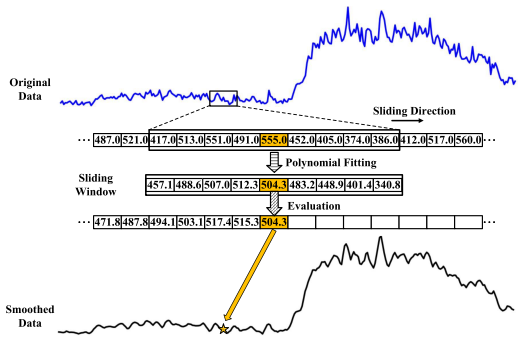



Fig. 1. Smoothing process of the SG filter.

$$P = \{p_{q+1}, p_{q+2}, \ldots, p_{q+O}\} = F\left(\{\vec{S_q^i} = \{s_{q-I+1}^i, \ldots, s_q^i\} | 1 \le i \le m\}\right) \quad (1)$$

where $P$ denotes the prediction sequence of the target feature, $F(\cdot)$ denotes the prediction process of SVAPI, and $\vec{S_q^i}$ denotes the subsequence of length $I$ taken from $S^i$ starting from the $q$th point. We aim to minimize the error between predicted and ground truth values.

To concisely show the process of handling sequences within each component, let $Z$ represent a sequence $S^i$ currently undergoing processing from the dataset $\mathcal{D}$ and $Z = \{z_1, \ldots, z_L\}$.

### B. SG Filter

The smoothed data is more conducive for capturing the variations in the data trend by the prediction model [28]. Consequently, this work adopts the SG filter to remove the noise of data to smooth it. The SG filter fits a polynomial to a sliding window and uses the fitted value at the center point as the filtered value [29]. This iterative process spans the entire dataset, resulting in a filtered sequence. Fig. 1 illustrates the smoothing process of the SG filter. In Fig. 1, the polynomial fitting is realized in (5).

Given an input sequence $Z$, a sliding window is established with $z_u$ as its center and a width of $w = 2h+1$, where $u$ ranges from $h+1$ to $L-h$. The sliding window is defined as a 1-D array, denoted as $Z(u)$, and $Z(u) = [z_{u-h}, \ldots, z_u, \ldots, z_{u+h}]^T$. Within this window, an $R$-order polynomial (2) is employed to fit each element, i.e.,

$$p(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_R x^R, x \in [-h, h]. \quad (2)$$

Then, the polynomial fitting for the sliding window $Z(u)$ can be represented as

$$\begin{aligned} Z(u) &= \begin{bmatrix} z_{u-h} & \ldots & z_u & \ldots & z_{u+h} \end{bmatrix}^T \\ &= \begin{bmatrix} p(-h) & \ldots & p(0) & \ldots & p(h) \end{bmatrix}^T \\ &= \begin{bmatrix} 1 & (-h) & (-h)^2 & \ldots & (-h)^R \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 0 & 0^2 & \ldots & 0^R \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & h & h^2 & \ldots & h^R \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_R \end{bmatrix} \\ &= W \cdot C \end{aligned} \quad (3)$$

TABLE I
COMPARISON AMONG THIS WORK AND ITS STATE-OF-THE-ART STUDIES

| Reference | Non-linearity | Data filtering | Large data set | Data-enhanced | Multi-feature | Frequency domain analysis |
|---|---|---|---|---|---|---|
| [18] | ✓ | × | ✓ | × | × | × |
| [19] | × | × | × | × | ✓ | × |
| [20] | ✓ | × | × | × | × | × |
| [21] | × | × | ✓ | × | × | ✓ |
| [22] | ✓ | × | × | × | × | ✓ |
| [4] | × | × | × | × | ✓ | × |
| [23] | ✓ | × | × | × | ✓ | × |
| [24] | × | × | × | × | ✓ | × |
| [25] | ✓ | ✓ | × | × | × | ✓ |
| [26] | ✓ | × | × | × | ✓ | × |
| [10] | ✓ | ✓ | × | × | ✓ | × |
| [27] | ✓ | ✓ | × | × | ✓ | × |
| [28] | ✓ | ✓ | × | × | ✓ | × |
| [29] | ✓ | × | ✓ | × | ✓ | × |
| **This work** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

where $W$ denotes a matrix with the size of $w \times R$, and $C$ represents a matrix with the size of $R \times 1$.

The least squares method [30] is utilized to calculate the coefficient vector $\hat{C}$ by minimizing the mean square error $E$ between the fitted values and the ground-truth ones. Then, $E$ and $\hat{C}$ are given as

$$E = \|W \cdot C - Z(u)\|^2 = \sum_{x=-h}^{h} (p(x) - z_{u+x})^2 \tag{4}$$

$$\hat{C} = \left(W^{\mathrm{T}} \cdot W\right)^{-1} \cdot W^{\mathrm{T}} \cdot Z(u). \tag{5}$$

Consequently, the fitted window (polynomial) is represented as $\tilde{Z}(u)$ and $\tilde{Z}(u) = W \cdot \hat{C}$. The center point of the fitted window is denoted as $p(0)$, which is the filtered value $\tilde{z}_u$.

Since the window can only compute filtered values for center points, only a portion of the filtered values $\{\tilde{z}_{h+1}, \tilde{z}_{h+2}, \ldots, \tilde{z}_{L-h}\}$ can be obtained. To acquire filtered values for the entire sequence, we insert $h$ initial values before the first item of the original sequence and $h$ trailing values after the last item. This results in an interpolated sequence with the length of $L+2h$. Subsequently, the filtering is applied to this interpolated sequence to yield a complete and filtered sequence of length $L$, denoted as $\tilde{Z}$ and $\tilde{Z} = \{\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_L\}$.

### C. Variational Mode Decomposition

After the data smoothing, we adopt VMD to analyze temporal characteristics of the original sequences in the frequency domain to improve data representation. VMD dynamically determines a sequence's frequency domain segmentation, precisely aligning each mode's central frequency and finite bandwidth through the iterative optimization [31]. This process achieves effective separation of sequence components. Consequently, applying VMD to filtered sequences generates multiple modes, offering the model a broader range of captureable features. VMD has been effectively applied in signal processing in power systems, and it analyzes power quality problems by separating various frequency components of power signals, identifying transients and harmonics [33].

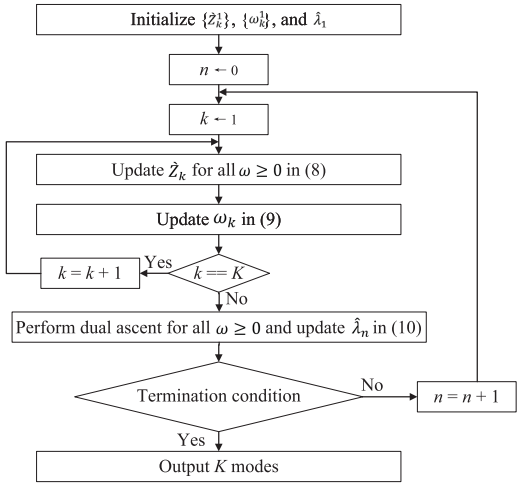



Fig. 2. Flow chart of VMD.

The flowchart of VMD is shown in Fig. 2. When a filtered sequence $\tilde{Z}$ is inputted, VMD determines the center frequencies of $K$ modes by seeking the optimal solution to a constrained variational problem formulated as

$$\min_{\{\check{Z}_k, \omega_k\}} \left\{ \sum_k^K \left\| \partial_t \left[ \left( \delta(t) + \frac{j}{\pi t} \right) \cdot \check{Z}_k(t) \right] e^{-j\omega_k t} \right\|_2^2 \right\}$$
$$\text{s.t.} \sum_k^K \check{Z}_k(t) = \tilde{Z}(t). \tag{6}$$

Here, $\check{Z}_k$ is the $k$th mode to be decomposed, and $\omega_k$ represents the corresponding center frequency. $\check{Z}_k(t)$ denotes point $t$ in $\check{Z}_k$. $\delta(t)$ is the Dirac function. $\partial_t$ denotes a partial derivative operation in time, i.e., index $t$ of sequences, and the symbol $j$ represents the imaginary unit.

A Lagrangian multiplier $\lambda$ and a quadratic penalty term $\alpha$ are introduced, transforming (6) into an unconstrained problem [32]. The quadratic penalty enhances the reconstruction fidelity, while Lagrangian multipliers guarantee strict fulfillment of constraints. Thus, the resulting augmented

Lagrangian function $\mathcal{L}$ is given as

$$\begin{aligned}\mathcal{L}\left(\left\{\check{Z}_k\right\},\{\omega_k\},\lambda\right)&=\alpha\sum_k^K\left\|\partial_t\left[\left(\delta(t)+\frac{j}{\pi t}\cdot\check{Z}_k(t)\right)\right]e^{-j\omega_k t}\right\|_2^2\\&+\left\|\tilde{Z}(t)-\sum_k^K\check{Z}_k(t)\right\|_2^2+\left\langle\lambda(t),\tilde{Z}(t)-\sum_k^K\check{Z}_k(t)\right\rangle.\end{aligned}\tag{7}$$

Subsequently, an iterative solution method, known as the alternate direction method of multipliers [34], is utilized to solve the transformed unconstrained problem and obtain $\lambda$, $\check{Z}_k$, and $\omega_k$. According to [36], modes $\check{Z}_k$ and their corresponding center frequencies $\omega_k$ are alternatively updated as

$$\grave{Z}_k^{n+1}(\omega)=\frac{\grave{Z}(\omega)-\sum_{f\neq k}\grave{Z}_f^n(\omega)+\frac{\grave{\lambda}^n(\omega)}{2}}{1+2\alpha\left(\omega-\omega_k^n\right)^2}\tag{8}$$

$$\omega_k^{n+1}=\frac{\int_0^\infty\omega\left|\grave{Z}_k^{n+1}(\omega)\right|^2\mathrm{d}\omega}{\int_0^\infty\left|\grave{Z}_k^{n+1}(\omega)\right|^2\mathrm{d}\omega}\tag{9}$$

where $n$ denotes the iteration number. $\grave{Z}_k^n$, $\grave{Z}$, and $\grave{\lambda}^n$ represent the Fourier transformation results of $\check{Z}_k^n$, $\tilde{Z}$, and $\lambda^n$, respectively. $\lambda$ in the Fourier domain is updated as

$$\grave{\lambda}^{n+1}(\omega)=\grave{\lambda}^n(\omega)+\tau\left(\grave{Z}(\omega)-\sum_k\grave{Z}_k^{n+1}(\omega)\right)\tag{10}$$

where $\tau$ denotes the noise tolerance. The above-mentioned iteration continues until convergence is achieved, defined as

$$\sum_{k=1}^K\frac{\left\|\grave{Z}_k^{n+1}(\omega)-\grave{Z}_k^n(\omega)\right\|^2}{\left\|\grave{Z}_k^n(\omega)\right\|^2}<\varepsilon\tag{11}$$

where $\varepsilon$ denotes the convergence threshold.

It is worth noting that numerical stability is guaranteed when VMD is applied to SG-filtered sequences rather than original sequences. This helps reduce the impact of noise in the original sequences. In addition, VMD computations are performed in advance at CDCs, which enables parallel processing in high-performance servers, significantly reducing computation time and ensuring real-time processing.

### *D. Amplitude-Aware Permutation Entropy*

VMD decomposes each feature in dataset $\mathcal{D}$ into $K$ modes, ultimately yielding $m\times K$ decomposed modes. However, an excessive number of features can lead to increased computational demands during the model training and may potentially cause overfitting [37]. To address this issue, similar to [41], this work designs an enhanced entropy method for assessing the significance of decomposed modes. Several entropy methods are available to measure the irregularity of time series, which include approximate entropy [38], sample entropy [39], fuzzy entropy [40], permutation entropy (PE) [41], and wavelet entropy [42]. However, each method has its disadvantages. PE has been applied in resource utilization forecasting for physical machines in data centers in the Alibaba Group [35].

PE relies on order relations among signal values or permutation patterns and compares the order of neighboring relative values. PE is computationally fast and robust to observational and dynamic noise. It is effective for many real-world signal and image processing applications. However, when a signal is symbolized, PE ignores the mean value of amplitudes and the differences between neighboring samples. Thus, vectors with different amplitudes may be assigned the same symbol. PE fails to address the issue precisely when two samples have equal amplitude values. Unlike PE, this work designs an amplitude-aware PE named AAPE to select the most informative $T$ modes from $K$ modes decomposed from each original sequence $S_i$, thereby enhancing the prediction accuracy.

We experimentally demonstrate that modes with higher AAPE are more beneficial for the model prediction. We select the top $T$ modes with the highest AAPE from $K$ modes, which are selected as the most informative features for reducing overfitting and computation loads. The main procedure of AAPE is given as follows. Given the embedded dimension $\eta$ and the time delay $D$, a sequence $Z=\{z_1,\ldots,z_L\}$ is reconstructed in the phase space, generating $L-(\eta-1)D$ reconstructed vectors marked as $Z_v$ and $Z_v=\{z_v,z_{v+D},\ldots,z_{v+(\eta-1)D}\}$, $v\in\{1,L-(\eta-1)D\}$. Next, $\eta$ elements in each vector are arranged in an ascending order. Then

$$Z_v=\{z_{v+(o_1-1)D}\leq z_{v+(o_2-1)D}\leq\cdots\leq z_{v+(o_\eta-1)D}\}\tag{12}$$

where $o_d$ denotes the position of the element before arrangement. Subsequently, $Z_v$ is mapped to $\theta_v$ where $\theta_v=[o_1,o_2,\ldots,o_\eta]$, representing one of $\eta!$ permutations of $\eta$ distinct symbols. The $g$th permutation is denoted as $\mu_g$. The relative frequency is then utilized to estimate the occurrence probability $\phi(\mu_g)$ of each permutation $\mu_g$ as follows:

$$\phi\left(\mu_g\right)=\frac{\sum_{v=1}^{L-(\eta-1)D}\varphi\left(\mu_g,\theta_v\right)}{L-(\eta-1)D}\tag{13}$$

where $\varphi(\mu,\theta)$ is the Kronecker delta function expressed as

$$\varphi(\mu,\theta)=\begin{cases}1, & \text{if } \mu=\theta\\ 0, & \text{if } \mu\neq\theta.\end{cases}\tag{14}$$

To incorporate the amplitude information of the inputted sequence, the average absolute $\mathcal{A}$ and the relative amplitude $\mathcal{R}$ of vectors are considered when calculating the probability of occurrence of each permutation pattern through relative frequency. The amplitude information $\mathcal{A}_v$ and $\mathcal{R}_v$ of vector $Z_v$ are given as

$$\mathcal{A}_v=\frac{1}{\eta}\sum_{b=1}^{\eta}\left|z_{v+(b-1)D}\right|\tag{15}$$

$$\%\mathcal{R}_v=\frac{1}{\eta-1}\sum_{b=2}^{\eta}\left|z_{v+(b-1)D}-z_{v+(b-2)D}\right|.\tag{16}$$

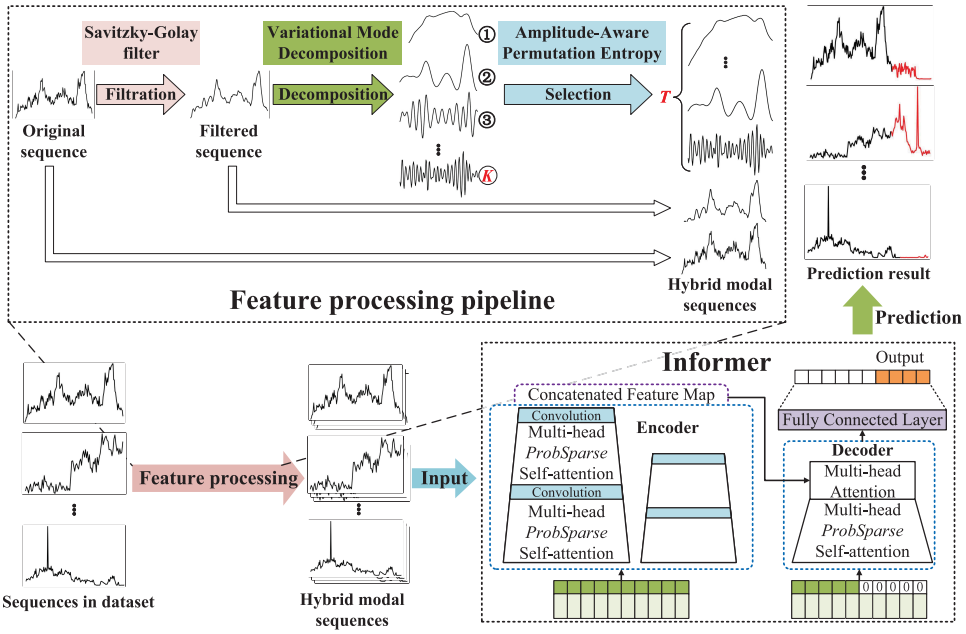



Fig. 3. Framework of SVAPI.

Consequently, the updated occurrence probability $\phi^*(\mu_g)$ of $\mu_g$ can be expressed as

$$\phi^*(\mu_g) = \frac{\sum_{v=1}^{L-(\eta-1)D} \varphi(\mu_g, \theta_v)\cdot(\beta\cdot\mathcal{A}_v + (1-\beta)\cdot\mathcal{R}_v)}{\sum_{v=1}^{L-(\eta-1)D}(\beta\cdot\mathcal{A}_v + (1-\beta)\cdot\mathcal{R}_v)} \quad (17)$$

where $\beta \in [0, 1]$ is a coefficient used to adjust the mean amplitude and amplitude deviation of the sequences. Finally, we form a set $\mathcal{G}$ consisting of $G$ values where $\phi^*(\mu_g) \geq 0$ and $G < \eta!$ Then, the AAPE value can be acquired according to the Shannon entropy, i.e.,

$$\text{AAPE}(Z, \eta, D, \beta) = \frac{-\sum_{g\in\mathcal{G}} \phi^*(\mu_g)\cdot \ln\phi^*(\mu_g)}{\ln(\eta!)}. \quad (18)$$

Based on the AAPE values, the most informative $T$ modes with higher AAPE can be selected from $K$ modes.

### E. Informer

Transformers need many computing resources and memory in training and inference [43] due to their high complexity of self-attention. The encoder suffers from memory bottlenecks, and the decoder is slow. Unlike Transformers, this work adopts the Informer [44], which addresses issues in Transformers. As shown in Fig. 3, position, time, and value encodings are performed to enrich feature representation before sequences enter the encoder-decoder. The self-attention mechanism in Informer is a multihead sparse self-attention, and it reduces complexity and increases efficiency by limiting each location to be associated with only a subset of locations. This reduces computational and storage complexity while maintaining presentation ability. It can be given as

$$A(Q, \vec{K}, V) = \text{Softmax}\left(\frac{\bar{Q}\vec{K}^T}{\sqrt{d}}V\right) \quad (19)$$

where $Q$, $\vec{K}$, and $V$ are query, key, and value matrices, respectively. $d$ is the input dimension, and Softmax is the activation function. $\bar{Q}$ is a query matrix reducing the probability of $Q$ and containing only important attention query matrices.

Informer trains a lightweight sparse self-attention block, which captures key features of the output of the sparse self-attention block' and constructs a focused feature map from layer $j$ to layer $j+1$, i.e.,

$$X_{j+1}^t = \text{MaxPool}\left(\text{ELU}\left(\text{Convld}\left(\left[X_j^t\right]_{AB}\right)\right)\right) \quad (20)$$

where Convld, ELU, and MaxPool denote the convolution operation, the activation function, and the pooled operation. $[X_j^t]_{AB}$ is the process of sparse self-attention blocks.

Informer introduces a masking mechanism into the self-attention, which restricts each location to its previous locations. It effectively deals with the autoregressive problem in decoders and ensures that its output conforms to the timing property of the sequence. It enables the decoder to generate the output step by step, predicting the value of the next position and achieving the ordered generation of the sequence. It generates the output sequence by utilizing features and input sequence of the encoder input, which can be given as

$$X_{de} = \text{Concat}\left(X_{\text{token}}^t, X_0^t\right) \in R^{L_{\text{tocken}}+L_y\times d_{\text{model}}} \quad (21)$$

where $X_{\text{token}}^t$ and $X_0^t$ are the start token and the target placeholder.

### F. SVAPI

SVAPI is designed by combining the above-mentioned components. Its overall architecture is shown in Fig. 3. In this framework, the SG filter, VMD, and AAPE are employed for the feature processing on the original sequences in dataset $\mathcal{D}$. SVAPI uses the SG filter to eliminate noise and fluctuations from the original sequences and yields a smoothed

time series to VMD. VMD explores temporal characteristics through the frequency domain information, enhancing data representation. VMD yields multiple decomposed modes, the significance of which is evaluated with AAPE. AAPE selects the most informative decomposed modes. Then, the Informer adopts the decomposed modes to use enhanced features to predict the target sequence precisely. Specifically, during the feature processing stage, each sequence $S^i$ undergoes the SG filtering for noise reduction and smoothing, followed by the VMD, resulting in extracting features comprising $K$ modes. Subsequently, AAPE is used to assess the information of the modes and selects the most informative $T$ features from the limited modes, constructing hybrid modal sequences alongside the original and filtered features. Finally, each original sequence corresponds to a hybrid modal sequence containing $T + 2$ sequences. Upon completing the feature processing stage, the dataset encompasses a total of $(T + 2)m$ temporal features, which are utilized for training Informer for prediction. The $(T + 2)m$ sequences are aligned in time and fed into Informer, each representing a distinct feature. SVAPI adopts the ProbSparse self-attention mechanism, significantly reducing time complexity and memory usage. It adopts the self-attention distilling method, highlighting dominating attention by halving the cascading layer input and efficiently processing the extremely long input sequences. In addition, it predicts the long time series at one forward operation, which significantly enhances the inference speed of long-sequence predictions.

### G. Complexity Analysis

In this work, $L$ denotes the length of the sequence. The complexity of the SG filter, VMD, and AAPE is $\mathcal{O}(Lw^2)$, $\mathcal{O}(LfK)$ and $\mathcal{O}(N)$, respectively, where $f(\cdot)$ denotes a function related to the number of decomposition layers. Improved from Transformers, SVAPI reduces its complexity to $\mathcal{O}(L\log L)$. Therefore, the complexity of SVAPI is $\mathcal{O}((w^2 + f(K) + \log L)L + N)$.

## IV. Performance Evaluation

This section presents the datasets and the parameter setting. Then, SVAPI is compared with its state-of-the-art peers, and ablation experiments are shown to predict workload and resource utilization in cloud computing systems.

### A. Datasets

This work focuses on predicting workload and resource utilization, and we adopt the Google cluster-utilization traces v3[1] as the primary datasets. These traces record the workloads running on Google compute cells managed by eight internal cluster management system units in May 2019. We aggregate the data into five-minute intervals and compute the average task arriving rates, the average CPU utilization, and the average memory utilization for a subset of these traces, forming small datasets **Gcum1** and **Gcum2** for training purposes, as illustrated in Fig. 4. Furthermore, this work conducts the comparative analysis by comparing SVAPI with state-of-the-art algorithms, including Autoformer and Informer, by using the widely recognized dataset of **ETT**[2] (Electricity Transformer Temperature) [44]. ETT encompasses two years of oil temperature data and six power load features collected from distinct counties in China. The datasets used in this work are divided into training, validation, and test sets in chronological order by the ratio of 7:2:1. These subsets are employed for the model training, the hyperparameter tuning, and the overfitting prevention, and assessing the performance of each prediction model, respectively.

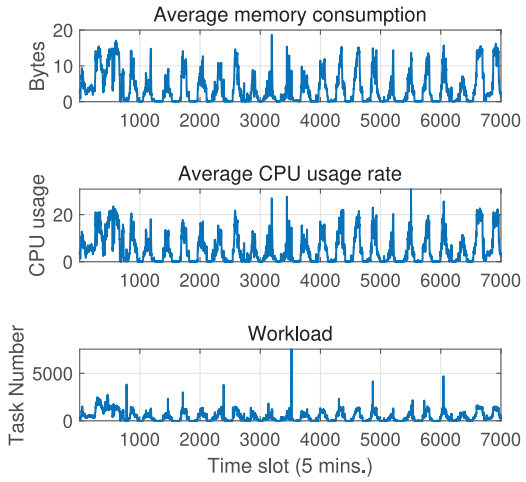



Fig. 4. Gcum1 dataset.

### B. Simulation Settings

This work presents a comprehensive set of ablation and comparison experiments to demonstrate the performance of SVAPI. We compare SVAPI with Autoformer, a cutting-edge decomposition architecture featuring self-attention mechanisms tailored for addressing complex time series problems. Furthermore, SG-VMD-Informer (SVI), SG-Informer (SI), and the original Informer serve as state-of-the-art benchmark peers for conducting ablative experiments. All algorithms are independently repeated 10 times, and the best-performing ones are selected through a testing set to yield the final comparison results. The reasons of choosing them are described as follows.

1) *Autoformer [14]:* This innovative decomposition architecture integrates self-attention mechanisms and demonstrates versatility across various applications. A comparative analysis between SVAPI and Autoformer underscores SVAPI's efficiency in predicting highly variable workloads in CDCs.
2) *Informer [44]:* Serving as the foundational component of SVAPI, Informer has low memory utilization and superior long-term prediction capabilities compared to Transformer. The comparison between SVAPI and Informer provides concrete evidence of SVAPI's improved prediction performance.
3) *SI [45]:* The SG filter enhances the prediction stability by smoothing and reducing noise in the time series

[1]https://github.com/google/cluster-data/tree/master

[2]https://paperswithcode.com/dataset/ett

TABLE II
PARAMETER SETTING OF EACH MODULE IN SVAPI

| Modules | Parameters | Values |
|---|---|---|
| SG filter | $w$ | 7 |
| | $R$ | 3 |
| VMD | $\alpha$ | 1000 |
| | $K$ | 5 |
| | $\tau$ | 0.1 |
| | $\varepsilon$ | $10^{-6}$ |
| AAPE | $\eta$ | 3 |
| | $D$ | 1 |
| | $\beta$ | 0.5 |
| Informer (Autoformer) | Embedding dimension | 512 |
| | Layer number of encoder | 2 |
| | Layer number of decoder | 1 |
| | Prob factor (Factor) | 5 |
| | Head number | 8 |
| | Training epoch number | 3 |
| | Fully convolutional network dimension | 2048 |
| | Dropout rate | 0.05 |
| | Activation function | GELU |
| | Batch size | 32 |
| | Loss function | MSE |
| | Learning rate | 0.0001 |

data. As an ablation algorithm, SI improves Informer's prediction ability brought by the SG filter. The comparison between SVAPI and SI proves that the combination of VMD and AAPE benefits the optimization of the model.

4) *SG-VMD-Informer (SVI) [46]:* VMD contributes multiple modal features, enhancing the model's fitting capacity. As another ablation algorithm, SVI demonstrates the performance improvement brought by VMD. The comparison between SVAPI and SVI shows the prediction improvement combined with AAPE.

Generating offloading schemes in a cloud computing system typically requires the computation time is in minutes. Therefore, this work aims to predict the arrival rate of tasks and resource utilization for the next hour. Unless otherwise specified, all experiments use samples in 96 points to predict the future 24 points (24 points represent one hour in the Gcum dataset). The parameter settings for the same module across all algorithms are shown in Table II, including specific configurations of the SG filter, VMD, and AAPE. This work utilizes a grid search method [47] to combine the possible values of hyperparameters into a grid. We exhaustively search all possible combinations, cross-validate each combination, and evaluate the model performance on the verification set. Finally, the combination of hyperparameters with the best performance on the validation set is selected as parameters of the final model. All experiments adopt PyTorch in a computer with an RTX2080Ti GPU, 64-GB memory, and an Intel Core i7-12700 CPU with 12-core 4.9-GHz processors.

### C. Experimental Results

Fig. 5 shows the prediction results with different $\alpha$. It is shown that mean squared error (MSE) is the smallest and $R^2$ is the biggest when $\alpha = 1000$. This work performs experiments with different $K \in \{3, 4, \ldots, 10\}$ to determine the best $K$ [22].

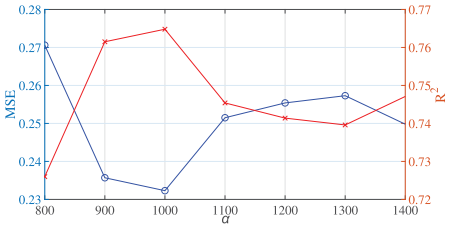



Fig. 5. Prediction result of SVAPI under different $\alpha$.

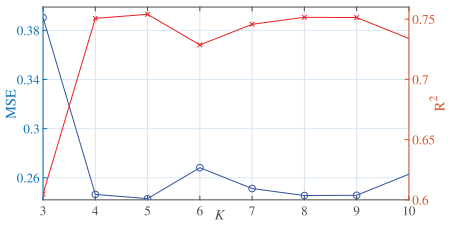



Fig. 6. Prediction result of SVAPI under different $K$.

Fig. 6 shows that MSE is the smallest and $R^2$ is the biggest when $K = 5$.

Table III compares the prediction results on the Google cluster data with different mode selection strategies.

1) *SVI:* It does not adopt the AAPE-based mode selection in SVAPI.
2) *SVAPI:* It selects seven modes with the highest AAPE values.
3) *SVAPI$^{\dagger}$:* It selects seven modes with the lowest AAPE values.

We vary the mode number to distinguish the performance difference among modes with the highest and the lowest AAPE. VMD decomposes each feature into 10 modes, and seven of these modes are chosen as a part of the input for the training. To evaluate the performance of SVAPI and peer algorithms, the MSE, the mean absolute error (MAE), the root mean squared error (RMSE), and the coefficient of determination ($R^2$) are used as metrics, i.e.,

$$\text{MSE} = \frac{1}{\kappa}\sum_{\gamma=1}^{\kappa}\left(y_\gamma - \hat{y}_\gamma\right)^2 \tag{22}$$

$$\text{MAE} = \frac{1}{\kappa}\sum_{\gamma=1}^{\kappa}|y_\gamma - \hat{y}_\gamma|^2 \tag{23}$$

$$\text{RMSE} = \sqrt{\frac{1}{\kappa}\sum_{\gamma=1}^{\kappa}\left(y_\gamma - \hat{y}_\gamma\right)^2} \tag{24}$$

$$R^2 = 1 - \frac{\sum_{\gamma=1}^{\kappa}\left(y_\gamma - \hat{y}_\gamma\right)^2}{\sum_{\gamma=1}^{\kappa}\left(y_\gamma - \bar{y}\right)^2} \tag{25}$$

where $\kappa$ denotes the number of samples, $y_\gamma$ denotes the ground-truth value of sample $\gamma$, $\hat{y}_\gamma$ denotes its predicted value, and $\bar{y}$ denotes the average value of all samples. Smaller MSE, MRSE, and MAE values indicate lower model errors, while larger $R^2$ indicates more accurate prediction results. The logical $p$-value of 1 represents that our algorithm is significantly different from the compared algorithm. For each

TABLE III
COMPARISON OF DIFFERENT MODE SELECTION STRATEGIES ON GOOGLE CLUSTER DATA

| Methods | | SVI | | | | SVAPI | | | | SVAPI† | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | | MAE | MSE | RMSE | $R^2$ | MAE | MSE | RMSE | $R^2$ | MAE | MSE | RMSE | $R^2$ |
| Gcum1 | Workload | 0.137 | 0.227 | 0.227 | 0.863 | **0.119** | **0.218** | **0.345** | **0.881** | 0.145 | 0.239 | 0.381 | 0.855 |
| | CPU | 0.085 | 0.209 | 0.292 | 0.942 | **0.083** | **0.215** | **0.288** | **0.944** | 0.097 | 0.236 | 0.312 | 0.934 |
| | Memory | 0.085 | 0.216 | 0.291 | 0.941 | **0.070** | **0.193** | **0.264** | **0.951** | 0.098 | 0.218 | 0.313 | 0.931 |
| Gcum2 | Workload | 0.131 | 0.210 | 0.362 | 0.223 | **0.128** | **0.185** | **0.357** | **0.243** | 0.139 | 0.192 | 0.373 | 0.176 |
| | CPU | 0.223 | 0.303 | 0.472 | 0.636 | **0.180** | **0.251** | **0.424** | **0.707** | 0.255 | 0.334 | 0.505 | 0.584 |
| | Memory | 0.303 | 0.381 | 0.550 | 0.728 | **0.276** | **0.368** | **0.526** | **0.751** | 0.321 | 0.372 | 0.567 | 0.711 |
| **Count** | | 0 | | | | 24 | | | | 0 | | | |

algorithm, we record its times of yielding the best result of each metric in the **Count** row. SVAPI achieves the best performance compared to SVI, while the prediction error with SVAPI† is increased. These results prove that AAPE and selecting modes enhance SVAPI's prediction performance because the modes with higher AAPE values include more information, which is beneficial for SVAPI in predicting the task arriving rate and resource demand.

Table IV summarizes the comparison results of all algorithms on 6 datasets. There are 10 prediction tasks, each utilizing all features from the dataset as the input to predict the target sequence. Among them, we predict the workload, CPU, and memory utilization given the Gcum1 and Gcum2 datasets, while we predict the oil temperature given the ETT dataset. The best results are highlighted in bold in Table IV. Finally, we compute the average values of each algorithm for each metric. For each algorithm, we record its times of yielding the best result of each metric in the **Count** row.

Table IV shows that SVAPI significantly outperforms other algorithms, obtaining 26 out of 30 best results. $p$-values for all compared algorithms are all one, proving that SVAPI is significantly different. For the **Average** row, SVAPI exhibits the average reduction of 37.7% in MSE compared to the original Informer (0.314→0.194). The ablation experiment results reveal that each module's features improve SVAPI's prediction performance. Specifically, the SG filter improves the prediction performance by 0.6% (0.314→0.312), VMD enhances the prediction performance by 34.9% (0.312→0.203), and AAPE boosts the prediction performance by 4.4% (0.203→0.194).

For the ETTm1 dataset, models face greater fitting challenges due to significant fluctuations in the data amplitude. Therefore, richer features are needed to characterize the dataset effectively. However, in SVAPI, the features extracted by VMD are selectively filtered by AAPE, reducing input features into the model. This reduction diminishes SVAPI's fitting capability for this dataset compared to SVI.

To validate SVAPI's prediction performance on the long time series, we adopt the ETTm2 dataset by taking 96 points as the input and predicting the future 96 points in the target sequence. The results show that in the long-time series prediction, SVAPI has a performance improvement of 16.3% in terms of MSE (0.178→0.149), which is higher than the improvement of 4.4% observed in the short-time series prediction. Therefore, SVAPI has the strong ability to predict the long time series.

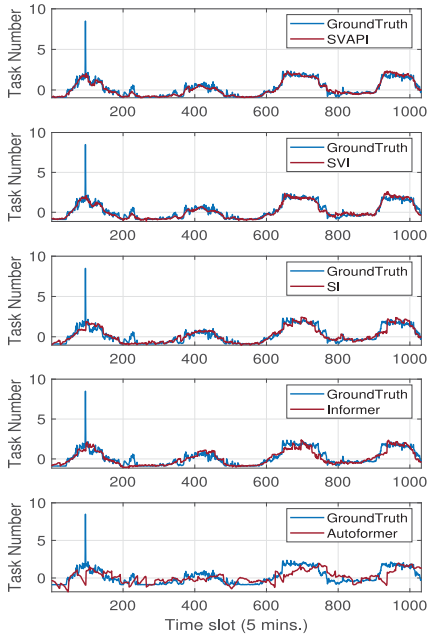



Fig. 7. Prediction results of different algorithms on the Gcum1 test dataset.

Fig. 7 illustrates the prediction results of five algorithms for the entire test set of the Gcum1 dataset. All algorithms do not exhibit overfitting for the outlier at point 90. Both SVAPI and SVI demonstrate high prediction accuracy. Compared to the other three models, SVAPI and SVI accurately predict the peak near point 200 and the upward trend around point 930. However, it is worth noting that SVAPI's predictions are more accurate for some turning points (such as the fluctuations between points 750 and 800). Additionally, SVAPI mitigates overshooting issues and more accurately predicts sequence transitioning from the ascent or descent trends to stability (such as the turning point near point 930) compared to SVI. SI provides more accurate predictions of the plateau sequence between points 600 and 800 than Informer. These results demonstrate that the SG filter, VMD, and AAPE modules all enhance the prediction performance of SVAPI. Autoformer's fitting performance is relatively poor, capturing the sequence's trend changes but exhibiting significant lag.

Fig. 8 presents the prediction result by selecting the test subset from the Gcum1 dataset. The black lines represent the ground truth values, while the red lines depict each model's prediction result. SVAPI achieves the best prediction results

TABLE IV
COMPARISON RESULTS OF DIFFERENT ALGORITHMS ON GOOGLE CLUSTER DATA AND TYPICAL DATASETS

| Algorithms | | SVAPI | | | SVI | | | SI | | | Informer | | | Autoformer | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | | MSE | MAE | $R^2$ | MSE | MAE | $R^2$ | MSE | MAE | $R^2$ | MSE | MAE | $R^2$ | MSE | MAE | $R^2$ |
| Gcum1 | Workload | **0.197** | **0.113** | **0.887** | 0.215 | 0.123 | 0.877 | 0.324 | 0.245 | 0.756 | 0.331 | 0.256 | 0.744 | 0.573 | 0.606 | 0.394 |
| | CPU | **0.229** | **0.106** | **0.924** | 0.239 | 0.123 | 0.911 | 0.365 | 0.296 | 0.787 | 0.369 | 0.326 | 0.765 | 0.693 | 0.796 | 0.426 |
| | Memory | **0.190** | **0.075** | **0.947** | 0.201 | 0.078 | 0.945 | 0.423 | 0.381 | 0.733 | 0.396 | 0.347 | 0.757 | 0.651 | 0.734 | 0.484 |
| Gcum2 | Workload | **0.220** | **0.150** | **0.359** | 0.224 | 0.153 | 0.345 | 0.290 | 0.220 | 0.060 | 0.269 | 0.192 | 0.179 | 0.371 | 0.275 | -0.179 |
| | CPU | **0.273** | **0.162** | **0.783** | 0.289 | 0.182 | 0.757 | 0.407 | 0.381 | 0.491 | 0.399 | 0.377 | 0.497 | 0.658 | 0.737 | 0.012 |
| | Memory | **0.263** | **0.178** | **0.765** | 0.277 | 0.179 | 0.764 | 0.430 | 0.407 | 0.464 | 0.441 | 0.365 | 0.519 | 0.652 | 0.725 | 0.043 |
| ETTh1 | | 0.174 | **0.053** | **0.550** | **0.173** | 0.053 | 0.547 | 0.189 | 0.066 | 0.434 | 0.198 | 0.069 | 0.411 | 0.285 | 0.128 | -0.102 |
| ETTh2 | | **0.174** | **0.055** | **0.893** | 0.179 | 0.056 | 0.891 | 0.297 | 0.158 | 0.694 | 0.307 | 0.164 | 0.682 | 0.368 | 0.234 | 0.554 |
| ETTm1 | | 0.068 | 0.008 | 0.929 | **0.055** | **0.005** | **0.955** | 0.129 | 0.031 | 0.737 | 0.146 | 0.034 | 0.709 | 0.137 | 0.036 | 0.686 |
| ETTm2 (96-96) | | **0.149** | **0.053** | **0.899** | 0.178 | 0.071 | 0.864 | 0.269 | 0.133 | 0.745 | 0.285 | 0.158 | 0.699 | 0.345 | 0.205 | 0.611 |
| **Average** | | **0.194** | **0.095** | **0.794** | 0.203 | 0.102 | 0.786 | 0.312 | 0.232 | 0.590 | 0.314 | 0.229 | 0.596 | 0.473 | 0.448 | 0.239 |
| **Count** | | 26 | | | 4 | | | 0 | | | 0 | | | 0 | | |
| ***p*-value** | | N/A | | | 1 | | | 1 | | | 1 | | | 1 | | |

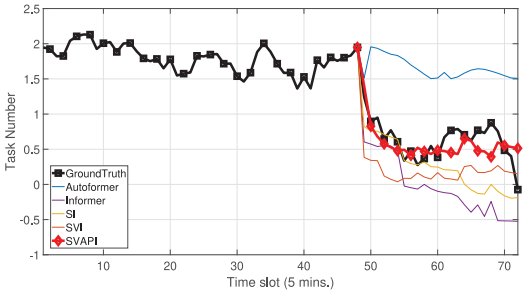



Fig. 8. Prediction result from the Gcum1 dateset under the input-96-predict-24 setting.

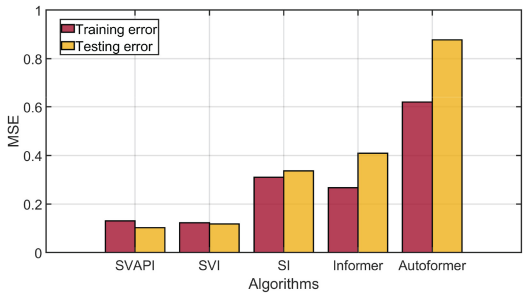



Fig. 9. Comparison of training errors and testing ones of different algorithms for the Gcum1 dataset.

and captures the descent trend better. Although SVI identifies the trend of task arriving rate, it suffers from the overshooting issue. SI and Informer exhibit similar predictions of the descent trend in the initial points but fail to forecast longer-term results due to insufficient features. Notably, the denoising features provided by the SG filter enable SI to better capture the turning points in the descent of task numbers compared to Informer. Autoformer, with the same network parameters as Informer, fails to predict the results accurately.

Fig. 9 shows the comparison of overfitting of all algorithms for the Gcum1 dataset. Five epochs of training are performed. MSE is used to compare five algorithms. Fig. 9 shows that testing errors of SVAPI and SVI are lower than their training errors, indicating an absence of overfitting. Conversely, the testing errors of Informer and Autoformer are approximately

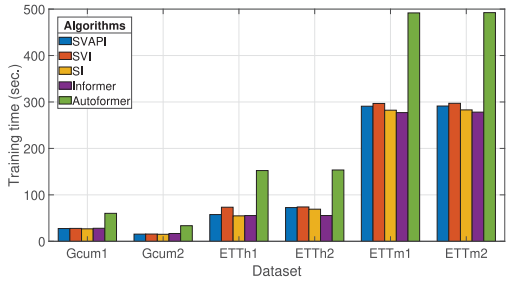



Fig. 10. Training time of five algorithms for typical datasets.

30% to 40% higher than their training errors, meaning a notable overfitting issue. The reason is that SVAPI and SVI adopt the VMD module, which enriches features to improve the prediction accuracy and alleviates the overfitting problem.

Fig. 10 compares the training time of five algorithms given six datasets. It is observed that across all datasets, SVAPI exhibits a shorter training time than SVI after feature selection with AAPE from the features generated by the VMD decomposition. This demonstrates the effectiveness of AAPE in reducing the training time. SVAPI's training time is much lower than Autoformer's because of its simple yet efficient feature-capturing ability.

It is worth noting that SVAPI requires significant computational resources during the training phase, similar to other deep learning models. Nevertheless, the trained SVAPI has high-speed computation with negligible time overhead. In cloud computing systems, the predicted workload with SVAPI at each time step can yield high-quality task offloading, enhancing real-time performance. In summary, SVAPI excels in prediction accuracy with minimal risk of overfitting and has more time efficiency during the training than Autoformer because of progressive decomposition.

## V. CONCLUSION AND FUTURE WORK

Workload prediction is paramount for organizations seeking to implement proactive service operation management in CDCs, ensuring high-quality computing services and sustainable costs. However, the inherent nonlinearity, high volatility, and interdependence of workload and resource utilization

data in cloud computing present significant challenges in thoroughly analyzing limited data. This work proposes a data-enhanced high-precision prediction model named SVAPI, which integrates the SG filter, the VMD, the AAPE, and the Informer model. SVAPI leverages the SG filter to eliminate noise and fluctuations from the original sequences in cloud computing while preserving essential features. It then applies VMD to explore the temporal characteristics of these sequences with frequency domain information, thereby enhancing data representation. Additionally, AAPE evaluates mode significance, facilitating the selection of the most informative decomposed modes. These operations strengthen features, aiding the model in completing prediction tasks in CDCs that require comprehensive feature extraction. Finally, the Informer is integrated to enable the comprehensive analysis of enhanced features, ensuring accurate prediction of the target sequence in CDCs. Experiments with real-world datasets in CDCs demonstrate that integrating each module contributes positively to SVAPI's performance enhancement. Specifically, it improves the prediction accuracy of tasks and resource usage in CDCs by 37.7% and 65.67% on average compared with Informer and Autoformer, respectively. It accurately forecasts task arrival rates and resource demands in CDCs, providing sufficient time for proactive planning and execution phases of task offloading, thereby ensuring both QoS and cost efficiency.

In the future, we will employ advanced methods, such as deep learning, to select the optimal combination of decomposed modes. We will exploit dynamic sparse patterns in attention and decrease unnecessary computations to enhance efficiency, analyze different parallelism levels, and design a more scalable system architecture by adopting parallel dataflow with balancing and hardware-enabled execution. Besides, we will design distributed training methods that optimize computation and communication to improve scalability, thereby reducing training time. We will explore the explainability of time series prediction algorithms by revealing hidden causes of model errors specific to both model inner logic and the currently observed input. Specifically, we will assign relevance scores to highly diverse input time series by investigating the Pearson correlation between relevance scores and enriched expressiveness of the input.

## REFERENCES


[1] Y. Xia, M. Zhou, X. Luo, Q. Zhu, J. Li, and Y. Huang, "Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 162–170, Jan. 2015.

[2] H. Yuan, J. Bi, J. Zhang, and M. Zhou, "Energy consumption and performance optimized task scheduling in distributed data centers," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 9, pp. 5506–5517, Sep. 2022.

[3] J. Bi, H. Yuan, K. Xu, H. Ma, and M. Zhou, "Large-scale network traffic prediction with LSTM and temporal convolutional networks," in *Proc. Int. Conf. Robot. Autom.*, 2022, pp. 3865–3870.

[4] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Forecasting cloud application workloads with cloudinsight for predictive resource management," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1848–1863, Sep. 2022.

[5] J. Bi et al., "Multi-swarm genetic gray wolf optimizer with embedded autoencoders for high-dimensional expensive problems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 7265–7271.

[6] J. Bi, H. Yuan, J. Zhang, and M. Zhou, "Green energy forecast-based bi-objective scheduling of tasks across distributed clouds," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 3, pp. 619–630, Jul.–Sep. 2022.

[7] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.

[8] K. Chargui, T. Zouadi, V. R. Sreedharan, A. E. Fallahi, and M. Reghioui, "A novel proactive–reactive scheduling approach for the quay crane scheduling problem: A VUCA perspective," *IEEE Trans. Eng. Manag.*, vol. 70, no. 7, pp. 2594–2607, Jul. 2023.

[9] W. Liu, H. Wang, Z. Xi, and R. Zhang, "Smooth deep learning magnetotelluric inversion based on physics-informed swin transformer and multiwindow Savitzky–Golay filter," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 4505214.

[10] Z. Xing, Y. He, X. Wang, K. Shao, and J. Duan, "VMD-IARIMA-based time series forecasting model and its application in dissolved gas analysis," *IEEE Trans. Dielectrics Electr. Insul.*, vol. 30, no. 2, pp. 802–811, Apr. 2023.

[11] H. Azami and J. Escudero, "Amplitude-aware permutation entropy: Illustration in spike detection and signal segmentation," *Comput. Methods Programs Biomed.*, vol. 128, pp. 40–51, May 2016.

[12] X. Fan, P. Gao, X. Gao, and Y. Huang, "Prediction of weld widths for laser-MIG hybrid welding using informer model," *IEEE Trans. Ind. Electron.*, vol. 71, no. 6, pp. 6221–6230, Jun. 2024.

[13] J. Bi, H. Ma, H. Yuan, and J. Zhang, "Accurate prediction of workloads and resources with multi-head attention and hybrid LSTM for cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 3, pp. 375–384, Jul.–Sep. 2023.

[14] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proc. 35th Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 7265–7271.

[15] D. Saxena, A. K. Singh, and R. Buyya, "OP-MLB: An online VM prediction-based multi-objective load balancing framework for resource management at cloud data center," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2804–2816, Dec. 2022.

[16] I. Kumara, M. H. Ariz, M. B. Chhetri, M. Mohammadi, W.-J. V. D. Heuvel, and D. Tamburri, "FOCloud: Feature model guided performance prediction and explanation for deployment configurable cloud applications," *IEEE Trans. Services Comput.*, vol. 16, no. 1, pp. 302–314, Jan./Feb. 2023.

[17] Y. Xie et al., "Real-time prediction of docker container resource load based on a hybrid model of ARIMA and triple exponential smoothing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 1386–1401, Apr.–Jun. 2022.

[18] A. K. Singh, D. Saxena, J. Kumar, and V. Gupta, "A quantum approach towards the adaptive prediction of cloud workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 12, pp. 2893–2905, Dec. 2021.

[19] H. Moradi, W. Wang, and D. Zhu, "Online performance modeling and prediction for single-VM applications in multi-tenant clouds," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 97–110, Mar. 2023.

[20] S. Ilager, K. Ramamohanarao, and R. Buyya, "Thermal prediction for efficient energy management of clouds using machine learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1044–1056, May 2021.

[21] D. Chen, X. Zhang, L. Wang, and Z. Han, "Prediction of cloud resources demand based on hierarchical pythagorean fuzzy deep neural network," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 1890–1901, Dec. 2021.

[22] J. Bi et al., "Multivariate resource usage prediction with frequency-enhanced and attention-assisted transformer in cloud computing systems," *IEEE Internet Things J.*, vol. 11, no. 15, pp. 26419–26429, Aug. 2024.

[23] A. C. Maroudis, T. Theodoropoulos, J. Violos, A. Leivadeas, and K. Tserpes, "Leveraging graph neural networks for SLA violation prediction in cloud computing," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 1, pp. 605–620, Feb. 2024.

[24] L. Ruan et al., "Cloud workload turning points prediction via cloud feature-enhanced deep learning," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1719–1732, Apr.–Jun. 2023.

[25] H. Yuan, J. Bi, S. Li, J. Zhang, and M. Zhou, "An improved LSTM-based prediction approach for resources and workload in large-scale data centers," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 22816–22829, Jun. 2024.

[26] J. Bi, K. Xu, H. Yuan, J. Zhang, and M. Zhou, "Network attack prediction with hybrid temporal convolutional network and bidirectional GRU," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 12619–12630, Apr. 2024.

[27] J. Gao, H. Wang, and H. Shen, "Task failure prediction in cloud data centers using deep learning," *IEEE Trans. Services Comput.*, vol. 15, no. 3, pp. 1411–1422, Jun. 2022.
[28] L. Cui, W. Li, X. Wang, D. Zhao, and H. Wang, "Comprehensive remaining useful life prediction for rolling element bearings based on time-varying particle filtering," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, Mar. 2022.
[29] 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. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 1869–1879, Jul. 2022.
[30] Q. Li, J. Lan, L. Zhang, B. Chen, and K. Zhu, "Augmented nonlinear least squares estimation with applications to localization," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 2, pp. 1042–1054, Apr. 2022.
[31] Y. Guo and Z. Zhang, "Generalized variational mode decomposition: A multiscale and fixed-frequency decomposition algorithm," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, Apr. 2021.
[32] A. Wang, P. Qin, X.-M. Sun, and Y. Li, "An automatic parameter setting variational mode decomposition method for vibration signals," *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 2053–2062, Feb. 2024.
[33] X. Cai and R.-J. Wai, "Intelligent DC arc-fault detection of solar PV power generation system via optimized VMD-based signal processing and PSO–SVM classifier," *IEEE J. Photovolt.*, vol. 12, no. 4, pp. 1058–1077, Jul. 2022.
[34] Q. Song, X. Jiang, S. Wang, J. Guo, W. Huang, and Z. Zhu, "Self-adaptive multivariate variational mode decomposition and its application for bearing fault diagnosis," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–13, Jan. 2022.
[35] Y. Zhang et al., "A multi-output prediction model for physical machine resource usage in cloud data centers," *Future Gener. Comput. Syst.*, vol. 130, pp. 292–306, May 2022.
[36] J. Guo et al., "Variational mode decomposition for NMR echo data denoising," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 5902014.
[37] C. Pu, H. Huang, and Y. Li, "Aggregated-attention transformation network for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 15, pp. 5674–5688, Jul. 2022.
[38] M. U. Zahid, M. D. Nisar, M. H. Shah, and S. A. Hussain, "Specific emitter identification based on multi-scale multi-dimensional approximate entropy," *IEEE Signal Process. Lett.*, vol. 31, pp. 850–854, Mar. 2024.
[39] H.-J. Chan et al., "Ultrasound sample entropy imaging: A new approach for evaluating hepatic steatosis and fibrosis," *IEEE J. Transl. Eng. Health Med.*, vol. 9, pp. 1–12, 2021.
[40] M. Aggarwal, "On agent-specific fuzzy entropy functions," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 1, pp. 2–11, Jan. 2023.
[41] Y. Sun, Y. Cao, G. Xie, and T. Wen, "Sound based fault diagnosis for RPMs based on multi-scale fractional permutation entropy and two-scale algorithm," *IEEE Trans. Veh. Technol.*, vol. 70, no. 11, pp. 11184–11192, Nov. 2021.
[42] Z. Guo, V. M. McClelland, O. Simeone, K. R. Mills, and Z. Cvetkovic, "Multiscale wavelet transfer entropy with application to corticomuscular coupling analysis," *IEEE Trans. Biomed. Eng.*, vol. 69, no. 2, pp. 771–782, Feb. 2022.
[43] R. Agarwal, H. Li, Z. Guo, and P. Cheetham, "The effects of PWM with high dv/dt on partial discharge and lifetime of medium-frequency transformer for medium-voltage (MV) solid state transformer applications," *IEEE Trans. Ind. Electron.*, vol. 70, no. 4, pp. 3857–3866, Apr. 2023.
[44] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 1654–1662.
[45] L. Zhao, H. Yuan, K. Xu, B. Jing, and B. H. Li, "Hybrid network attack prediction with Savitzky–Golay filter-assisted informer," *Expert Syst. Appl.*, vol. 235, Jan. 2024, Art. no. 121126.
[46] Y. Wu, X. Pan, and J. Yang, "VMD-informer-DCC for photovoltaic power prediction," *IEICE Trans. Commun.*, vol. 107, no. 7, pp. 487–494, 2024.
[47] A. J. Batten, J. Thorpe, R. I. Piegari, and A.-M. Rosland, "A resampling based grid search method to improve reliability and robustness of mixture-item response theory models of multimorbid high-risk patients," *IEEE J. Biomed. Health Inform.*, vol. 24, no. 6, pp. 1780–1787, Jun. 2020.

**Haitao Yuan** (Senior Member, IEEE) received the Ph.D. degree in computer engineering from New Jersey Institute of Technology, Newark, NJ, USA, in 2020.

He is currently a Deputy Director with the Department of Science and Technology Innovation, Wenchang International Aerospace City, Hainan, China. He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. He is named in the world's top 2% of Scientists List. His research interests include the Internet of Things, edge computing, deep learning, data-driven optimization, and computational intelligence algorithms.

Dr. Yuan received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, the Best Paper Award in the 17th ICNSC, and the Best Student Paper Award Nominees in 2024 IEEE SMC. He is an Associate Editor for IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART I: SYSTEMS, IEEE INTERNET OF THINGS JOURNAL, and *Expert Systems With Applications*.

**Qinglong Hu** (Student Member, IEEE) received the bachelor's degree in automation from Shandong University, Jinan, China, in 2021, and the master's degree from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, in 2024. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong, Hong Kong.

His research interests include deep learning to optimization, evolutionary computing, big data analysis, machine learning, and large language models.

**Meijia Wang** (Student Member, IEEE) received the B.S. degree in automation from Beihang University, Beijing, China, in 2022, where she is currently pursuing the master's degree with the School of Automation Science and Electrical Engineering.

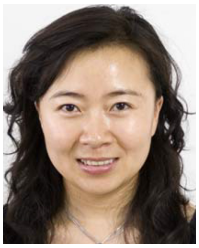
Her research interests include cloud computing, edge computing, energy management, big data, machine learning, and deep learning.

Ms. Wang received the Excellent Student Cadre in 2019, 2020, and 2021, the Outstanding Graduate in 2022, the Merit Student in 2023 and 2024, and the National Scholarship of China in 2024.

**Shen Wang** (Student Member, IEEE) received the B.S. degree in quality and reliability of aircraft from Beihang University, Beijing, China, in 2022, where he is currently pursuing the master's degree with the School of Automation Science and Electrical Engineering.

His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms.

Mr. Wang was the recipient of the Merit Student of Beihang University in 2019.



**Jing Bi** (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively.

She is a Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. Her research interests include cloud computing, cloud computing, large-scale data analytics, and performance optimization.

Prof. Bi received the IBM Fellowship Award, the Best Paper Award at the 17th IEEE International Conference on Networking, Sensing and Control, and the First-Prize Progress Award of the Chinese Institute of Simulation Science and Technology. She is an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART I: SYSTEMS.

**Rajkumar Buyya** (Fellow, IEEE) received the B.E degree in computer science and engineering from Mysore University, Mysuru, India, in 1992, the M.E degree in computer science and engineering from Bangalore University, Bengaluru, India, in 1995, and the Ph.D. degree in computer science and software engineering from Monash University, Melbourne, VIC, Australia, in 2002.

He is a Redmond Barry Distinguished Professor and the Director with the Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Melbourne. He was a Future Fellow of the Australian Research Council from 2012 to 2016. He has authored over 800 publications and seven textbooks.

Dr. Buyya is one of the Highly Cited Authors in Computer Science and Software Engineering worldwide, with over 146 470 citations and an h-index of 166. He was recognized as a Web of Science Highly Cited Researcher from 2016 to 2021 by Thomson Reuters.

**Shuyuan Shi** received the Ph.D. degree in electircal and computer engineering from the National University of Singapore, Singapore in 2020.

She is currently a Full Professor with the School of Integrated Circuit Science and Engineering, Beihang University, Beijing, China. Her research interests include novel spintronic materials, devices, and in-memory computing applications.

Prof. Shi was awarded the Institute of Microelectronics Prize 2019 in Singapore. She has been recognized as an Excellent Young Scientist (Overseas) in 2022. She serves as a member of the editorial board for Moore and More.

**Jinhong Yang** received the Ph.D. degree in computer application technology from Harbin Engineering University, Harbin, Heilongjiang, China, in 2017.

She is currently a Senior Engineer with CSSC Systems Engineering Research Institute, Beijing, China. Her research interests include machine learning, data mining, knowledge reasoning, deep learning, and intelligent optimization.

Dr. Yang serves as a Reviewer for *Expert Systems With Applications* and *International Journal of Machine Learning and Cybernetics*.

**Jia Zhang** (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2000.

She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering, a Professor with the Department of Computer Science in the Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA. 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 graphs, and interdisciplinary applications of all of these interests in earth science.

**Mengchu Zhou** (Fellow, IEEE) received the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, where he has been a Distinguished Professor since 2013. He has 1300+ publications including 17 books, 900+ journal papers (680+ in IEEE Trans.), 31 patents, and 32 book-chapters. His interests are in Petri nets, automation, robotics, big data, Internet of Things, cloud/edge computing, and AI.

Dr. Zhou is Fellow of IFAC, AAAS, CAA, and NAI.